

Algoritmo de Shor e sua aplicação à fatoração de números inteiros

Adriana Xavier Freitas

Fevereiro 2010

Algoritmo de Shor e sua aplicação à fatoração de números inteiros

Adriana Xavier Freitas

Orientador: Prof. Marcelo de Oliveira Terra Cunha

Dissertação apresentada à UNIVERSIDADE
FEDERAL DE MINAS GERAIS, como requisito
parcial para a obtenção do grau de mestre em
matemática.

Fevereiro de 2010

Aos meus pais e irmãs.

Agradecimentos

A Deus por tudo que aconteceu em minha vida, permitindo que eu conseguisse realizar o meu sonho de estudar na UFMG e fazer mestrado com bolsa.

Aos meus pais por todo amor que me dedicaram e pela compreensão da minha ausência em suas vidas.

As minhas irmãs que eu muito amo, em especial a Ayessa, razão do meu viver.

As minhas colegas de república Neila e Adriana por terem tido paciência em me ouvir dizer inúmeras vezes nesses últimos meses: 'será que vou entrar no doutorado?' Em especial a Adriana por sempre me salvar na véspera das provas daquelas dúvidas cruciais e pelo companheirismo desses 5 anos.

Ao Hέλvecio meu amor querido que sempre ameniza as saudades que sinto da minha família, me compreende nas vésperas das provas e por todas as figuras desse texto.

Ao Marcelo pela orientação nesses últimos 3 anos em especial pela sua paciência e generosidade de ler o meu texto cheio de erros de português e por ter ajudado a torná-lo algo digno de chamar de dissertação.

Aos professores que passaram pela minha vida, tantos que é impossível citar seus nomes. Mas, não posso me esquecer do Bernardo, sem ele hoje não estaria aqui. Foi ele que desde o segundo período da graduação me incentivou a fazer mestrado, sempre que eu estava prestes a desistir por causa das dificuldades conversava com ele e tinha ânimo para continuar. Sem dúvidas sem ele e a minha família eu não estaria aqui.

A todos os amigos da graduação que hoje são para mim mais que amigos, já fazem parte da minha família; se há um nome desse período que não posso esquecer é o da Silvine, minha amiga e companheira para toda hora.

Aos amigos que fiz durante o mestrado que são muito especiais.

A Clélia e a Cláudia. Não posso esquecer do pessoal da Olimpíada Mi-

neira: Seme, Fábio, Mário Jorge e todos os monitores que por aí passaram. Saudades de todos. Aos integrantes da OBMEP, que são muitos, por isso nem me atrevo a citar nomes.

Aos funcionários do Departamento de Matemática, em especial ao Valdney e a Andréa que sempre nos socorrem com as burocracias.

Ao CNPQ por ter me tornado um rica bolsista de mestrado e pelo incentivo à pesquisa.

Não poderia esquecer do Lula por um único motivo: ele apoiou a criação da OBMEP.

Resumo

O algoritmo de Shor é um algoritmo quântico que encontra com alta probabilidade a ordem de um elemento $x \in \mathbb{Z}_N^*$. Uma de suas aplicações é a construção de um algoritmo que encontra fatores de N . Nos capítulos iniciais abordaremos ferramentas necessárias para o entendimento do algoritmo de Shor, tais como: aritmética modular, algoritmos, frações contínuas, conceitos introdutórios de computação quântica e transformada quântica de Fourier. Nos capítulos seguintes apresentamos o algoritmo de Shor e sua aplicação à fatoração de números inteiros.

Abstract

Shor's algorithm is a quantum algorithm that finds with high probability the order of an element $x \in \mathbb{Z}_N^*$. One of its applications is the construction of an algorithm that finds the factors of N . In the initial chapters we approach necessary tools for the comprehension of Shor's algorithm such as: modular arithmetic, algorithms, continued fractions, basic concepts of quantum computing and Fourier quantum transform. In the following chapters we present Shor's algorithm and its application in factorization.

Sumário

Agradecimentos	i
Resumo	iii
Abstract	iv
Introdução	1
1 Noções Básicas	3
1.1 Aritmética Modular	3
1.2 Grupos	7
1.3 Algoritmo	8
1.4 Frações Contínuas	15
1.5 Álgebra linear	22
1.5.1 Produto tensorial	25
2 Um breve relato quântico	28
2.1 Comentários sobre mecânica quântica	28
2.2 Computação quântica	30
2.3 Portas Lógicas	32
2.4 Circuitos Quânticos	34
3 Transformada de Fourier Quântica	37
3.1 Complexidade da Transformada Quântica de Fourier	44

4	Algoritmo de Shor	46
4.1	Algoritmo de Shor para uma ordem igual a uma potência de 2	48
4.2	Caso Geral	51
4.3	Exemplo	59
5	Fatoração de um Número e sua Ordem	63
A	Algoritmo quântico para calcular $x^j y \pmod N$	73
	Referências Bibliográficas	77

Introdução

Desde muito tempo existe a necessidade de algumas pessoas de se comunicarem sem que outras pessoas fiquem sabendo a respeito do que elas estão falando. Às vezes, essa comunicação tinha que ser feita por escrito. Disso surgiu a demanda por uma maneira de ocultar uma mensagem através de símbolos de forma que apenas o destinatário conseguisse ler. A esse processo deu-se o nome de **criptografia**.

Existem vários modos de se ocultar uma mensagem. Um bem simples é atribuir a cada letra do alfabeto um número e no momento de escrever a mensagem substituí-la pelo seu número correspondente. Esse método é ineficiente por vários motivos, um deles é a necessidade de uma maneira de enviar para o destinatário qual o código que se está usando, e caso ele pare em mãos erradas essa pessoa saberá como decodificar as mensagens. Outro problema de usar esse método é dado pela frequência que cada letra ocorre quando estamos escrevendo um texto. Nesse parágrafo, se você olhar, a letra **a** ocorre 124 vezes enquanto a **z** aparece 4 vezes, apenas as 4 vezes dessa frase. Existem estudos sobre a frequência com que cada letra aparece em cada língua utilizada e com eles é possível descobrir qual é a mensagem codificada. Esses métodos de descobrir qual é a mensagem através de um padrão observado são chamados de **ataques frequencistas**. Nesse exemplo, a chave para descobrir qual é a mensagem é a tabela com o valor de cada letra, que tem de ficar oculta. A esse tipo de criptografia no qual apenas o destinatário e o remetente podem saber os códigos de codificar denominamos **criptografia de chave privada**.

A fraqueza do método da substituição vem da repetição do uso da mesma tabela. Uma solução natural é trocar essa tabela após algumas utilizações. Uma única utilização da tabela torna seguro esse método. Mas, isso gera um problema na distribuição das chaves. Uma interessante solução para o problema da distribuição de chaves é a chamada **criptografia de chave**

pública. Nela todos podem saber como codificar uma mensagem, porém apenas o destinatário saberá como decodificá-la

Um dos métodos de chave pública mais conhecido e utilizado é a **criptografia RSA**. Este código foi inventado em 1978 por R. L. Rivest, A. Shamir e L. Adleman, que na época trabalhavam no Massachusetts Institute of Technology (MIT). As letras RSA correspondem às iniciais dos sobrenomes dos criadores. Este método é fantástico porque utiliza idéias básicas da matemática para fazer algo surpreendente. A segurança do método é devida ao fato de que não se conhece ainda um algoritmo rápido para fatorar números compostos grandes em um computador convencional.

O desenvolvimento da computação quântica, um modelo baseado na mecânica quântica, possibilitou um grande avanço na computação teórica. Ele possibilitou a descoberta de alguns algoritmos rápidos para resolver problemas para os quais até o presente momento acredita-se que não exista nenhum análogo clássico. Um desses algoritmos é o algoritmo de Shor, que data de 1994. Com ele e um computador quântico tem-se um algoritmo eficiente para fatoração de inteiros. Até o presente momento os computadores quânticos construídos realizam tarefas modestas, por exemplo, o algoritmo para fatoração implementado nele fatora números pequenos como o 15, veja [13]. Caso esses modelos se desenvolvam de modo a trabalharem com números tão grandes quanto os computadores clássicos atuais trabalham, a criptografia RSA se tornará completamente obsoleta, provocando um caos nos sistemas criptográficos.

O assunto desse trabalho é o algoritmo de Shor. Mas, antes de apresentá-lo será necessário que o leitor adquira conhecimentos sobre teoria dos números básica, algoritmos, mecânica e computação quântica. Isso será feito nos capítulos 1 e 2. Se o leitor já for familiarizado com um desses assuntos sugiro que pule o respectivo capítulo, pois acredito que se não o fizer achará o texto tedioso. Esse sentimento não pode existir, pois as idéias por trás do algoritmo são lindas e brilhantes.

Capítulo 1

Noções Básicas

Neste capítulo aprenderemos alguns resultados que serão úteis posteriormente. Eles serão introduzidos de forma bem básica para que mesmo quem nunca tenha tido contato com o assunto consiga acompanhar.

1.1 Aritmética Modular

Nos números inteiros já aprendemos a somar e a multiplicar, por exemplo $2 + 2 = 4$ ou $2 \cdot 3 = 6$. Vamos olhar para os números inteiros de uma forma diferente. Se pegarmos um número $a \in \mathbb{Z}$ e dividí-lo por 2 temos duas possibilidades para o resto, 0 se o número for par, e 1 se ele for ímpar. Vamos definir o conjunto \mathbb{Z}_2 para ser formado pelos restos que temos quando dividimos um número por 2, portanto $\mathbb{Z}_2 = \{\bar{0}, \bar{1}\}$, onde o símbolo - acima do 0 é para indicar que não é apenas o número 0, mas sim o conjunto de todos os pares. Logo $\bar{0}$ é o conjunto dos elementos que deixam resto 0. Analogamente, $\bar{1}$ é o conjunto dos números que deixam resto 1. Observe que em $\bar{0}$ os números $\{\dots, -4, -2, 0, 2, 4, \dots\}$ representam o mesmo elemento de \mathbb{Z}_2 . Como podemos ter vários representantes para um mesmo conjunto definiremos que usaremos sempre o menor natural. Agora que já conhecemos o \mathbb{Z}_2 , podemos definir o \mathbb{Z}_n . Este será formado por todos os restos possíveis na

divisão por n , $\mathbb{Z}_n = \{\bar{0}, \bar{1}, \bar{2}, \dots, \overline{n-1}\}$ ¹, onde o símbolo - em cima de cada elemento é para lembrar que esses elementos representam vários outros e não apenas exerce o papel que ele tem em \mathbb{Z} . Em \mathbb{Z}_n um número b pertence ao conjunto \bar{a} , isto é, $\bar{b} = \bar{a}$, se $b - a = nq$, para algum $q \in \mathbb{Z}$.

Nesse novo conjunto, o ideal seria se conseguíssemos fazer algumas operações aritméticas nele. Se somarmos dois números pares teremos como resultado um par, se forem dois ímpares o resultado será um par, se for um par com um ímpar o resultado será um ímpar. Escrevendo a frase acima na linguagem matemática que estamos introduzindo teremos:

$$\begin{aligned}\bar{0} + \bar{0} &= \bar{0}, \\ \bar{1} + \bar{1} &= \bar{0}, \\ \bar{1} + \bar{0} &= \bar{1}, \\ \bar{0} + \bar{1} &= \bar{1}.\end{aligned}$$

Em \mathbb{Z}_n definiremos que $\bar{a} + \bar{b} = \overline{a + b}$, onde $\overline{a + b}$ é o mesmo que somar a e b como elementos de \mathbb{Z} e depois dividir por n e calcular o resto. Para fazer isso sempre temos que mostrar que essa operação está bem definida, isto é, qualquer que sejam os representantes escolhidos para efetuar a soma de dois conjuntos, o resultado sempre será o mesmo conjunto. Observe: mesmo conjunto, mas não necessariamente mesmo representante. Isto é fácil de verificar. Digamos que em \mathbb{Z}_n temos dois conjuntos, representados por \bar{a} e \bar{b} , respectivamente. Digamos ainda que $\bar{a} = \bar{a'}$ e $\bar{b} = \bar{b'}$. Queremos verificar que $\overline{a + b} = \overline{a' + b'}$. Mas, $\bar{a} = \bar{a'}$ é equivalente a dizer que $a - a'$ é múltiplo de n ; e o mesmo vale para $b - b'$. Somando dois múltiplos de n temos um múltiplo de n , logo

$$(a - a') + (b - b') = (a + b) - (a' + b')$$

¹Se o leitor já tiver tido contato com aritmética modular antes deve lembrar que os elementos de \mathbb{Z}_n são classes de equivalência. Nesse trabalho optamos por evitar esse conceito. Nosso intuito era apenas que o leitor que nunca tinha visto esse assunto pudesse acompanhar os cálculos que serão feitos nos capítulos 4 e 5. Mas, se o leitor tiver curiosidade em ver essa outra abordagem basta consultar [2].

é múltiplo de n . Portanto, $\overline{a + b} = \overline{a' + b'}$, como queríamos mostrar.

Vamos passar à multiplicação. A forma natural para multiplicar os elementos \bar{a} e \bar{b} de \mathbb{Z}_n deveria ser,

$$\bar{a} \cdot \bar{b} = \overline{ab}.$$

Como no caso da soma, temos que verificar se esta fórmula dá um resultado que é independente da escolha de representantes para os conjuntos. Digamos que $\bar{a} = \overline{a'}$ e que $\bar{b} = \overline{b'}$. Queremos mostrar que $\overline{ab} = \overline{a'b'}$. Como $\bar{a} = \overline{a'}$, temos que $a - a'$ é um múltiplo de n , ou seja, $a = a' + rn$, para algum inteiro r . De maneira análoga, $b = b' + sn$, para algum inteiro s . Multiplicando,

$$ab = (a' + rn)(b' + sn) = a'b' + (a's + rb' + srn)n.$$

Logo $ab - a'b'$ é um múltiplo de n . Portanto, $\overline{ab} = \overline{a'b'}$, que é o que queríamos provar. Dessa forma, sabemos como somar e multiplicar elementos de \mathbb{Z}_n .

Dada uma operação $(*)$ em um conjunto A , se existe um elemento e tal que $e * b = b = b * e$ para todo $b \in A$, chamaremos e de elemento neutro. Um elemento $b \in A$ tem inverso se existe um elemento c tal que $c * b = e = b * c$.

Considerando o conjunto dos \mathbb{Z} inteiros com a operação soma sabemos que o 0 é o elemento neutro e que para todo elemento a existe um elemento inverso, $-a$. Vamos verificar que em \mathbb{Z}_n , $\bar{0}$ é o elemento neutro da soma. Dado $\bar{a} \in \mathbb{Z}_n$ temos $\bar{a} + \bar{0} = \bar{a} = \bar{0} + \bar{a}$. Além disso, para todo elemento \bar{a} existirá $\overline{-a}$, tal que $\bar{a} + \overline{-a} = \overline{a - a} = \bar{0} = \overline{-a} + \bar{a}$. Concluimos que qualquer elemento de \mathbb{Z}_n tem inverso com a operação soma. A diferença é definida da mesma forma que a soma: $\bar{a} - \bar{b} = \overline{a - b}$. A operação diferença em \mathbb{Z}_n entre dois elementos \bar{a} e \bar{b} pode ser vista como a soma entre os elementos \bar{a} e $\overline{-b}$.

Na multiplicação em \mathbb{Z} o elemento neutro é o 1 e os únicos elementos que têm inverso são o 1 e -1 . Em \mathbb{Z}_n o elemento $\bar{1}$ é o elemento neutro da multiplicação, pois $\bar{1} \cdot \bar{a} = \bar{a} = \bar{a} \cdot \bar{1}$, para todo $\bar{a} \in \mathbb{Z}_n$. Mas, com esta operação não é todo elemento de \mathbb{Z}_n que possui inverso. Vamos ver o que

acontece em \mathbb{Z}_4 , com o elemento $\bar{2}$,

$$\begin{aligned}\bar{2} \cdot \bar{0} &= \bar{0} \\ \bar{2} \cdot \bar{1} &= \bar{2} \\ \bar{2} \cdot \bar{2} &= \bar{4} = \bar{0} \\ \bar{2} \cdot \bar{3} &= \bar{6} = \bar{2},\end{aligned}$$

temos que este elemento não tem inverso multiplicativo. Agora iremos analisar o elemento $\bar{2} \in \mathbb{Z}_5$,

$$\begin{aligned}\bar{2} \cdot \bar{0} &= \bar{0} \\ \bar{2} \cdot \bar{1} &= \bar{2} \\ \bar{2} \cdot \bar{2} &= \bar{4} \\ \bar{2} \cdot \bar{3} &= \bar{6} = \bar{1} \\ \bar{2} \cdot \bar{4} &= \bar{8} = \bar{3},\end{aligned}$$

concluimos que este elemento possui inverso, o qual é o $\bar{3}$. Com isso podemos perceber que em \mathbb{Z}_n podem existir elementos além do $\bar{1}$ e $\overline{-1}$ que possuem inverso. Se um elemento a possui inverso dizemos que ele é **inversível**. O teorema abaixo nos dá um critério para determinar se um elemento de \mathbb{Z}_n é inversível.

Teorema 1 [1] *Um elemento \bar{a} em \mathbb{Z}_n é inversível se, e somente se,*

$$\text{mdc}(a, n) = 1.$$

Os elementos inversíveis de \mathbb{Z}_n formam o conjunto definido por,

$$\mathbb{Z}_n^* = \{\bar{a} \in \mathbb{Z}_n \text{ tal que } \text{mdc}(n, a) = 1\}.$$

Definição 1 *A função $\varphi(n)$ conhecida como função φ de Euler é dada por:*

$$\begin{aligned}\varphi : \mathbb{N} &\rightarrow \mathbb{N} \\ n &\mapsto \#\mathbb{Z}_n^*,\end{aligned}$$

onde o símbolo $\#A$ simboliza a quantidade de elementos do conjunto A .

Propriedades de φ :

- 1) Se p é primo, $\varphi(p) = p - 1$.
- 2) $\varphi(p^\alpha) = p^{\alpha-1} \cdot (p - 1)$, para p primo.
- 3) Se m e n são inteiros positivos tais que $\text{mdc}(m, n) = 1$, então, $\varphi(n \cdot m) = \varphi(n) \cdot \varphi(m)$.

Para demonstrações veja [1].

Por último, apresentaremos a notação de congruência modular no conjunto \mathbb{Z}_n . Dizemos que $a \equiv b \pmod n$ se $\bar{a} = \bar{b}$, caso contrário, denotaremos $a \not\equiv b \pmod n$. Suponha que $a \equiv b \pmod n$ e $c \equiv d \pmod n$, se somarmos essas duas expressões obtemos: $a + c \equiv b + d \pmod n$, se tivéssemos multiplicado teríamos $a \cdot c \equiv b \cdot d \pmod n$. Introduzimos essa notação, pois ela é muito útil para realizarmos operações aritméticas em \mathbb{Z}_n .

Teorema 2 Teorema de Euler[1] - Se $\text{mdc}(x, n) = 1$ então, $x^{\varphi(n)} \equiv 1 \pmod n$.

1.2 Grupos

Um conjunto G com uma operação $(*)$ é um **grupo**, se $(*)$ possui as seguintes propriedades:

- 1) Fechado: dados a e $b \in G$ se $a * b = c$ então $c \in G$.
- 2) Elemento neutro: existe um elemento $e \in G$ tal que para todo $a \in G$ temos $a * e = e * a = a$.
- 3) Associatividade: dados $a, b, c \in G$ temos que $a * (b * c) = (a * b) * c$.
- 4) Elemento inverso: dado um elemento $a \in G$ qualquer, existe um elemento $a' \in G$ tal que $a * a' = a' * a = e$.

Um grupo pode possuir infinitos elementos, caso a quantidade de elementos seja finita, dizemos que G é um grupo finito. No último caso chamamos a quantidade de elementos de G de sua **ordem**. Seja G um grupo com a operação $(*)$, dado $a \in G$ denotamos $\underbrace{a * a * a \dots * a}_{k \text{ vezes}} = a^k$. O menor inteiro k tal que $a^k = e$ é a **ordem** de a .

Teorema 3 O conjunto \mathbb{Z}_n^* é um grupo com a operação produto.

DEMONSTRAÇÃO: Para \mathbb{Z}_n^* ser grupo verificaremos as propriedades acima:

1) (Fechado) Dados $\bar{a}, \bar{b} \in \mathbb{Z}_n^*$, sabemos que o $\text{mdc}(a, n) = 1$ e $\text{mdc}(b, n) = 1$, logo o $\text{mdc}(a \cdot b, n) = 1$. Portanto, $\bar{a} \cdot \bar{b} \in \mathbb{Z}_n^*$.

2) (Elemento neutro) O elemento $\bar{1} \in \mathbb{Z}_n^*$ e dado qualquer elemento $\bar{a} \in \mathbb{Z}_n^*$ temos que $\bar{a} \cdot \bar{1} = \bar{1} \cdot \bar{a} = \bar{a}$. Logo, $\bar{1}$ é o elemento neutro.

3) (Associatividade) Dados $\bar{a}, \bar{b}, \bar{c} \in \mathbb{Z}_n^*$, temos que $(\bar{a} \cdot \bar{b}) \cdot \bar{c} = \overline{(a \cdot b) \cdot c} = \overline{a \cdot (b \cdot c)} = \bar{a} \cdot (\bar{b} \cdot \bar{c})$.

4) (Elemento inverso) Dado $\bar{a} \in \mathbb{Z}_n^*$ sabemos que $\text{mdc}(a, n) = 1$ e pelo teorema 1 sabemos que \bar{a} é inversível. Portanto, existe \bar{a}' tal que $\bar{a} \cdot \bar{a}' = \bar{1} = \bar{a}' \cdot \bar{a}$. ■

Definição 2 *Seja G um grupo com a operação $(*)$, falamos que G é um grupo cíclico se existe $a \in G$ tal que $\forall g \in G$ $g = a^n$ para algum $n \in \mathbb{N}$. Neste caso, dizemos que a é um gerador para G .*

Teorema 4 *A ordem de \mathbb{Z}_N^* é $\varphi(N)$.*

DEMONSTRAÇÃO: Segue direto da definição de $\varphi(N)$. ■

1.3 Algoritmo

A etimologia da palavra algoritmo vem da forma latinizada do árabe Al-Khowarazmi, o 'homem de Khowarazm'. Essa é a maneira como era conhecido o matemático árabe Ben Musa que viveu no século IX. Foi através do seu livro 'Al-jabr wa'l muqabalah' que os algarismos indo-arábicos chegaram ao ocidente. Mas, atualmente o significado da palavra algoritmo é outro. Um algoritmo é simplesmente um método prático para resolver um problema. Como uma receita de bolo que se você seguir passo a passo conseguirá fazer o bolo. Quando formos descrevê-lo temos que deixar claro qual é a sua entrada e qual é a sua saída. A entrada é como se fosse os ingredientes na receita e a saída é como se fosse o bolo pronto. Além disso, temos o modo de fazer. É neste momento que iremos dizer como os ingredientes deverão ser misturados para termos o produto final.

Se tivermos um candidato a algoritmo temos de verificar duas coisas: se ele implementa a tarefa desejada e se faz isso em um tempo finito.

Antes de prosseguirmos com a discussão sobre os algoritmos, apresentaremos um teorema que justifica podermos sempre dividir dois números inteiros positivos.

Teorema 5 Teorema da divisão[1] *Sejam a e b números inteiros positivos. Existem números inteiros q e r tais que*

$$a = b \cdot q + r \text{ e } 0 \leq r < b.$$

Além disso, os valores de q e r satisfazendo as relações acima são únicos.

Iremos apresentar o algoritmo de Euclides que calcula o máximo divisor de dois números inteiros positivos:

Entrada: números inteiros positivos distintos a e b

Saída: $\text{mdc}(a, b)$

Etapa 1: Comece fazendo $c_1 = a$, $d_1 = b$ e $i = 1$

Etapa 2: utilize o algoritmo da divisão para dividir c_i por d_i e retornar r_i .

Etapa 3: Se $r_i = 0$ retorne d_i , caso contrário faça $c_{i+1} = d_i$, $d_{i+1} = r_i$, acrescente 1 ao contador i e volte a etapa 2.

Vamos fazer a leitura do algoritmo. Ele usa as variáveis i , c_i , d_i e r_i . Ao final da execução retornará o $\text{mdc}(a, b)$. Para encontrar esse valor ele terá que fazer a Etapa 2 e 3 várias vezes, chamamos isso de um laço. Na Etapa 1 atribuímos a c_1 e d_1 os valores de a e b , respectivamente e a i o valor 1. Na Etapa 2 usamos o algoritmo da divisão para encontrar r_1 , isto é, escrever $a = bq_1 + r_1$. Se na Etapa 2 o valor encontrado para r_1 for 0, na Etapa 3 o algoritmo retornará o $\text{mdc}(a, b)$, se não for, ele atribuirá valores para c_2 e d_2 , acrescentará 1 ao contador i e retornará a Etapa 2. Esse procedimento será repetido até encontrarmos o $\text{mdc}(a, b)$.

Temos de verificar se o algoritmo acima para e realmente realiza a tarefa a qual se propõe.

Verificando se termina em um número finito de passos:

Sabemos que a sequência de restos $r_1, r_2, \dots, r_n, \dots$, obedece às seguintes desigualdades $b > r_1 > r_2 > \dots > r_n > \dots \geq 0$. Com isso, verificamos que existem no máximo b restos possíveis. Como b é um número fixo, concluímos que o algoritmo para.

Verificando se a saída é o $\text{mdc}(a, b)$:

Lema 1 [1] *Sejam a e b números inteiros positivos. Suponhamos que existam inteiros g e s tais que $a = b \cdot g + s$. Então, $\text{mdc}(a, b) = \text{mdc}(b, s)$.*

Usaremos o lema para mostrar que o último resto não-nulo na sequência de divisões é o $\text{mdc}(a, b)$. Suponha que o algoritmo para depois de n laços. Temos:

$$\begin{aligned} a &= b \cdot q_1 + r_1 \\ b &= r_1 \cdot q_2 + r_2 \\ &\vdots \\ r_{n-3} &= r_{n-2} \cdot q_{n-1} + r_{n-1} \\ r_{n-2} &= r_{n-1} \cdot q_n + r_n \text{ e } r_n = 0 \end{aligned}$$

Da última linha temos que r_{n-1} divide r_{n-2} . Logo, $\text{mdc}(r_{n-1}, r_{n-2}) = r_{n-1}$. Agora aplicando o lema à penúltima linha temos

$$\text{mdc}(r_{n-3}, r_{n-2}) = \text{mdc}(r_{n-1}, r_{n-2}) = r_{n-1}.$$

Aplicando o lema às linhas anteriores até chegarmos à primeira temos que $\text{mdc}(a, b) = r_{n-1}$. Desse modo, o algoritmo retorna o $\text{mdc}(a, b)$.

Poderíamos nos perguntar se apenas as informações: o algoritmo realiza a tarefa e o faz em tempo finito não seriam suficientes para ficarmos satisfeitos com ele e implementá-lo em um computador. Não, pois este tempo finito poderia ser dez anos. Imagine se o algoritmo de Euclides levasse dez anos para calcular $\text{mdc}(272828281, 3242)$, na prática ele não seria interessante. Veremos mais adiante que isso não ocorre.

Estamos acostumados com a representação dos números naturais na base 10. Por exemplo, $523 = 5 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$. Porém, podemos representá-lo em outras bases. Como a base 2 (ou binária). Nesta base diremos que um número tem um comprimento de L bits se ele assim for escrito: $a_{L-1} \cdot 2^{L-1} + a_{L-2} \cdot 2^{L-2} \dots + a_0 \cdot 2^0$ que é representado por $a_{L-1}a_{L-2} \dots a_0$. Note que a_i assume apenas os valores 0 ou 1.

Quando analisamos um algoritmo é importante verificar quantas vezes cada parte está sendo executada. Às vezes, estuda-se a quantidade de memória necessária. Neste trabalho nosso foco será apenas no tempo de execução, deixaremos de lado qualquer outro fator que influencie no desempenho do algoritmo. Ao fazermos o estudo do tempo, utilizamos uma função de custo f que dependerá apenas do tamanho da entrada do algoritmo. Nesta seção, a entrada será um número inteiro positivo e o seu tamanho será L , a quantidade de bits.

No momento de fazer a análise temos como analisar sob três pontos de vista diferentes: **melhor caso**, **caso médio** e **pior caso**. O melhor caso corresponde ao menor tempo de execução sobre todas as entradas de tamanho n . O pior corresponde ao maior tempo de execução sobre todas as entradas de tamanho n . O caso médio corresponde à média do tempo de execução sobre todas as entradas de tamanho n . Nesta seção, por segurança nossas análises serão feitas sempre levando em consideração o pior caso.

Para valores pequenos da entrada n não precisamos nos preocupar com a eficiência do algoritmo, pois todos têm custo pequeno, mesmo os ineficientes. O problema é quando n começa a ficar muito grande. Para resolver isso estudamos o que acontece com o algoritmo para valores grandes de n , isto é, estuda-se o comportamento assintótico da função custo.

Definição 3 *Uma função $f(n)$ positiva domina assintoticamente outra função $g(n)$ positiva se existem duas constantes positivas c e m tais que se $m \leq n$ temos que $g(n) \leq c \cdot f(n)$. Neste caso, denotamos que $g(n)$ é $O(f(n))$, que se lê $g(n)$ é de ordem $O(f(n))$.*

Propriedades:

1. $f(n)$ é $O(f(n))$.
2. $O(c \cdot f(n))$ é $O(f(n))$ se c é uma constante positiva.
3. Se f domina assintoticamente g então $f(n) + g(n)$ é de ordem $O(f(n))$.
4. Se F domina assintoticamente f , e G domina assintoticamente g , $f(n) \cdot g(n)$ é de ordem $O(F(n) \cdot G(n))$.

Se f é a função custo para um algoritmo A , então $O(f)$ é considerada a **complexidade assintótica** ou o **comportamento assintótico** do algoritmo A .

Diz-se que um algoritmo no qual a função f de custo é dominada por um polinômio em n tem **complexidade polinomial**, caso contrário, diz-se que a sua complexidade é **exponencial**. Observe que há um abuso de linguagem na definição de complexidade exponencial, pois a função $n^{\log n}$ não é limitada por nenhum polinômio, logo seu crescimento é considerado exponencial. Mas, na matemática essa função não representa uma exponencial.

Definição 4 *Um problema é considerado **fácil**, **tratável** ou **solúvel** computacionalmente se existir um algoritmo com complexidade polinomial que o resolva. Um problema é considerado **difícil**, **intratável** ou **não-solúvel** computacionalmente se o melhor algoritmo existente tiver complexidade exponencial.*

Se um problema é solúvel é porque existe um algoritmo com complexidade polinomial que o resolve. Falamos que esse algoritmo é eficiente.

Determinar com exatidão o valor de $f(n)$ é uma tarefa difícil, mas determinar a sua complexidade assintótica é mais fácil. A partir de agora estaremos sempre preocupados com a complexidade assintótica da função, não nos preocuparemos com o seu valor exato.

Infelizmente não existe um conjunto completo de regras para analisar algoritmos. Mas, existem alguns princípios que norteiam essa tarefa. São eles:

1- se o tempo de execução do comando não varia com o tamanho da entrada ele é $O(1)$, por exemplo: comando de leitura, escrita e atribuição;

2- o tempo de execução de uma sequência de comandos é o do comando de maior tempo de execução;

3- o tempo para executar um laço é o tempo de executar o seu corpo vezes quantas iterações são feitas;

Agora que sabemos que para um algoritmo ser implementado ele tem de ser eficiente, analisaremos o algoritmo de Euclides sob esse ponto de vista.

Suponha que a e b estão escritos em binários e o maior comprimento deles é L , isto é, o número possui L -bits. A Etapa 1 tem um custo $O(1)$, pois é um comando de atribuição. O custo da Etapa 2 é $O(L^2)$, pois ela implementa o algoritmo da divisão que tem esse custo computacional. A Etapa 3 é constituída de um comando condicional que é $O(1)$ e um de atribuição que é $O(1)$, logo pelo princípio 2 o custo dessa etapa é $O(1)$. O custo de cada iteração do laço é $\max\{O(L^2), O(1)\} = O(L^2)$. Para determinarmos o custo total do algoritmo basta sabermos qual a maior quantidade de vezes que o laço será implementado. Temos que os restos r_i da sequência de divisão do algoritmo têm tamanho menor ou igual que L -bits. Vamos verificar que $\frac{r_i}{2} \geq r_{i+2}$, temos dois casos a considerar:

1) $\frac{r_i}{2} \geq r_{i+1}$, neste caso é claro, pois $r_{i+1} \geq r_{i+2}$.

2) $r_{i+1} > \frac{r_i}{2}$, neste caso, temos que $r_i = q_{i+1} \cdot r_{i+1} + r_{i+2}$ com $q_{i+1} > 0$, logo $r_{i+2} \leq r_i - r_{i+1} \leq \frac{r_i}{2}$.

O pior caso que poderá acontecer é a , b e r_1 terem L -bits e $r_{i+2} = \frac{r_i}{2}$. Se isso ocorrer, teremos que o laço será iterado $2 \cdot L$ vezes que é $O(L)$, portanto o custo da Etapa 2 e 3 é $O(L^3)$, e o custo total do algoritmo é $\max\{O(L^3), O(1)\} = O(L^3)$. Concluimos que o algoritmo de Euclides é eficiente, pois ele é $O(L^3)$.

Quando analisamos se o algoritmo parava, a justificativa foi que como os restos sempre diminuem de uma etapa para a outra e na primeira etapa r_1 é menor que b concluimos que ele irá parar. Com esse argumento não é possível concluir que o algoritmo de Euclides é eficiente. Já que o pior caso

é o resto diminuir de uma unidade em cada etapa. Desse modo, o pior caso é o laço ser iterado b vezes, e b ter L -bits. Com isso teríamos uma função custo $O(L^2 2^L)$, portanto de complexidade exponencial.

Apresentaremos um algoritmo que determina se um dado número natural N pode ser escrito como a^b , onde a e $b \in \mathbb{N}$. Além disso, $a \geq 1$ e $b \geq 2$. Se este fato ocorre $b \leq L$, onde L é a quantidade de bits de N . Para ver isto temos dois casos: o primeiro é $N = 1$ que é trivial; o segundo é $N \neq 1$, logo $a \geq 2$. Portanto,

$$\begin{aligned} N &= a^b \\ \log_2 N &= b \log_2 a \\ L &> b \log_2 a, \end{aligned}$$

onde $L = \lfloor \log_2 N + 1 \rfloor$. Mas, $\log_2 a \geq 1$. Desse modo, $b \leq L$.

Algoritmo

Entrada: N ;

Etapa 1: faça $i = 2$;

Etapa 2: faça $x = \frac{\log_2 N}{i}$. Calcule os dois inteiros u_1 e u_2 mais próximos de 2^x .

Etapa 3: faça u_1^i e u_2^i e verifique se um dos resultados é N . Se for retorne os valores de i e u_j tais que $u_j^i = N$, para $j = 1, 2$. Senão, faça $i = i + 1$ e retorne a Etapa 2.

Para calcular $x = \frac{\log_2 N}{i}$, primeiro é necessário calcular $\log_2 N$ e depois aplicar o algoritmo da divisão. Na determinação desse custo precisamos saber qual complexidade é maior: do algoritmo da divisão ou do $\log_2 N$? O algoritmo da divisão sabemos que é $O(L^2)$. Uma maneira de calcular $\log_2 N$ é utilizar a igualdade abaixo e a série de Taylor da função \log na vizinhança do 1,

$$\begin{aligned} \log_2 N &= L - (L - \log_2 N) \\ &= L - \log_2 \frac{2^L}{N} = L - \frac{1}{\ln 2} \left(\sum_{j=1}^{\infty} \frac{(-1)^{j-1} \left(\frac{2^L}{N} - 1\right)^j}{j} \right). \end{aligned}$$

Como esse cálculo será feito em um computador não utilizaremos a série toda, apenas os n primeiros termos, note que n é fixo. Então, para obtermos $\log_2 N$ precisamos calcular $(\frac{2^L}{N} - 1)^j$ e sua respectiva divisão por j . Mas, esse cálculo em cada etapa será repetido no máximo n vezes, como n é fixo temos que esse custo é $O(L^2)$. Logo a etapa 2 é $O(L^2)$.

Na etapa 3 para fazer u_1^i ou u_2^i é $O(L^3)$. A multiplicação é $O(L^2)$ e $i \leq b$, logo serão feitas no máximo L multiplicações. Como as etapas 2 e 3 podem ser repetidas no máximo $L - 2$ vezes o custo do algoritmo é $O(L^4)$.

1.4 Frações Contínuas

O intuito das frações contínuas é expressar os números racionais por expressões do tipo:

$$a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{\dots + \frac{1}{a_m}}}} \quad (1.1)$$

em que a_1, \dots, a_m são números inteiros. Mas, nesse texto assumiremos que a_1, \dots, a_m são números naturais. Às vezes, representaremos a expressão (1.1) como $[a_1, a_2, \dots, a_{m-1}, a_m]$.

Veremos como o algoritmo de Euclides apresentado na seção 1.3 ajudará a encontrar a expressão (1.1) para um dado número racional $\frac{a}{b}$. Suponha que nosso objetivo seja encontrar o $\text{mdc}(69, 15)$. Pelo algoritmo de Euclides, temos:

$$\begin{aligned} 69 &= 4 \cdot 15 + 9 \\ 15 &= 1 \cdot 9 + 6 \\ 9 &= 1 \cdot 6 + 3 \\ 6 &= 2 \cdot 3 + 0. \end{aligned}$$

Logo, $\text{mdc}(69, 15) = 3$, visto que 3 é o último resto não-nulo nesta sequência de divisões. Uma consequência destas igualdades é podermos expressar $\frac{69}{15}$

como:

$$\begin{aligned}
\frac{69}{15} &= 4 + \frac{9}{15} = 4 + \frac{1}{\frac{15}{9}} \\
&= 4 + \frac{1}{1 + \frac{6}{9}} = 4 + \frac{1}{1 + \frac{1}{\frac{9}{6}}} \\
&= 4 + \frac{1}{1 + \frac{1}{1 + \frac{3}{6}}} = 4 + \frac{1}{1 + \frac{1}{1 + \frac{1}{\frac{6}{3}}}} \\
&= 4 + \frac{1}{1 + \frac{1}{1 + \frac{1}{2}}}. \tag{1.2}
\end{aligned}$$

A expressão (1.2) recebe o nome de fração contínua do número $\frac{69}{15}$ e também pode ser representada por $[4, 1, 1, 2]$. Note que esses números são os quocientes da sequência de divisões do algoritmo de Euclides.

Para obtermos a expressão (1.1) basta determinarmos os valores dos $a_{i's}$. Pelo exemplo acima vimos que esses números são os quocientes que aparecem no algoritmo de Euclides. Logo, para encontrá-los, basta fazermos uma pequena modificação no algoritmo, mandar ele guardar os valores dos $q_{i's}$. Essa operação é $O(1)$. Portanto, essa tarefa adicional não aumenta a complexidade do algoritmo. Assim, temos um algoritmo $O(L^3)$ para encontrar a fração contínua de um número racional, onde L é a quantidade de bits do $\max\{a, b\}$.

Teorema 6 *Todo número racional possui uma única representação sob a forma de fração contínua.*

DEMONSTRAÇÃO: Este fato é facilmente visto, se lembrarmos que o algoritmo de Euclides garante que cada linha do cálculo do máximo divisor comum existe e é escrita de modo único. ■

Se p e q pertencem aos naturais, sabemos que $\frac{p}{q} = [a_1, a_2, \dots, a_m]$. Consideremos as seguintes frações:

$$c_1 = \frac{a_1}{1}, c_2 = a_1 + \frac{1}{a_2}, c_3 = a_1 + \frac{1}{a_2 + \frac{1}{a_3}}, \dots, c_m = a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{\dots + \frac{1}{a_m}}}}$$

obtidas pelas expansões das seguintes frações contínuas:

$$[a_1], [a_1, a_2], [a_1, a_2, a_3], \dots, [a_1, \dots, a_m].$$

Estas frações são chamadas de primeiro, segundo, terceiro, ..., m -ésimo convergentes, respectivamente, da fração contínua $[a_1, a_2, \dots, a_m]$. Note que o m -ésimo convergente é igual à própria fração contínua.

Teorema 7 *Seja $c_i = \frac{p_i}{q_i}$ o i -ésimo convergente da fração contínua $[a_1, a_2, \dots, a_m]$. Então o numerador p_i e o denominador q_i de c_i satisfazem as seguintes relações:*

$$p_i = a_i p_{i-1} + p_{i-2} \quad (1.3)$$

$$q_i = a_i q_{i-1} + q_{i-2}, \quad (1.4)$$

para $i = 3, 4, \dots, m$; onde:

$$p_1 = a_1, p_2 = a_2 a_1 + 1, q_1 = 1, q_2 = a_2.$$

DEMONSTRAÇÃO: Esse resultado será provado por indução. Para isso, demonstraremos que essas relações são satisfeitas para $i = 3$.

$$\begin{aligned} c_3 &= a_1 + \frac{1}{a_2 + \frac{1}{a_3}} \\ &= a_1 + \frac{1}{\frac{a_2 a_3 + 1}{a_3}} \\ &= a_1 + \frac{a_3}{a_2 a_3 + 1} \\ &= \frac{a_1 a_2 a_3 + a_1 + a_3}{a_2 a_3 + 1} \\ &= \frac{a_3(a_1 a_2 + 1) + a_1}{a_3 a_2 + 1}. \end{aligned}$$

Por hipótese temos: $p_1 = a_1, p_2 = a_2 a_1 + 1, q_1 = 1, q_2 = a_2$. Logo, a equação acima torna-se:

$$c_3 = \frac{a_3 p_2 + p_1}{a_3 q_2 + q_1} = \frac{p_3}{q_3}.$$

Assumiremos que as equações (1.3) e (1.4) são válidas para todo $j \leq i$. Isto é,

$$c_i = [a_1, \dots, a_i] = \frac{p_i}{q_i} = \frac{a_i p_{i-1} + p_{i-2}}{a_i q_{i-1} + q_{i-2}}, \quad (1.5)$$

e mostraremos que isto implica a validade do teorema para $i + 1$. Tomemos o seguinte convergente $c_i = [a_1, \dots, a_i]$, escrevendo a sua expressão em frações contínuas,

$$c_i = a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots + \frac{1}{a_{i-1} + \frac{1}{a_i}}}}, \quad (1.6)$$

o próximo convergente é dado por:

$$c_{i+1} = a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \dots + \frac{1}{a_{i-1} + \frac{1}{a_i + \frac{1}{a_{i+1}}}}}}. \quad (1.7)$$

Das equações (1.6) e (1.7), percebemos que c_{i+1} é obtido da expressão de c_i , pela substituição de a_i por $a_i + \frac{1}{a_{i+1}}$.

Como estamos supondo por indução que (1.3) e (1.4) são válidas para todo $j \leq i$. Logo,

$$\frac{p_{i-1}}{q_{i-1}} = \frac{a_{i-1}p_{i-2} + p_{i-3}}{a_{i-1}q_{i-2} + q_{i-3}},$$

os números p_{i-1} e q_{i-1} dependem apenas dos números a_{i-1} e dos números $p_{i-2}, q_{i-2}, p_{i-3}, q_{i-3}$ os quais dependem dos precedentes a 's, p 's e q 's. Deste modo $p_{i-2}, q_{i-2}, p_{i-1}$ e q_{i-1} são independentes de a_i . Logo, quando substituirmos a_i por $a_i + \frac{1}{a_{i+1}}$, os números p_i, p_{i-1}, q_i e q_{i-1} não serão afetados. Podemos utilizar (1.5) para calcular c_{i+1} , bastando fazer a seguinte troca, a_i por $a_i + \frac{1}{a_{i+1}}$

$$\begin{aligned} c_{i+1} &= \frac{\left(a_i + \frac{1}{a_{i+1}}\right) p_{i-1} + p_{i-2}}{\left(a_i + \frac{1}{a_{i+1}}\right) q_{i-1} + q_{i-2}} \\ &= \frac{(a_{i+1}a_i + 1)p_{i-1} + a_{i+1}p_{i-2}}{(a_{i+1}a_i + 1)q_{i-1} + a_{i+1}q_{i-2}} \\ &= \frac{a_{i+1}(a_i p_{i-1} + p_{i-2}) + p_{i-1}}{a_{i+1}(a_i q_{i-1} + q_{i-2}) + q_{i-1}} \\ &= \frac{a_{i+1}p_i + p_{i-1}}{a_{i+1}q_i + q_{i-1}} \\ &= \frac{p_{i+1}}{q_{i+1}}. \end{aligned}$$

Portanto, nossa demonstração por indução está concluída. ■

Definiremos $p_0 = 1, p_{-1} = 0, q_0 = 0, q_{-1} = 1$. Com isso, as equações

$$p_i = a_i p_{i-1} + p_{i-2} \quad (1.8)$$

$$q_i = a_i q_{i-1} + q_{i-2} \quad (1.9)$$

tornam-se verdadeiras para $i \geq 1$

Teorema 8 *A relação*

$$p_i q_{i-1} - p_{i-1} q_i = (-1)^i, \quad (1.10)$$

se verifica para todo $i \geq 0$, onde p_i e q_i são, respectivamente, o numerador e o denominador do i -ésimo convergente.

DEMONSTRAÇÃO: Provaremos esse resultado usando o princípio de indução.

Primeiro verificaremos para o caso $i = 0$

$$p_0 q_{-1} - p_{-1} q_0 = 1 = (-1)^0.$$

Assumiremos que esse resultado é válido para i , isto é, $p_i q_{i-1} - p_{i-1} q_i = (-1)^i$ e vamos mostrar que isso implica o resultado para $i + 1$. Do teorema 7 temos que: $p_{i+1} = a_{i+1} p_i + p_{i-1}$ e $q_{i+1} = a_{i+1} q_i + q_{i-1}$. Com isso,

$$\begin{aligned} p_{i+1} q_i - p_i q_{i+1} &= (a_{i+1} p_i + p_{i-1}) q_i - p_i (a_{i+1} q_i + q_{i-1}) \\ &= a_{i+1} p_i q_i + p_{i-1} q_i - a_{i+1} p_i q_i - p_i q_{i-1} \\ &= (-1)(p_i q_{i-1} - p_{i-1} q_i), \end{aligned}$$

pela hipótese de indução, temos:

$$p_{i+1} q_i - p_i q_{i+1} = (-1)(-1)^i = (-1)^{i+1}.$$

O que prova o resultado. ■

Corolário 1 *Para todo convergente $c_i = \frac{p_i}{q_i}$ temos que $\text{mdc}(p_i, q_i) = 1$.*

DEMONSTRAÇÃO: Pelo teorema 8 temos que $p_i q_{i-1} - p_{i-1} q_i = (-1)^i$. Se existir um d tal que d seja um divisor de p_i e q_i , temos que d também é um divisor de 1 ou -1 . Portanto, o $\text{mdc}(p_i, q_i) = 1$. ■

Sabemos que qualquer número racional α pode ser escrito na forma:

$$\alpha = [a_1, a_2, \dots, a_{i-1}, a_i] = \frac{a_i p_{i-1} + p_{i-2}}{a_i q_{i-1} + q_{i-2}}.$$

Fazendo a diferença

$$\begin{aligned} \alpha - \frac{p_{i-1}}{q_{i-1}} &= \frac{\alpha q_{i-1} - p_{i-1}}{q_{i-1}} = \frac{a_i p_{i-1} + p_{i-2}}{a_i q_{i-1} + q_{i-2}} - \frac{p_{i-1}}{q_{i-1}} \\ &= \frac{-(p_{i-1} q_{i-2} - p_{i-2} q_{i-1})}{q_{i-1}(a_i q_{i-1} + q_{i-2})} \end{aligned}$$

e aplicando o teorema 8, obtemos a expressão:

$$\frac{\alpha q_{i-1} - p_{i-1}}{q_{i-1}} = \frac{(-1)^i}{q_{i-1}(a_i q_{i-1} + q_{i-2})}. \quad (1.11)$$

Pela equação (1.11), observamos que $\alpha q_i - p_i$ e $\alpha q_{i+1} - p_{i+1}$ possuem sinais opostos. Este fato será utilizado na demonstração do lema seguinte.

Lema 2 *Seja α um número racional e $\frac{p_n}{q_n}$ os convergentes da expansão de α em fração contínua. Se $\frac{a}{b}$ for um racional com $b > 0$ tal que:*

$$|\alpha b - a| < |\alpha q_n - p_n|,$$

para algum $n \geq 0$, então $b \geq q_{n+1}$.

DEMONSTRAÇÃO: A prova do teorema será feita por contradição. Suponha $|\alpha b - a| < |\alpha q_n - p_n|$ e $b < q_{n+1}$.

Analisaremos o seguinte sistema linear em x e y :

$$\begin{cases} p_n x + p_{n+1} y = a \\ q_n x + q_{n+1} y = b. \end{cases}$$

Pelo teorema 8 temos que o determinante deste sistema é 1 ou -1 , logo esse sistema possui uma única solução. Além disso, essa solução pertence aos números inteiros.

Mostraremos que x e y são diferentes de zero. Pois, se $x = 0$, então $b = yq_{n+1}$, portanto, $y > 0$ e $b \geq q_{n+1}$, que nos dá uma contradição com $b < q_{n+1}$. Se $y = 0$, então $a = xp_n$, $b = xq_n$ e

$$|\alpha b - a| = |\alpha xq_n - xp_n| = |x||\alpha q_n - p_n|.$$

Como $x \in \mathbb{Z}$, temos que $|x| \geq 1$. Portanto, $|\alpha b - a| \geq |\alpha q_n - p_n|$, o que novamente nos leva a uma contradição.

Nosso próximo objetivo é mostrar que x e y possuem sinais opostos. Se $y < 0$, sabemos que $xq_n = b - yq_{n+1}$, como $b > 0$, temos $x > 0$. Se $y > 0$, então $b < yq_{n+1}$, pois $b < q_{n+1}$. Portanto, xq_n é negativo. Mostrando que $x < 0$.

De (1.11) sabemos que $\alpha q_n - p_n$ e $\alpha q_{n+1} - p_{n+1}$ possuem sinais opostos e, logo, $x(\alpha q_n - p_n)$ e $y(\alpha q_{n+1} - p_{n+1})$ possuem o mesmo sinal.

Do sistema acima obtemos $\alpha b - a = x(\alpha q_n - p_n) + y(\alpha q_{n+1} - p_{n+1})$. Como os dois termos do lado direito da equação possuem o mesmo sinal, temos:

$$\begin{aligned} |\alpha b - a| &= |x(\alpha q_n - p_n) + y(\alpha q_{n+1} - p_{n+1})| \\ &= |x(\alpha q_n - p_n)| + |y(\alpha q_{n+1} - p_{n+1})| \\ &> |x(\alpha q_n - p_n)| \geq |\alpha q_n - p_n|, \end{aligned}$$

as equações acima nos dão uma contradição, o que prova o teorema. ■

Teorema 9 *Seja α um número racional. Se existir um racional $\frac{a}{b}$, com $b \geq 1$ tal que*

$$\left| \alpha - \frac{a}{b} \right| < \frac{1}{2b^2},$$

então $\frac{a}{b}$ é um dos convergentes da expansão de α em fração contínua.

DEMONSTRAÇÃO: Podemos supor $\text{mdc}(a, b) = 1$ sem perda de generalidade. Provaremos esse resultado por contradição. Suponhamos que exista um racional $\frac{a}{b}$ que satisfaz as hipóteses do teorema e que $\frac{a}{b}$ não seja um convergente da fração contínua de α . Seja n o inteiro tal que $q_n \leq b \leq q_{n+1}$. Para este inteiro a desigualdade $|\alpha b - a| < |\alpha q_n - p_n|$ é impossível pelo lema 2. Logo,

$$|\alpha q_n - p_n| \leq |\alpha b - a| < \frac{1}{2b}$$

$$\left| \alpha - \frac{p_n}{q_n} \right| < \frac{1}{2bq_n}.$$

Utilizando o fato de que $\frac{a}{b} \neq \frac{p_n}{q_n}$ e que $bp_n - aq_n$ é um número inteiro não-nulo, obtemos:

$$\frac{1}{bq_n} \leq \frac{|bp_n - aq_n|}{bq_n} = \left| \frac{p_n}{q_n} - \frac{a}{b} \right|$$

$$\leq \left| \alpha - \frac{a}{b} \right| + \left| \alpha - \frac{p_n}{q_n} \right| < \frac{1}{2b^2} + \frac{1}{2bq_n}.$$

Logo,

$$\frac{1}{2bq_n} < \frac{1}{2b^2}.$$

Com isto concluímos que $b < q_n$, o que é uma contradição. ■

1.5 Álgebra linear

Esta seção será dedicada a apresentar alguns fatos de álgebra linear em uma notação diferente da que matemáticos estão acostumados. Essa nova maneira de ver a álgebra linear facilita muito os cálculos na mecânica quântica. Neste texto, por convenção, um espaço vetorial será sempre complexo, de dimensão finita e com produto interno. Às vezes, o denotaremos por H_n , onde n é a dimensão do espaço vetorial. Assumiremos que H_n será sempre dado com uma base ortonormal.

Um vetor em um espaço complexo será descrito como $|\psi\rangle$, (lê-se ket psi). O elemento 0 será usado para denotar a origem, pois o vetor $|0\rangle$ será utilizado em todo o texto como o primeiro vetor de uma base ortonormal do espaço. Por exemplo, a base $\{(1, 0), (0, 1)\}$ de \mathbb{C}^2 pode ser vista nessa nova linguagem como: $\{|v_0\rangle, |v_1\rangle\}$, onde $|v_0\rangle = (1, 0)$ e $|v_1\rangle = (0, 1)$. O vetor $\psi = (a, b)$ será reescrito como $|\psi\rangle = a|v_0\rangle + b|v_1\rangle$. Essa notação facilita muito quando formos

fazer computação quântica. Caso o leitor não tenha conseguido dominá-la, não tem problema. Basta pensar no vetor $|\psi\rangle$ com a representação que ele esteja familiarizado.

Definição 5 *Consideremos o seguinte conjunto,*

$$H'_n = \{t : H_n \rightarrow \mathbb{C} | t \text{ é linear}\}.$$

Em H'_n podemos definir as seguintes operações: para $t, m \in H'_n$ e $\lambda \in \mathbb{C}$,

$$(t + m)(h) = t(h) + m(h), (\lambda t)(h) = \lambda t(h).$$

Com estas operações H'_n é um espaço vetorial. Chamamos H'_n de espaço dual de H_n . Os elementos de H'_n são chamados funcionais lineares.

Em [4] o leitor pode encontrar uma demonstração do seguinte fato: a dimensão de H_n e H'_n é a mesma.

Definição 6 *Sejam $L = \{f_1, \dots, f_n\}$ uma base para H'_n e $S = \{e_1, \dots, e_n\}$ uma base para H_n . Se $f_i(e_j) = \begin{cases} 0, & \text{se } i \neq j \\ 1, & \text{se } i = j \end{cases}$, dizemos que L é a base de H'_n dual a base S de H_n , ou simplesmente L é a base dual, se ficar claro qual é a base de H_n que está sendo considerada.*

O produto interno (\cdot, \cdot) é uma função de $H_n \times H_n \rightarrow \mathbb{C}$ que satisfaz as seguintes propriedades:

- 1- (\cdot, \cdot) é linear na segunda entrada, $(|v\rangle, \sum_i \lambda_i |w_i\rangle) = \sum_i \lambda_i (|v\rangle, |w_i\rangle)$;
- 2- $(|v\rangle, |w\rangle) = (|w\rangle, |v\rangle)^*$; onde $*$ é para denotar a conjugação complexa.
- 3- $(|v\rangle, |v\rangle) \geq 0$, com a igualdade valendo se, e somente se, $|v\rangle = 0$.

Em H_n usaremos o seguinte produto interno $((v_1, \dots, v_n), (z_1, \dots, z_n)) = \sum_i v_i^* z_i$. No produto interno se mantivermos fixo o elemento da primeira entrada, isto é, $(|v\rangle, \cdot)$ teremos um funcional linear l_v dado por,

$$\begin{aligned} l_v : H_n &\rightarrow \mathbb{C} \\ |w\rangle &\mapsto (|v\rangle, |w\rangle). \end{aligned}$$

Dada uma base ortonormal $\{|h_1\rangle, |h_2\rangle, \dots, |h_n\rangle\}$, para H_n , utilizando o produto interno definiremos os seguintes funcionais lineares:

$$H = \{(|h_1\rangle, \cdot), (|h_2\rangle, \cdot), \dots, (|h_n\rangle, \cdot)\}.$$

É fácil ver que esse conjunto é linearmente independente, e como a dimensão de H'_n é n , temos que H é uma base para H'_n . Mais ainda, a base escolhida para H_n é ortonormal logo, H coincide com a base dual. Como H_n e H'_n possuem a mesma dimensão eles são isomorfos. Um dos isomorfismos existente é $L : |h_i\rangle \mapsto (|h_i\rangle, \cdot)$, conhecido como isomorfismo canônico. Portanto, $|v\rangle = \sum_{i=1}^n \alpha_i |h_i\rangle$ através de L é mapeado no seguinte elemento $\sum_{i=1}^n \alpha_i^* (|h_i\rangle, \cdot)$ de H'_n . Esse elemento é conhecido como o **dual** do vetor $|v\rangle$ e denotado por $\langle v|$. Com essa nova notação $H = \{\langle h_1|, \langle h_2|, \dots, \langle h_n|\}$ e $\langle v| = \sum_{i=1}^n \alpha_i^* \langle h_i|$. Desse modo, podemos denotar o produto escalar de $|v\rangle$ e $|w\rangle$ por $\langle v | w \rangle$.

Nosso próximo objetivo é encontrar uma representação matricial para $\langle v | w \rangle$. Sejam $|v\rangle = \sum_i v_i |i\rangle$ e $|w\rangle = \sum_i w_i |i\rangle$ representações de $|v\rangle$ e $|w\rangle$ em uma base ortonormal $|i\rangle$. Logo, como $(|i\rangle, |j\rangle) = \delta_{ij} = \langle i | j \rangle$, temos:

$$\begin{aligned} \langle v | w \rangle &= \left(\sum_i v_i |i\rangle, \sum_i w_i |i\rangle \right) = \sum_{ij} v_i^* w_j \delta_{ij} \\ &= \sum_i v_i^* w_i = \begin{bmatrix} v_1^* & \dots & v_n^* \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}. \end{aligned}$$

Assim, podemos representar o dual $\langle v|$ como o vetor linha cujas entradas são os complexos conjugados das entradas correspondentes de $|v\rangle$. Se quiséssemos calcular o produto interno de $a = (0, i)$ e $b = (1, 0) \in \mathbb{C}^2$ com a base canônica, bastaria fazer $\langle a | b \rangle = \begin{bmatrix} 0 & -i \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, que produz como resultado 0. Observe que $\langle i | j \rangle = 0$ sempre que $|i\rangle$ e $|j\rangle$ forem vetores ortogonais. Logo se o espaço vetorial V é considerado com uma base ortonormal $\{|v_1\rangle, \dots, |v_m\rangle\}$, os vetores $|\psi\rangle$ e $|\phi\rangle \in V$ serão dados por $|\psi\rangle = \sum_i \psi_i |v_i\rangle$ e $|\phi\rangle = \sum_i \phi_i |v_i\rangle$. Fazendo o produto interno temos, $\langle \psi | \phi \rangle = \sum_{i,j} \psi_i^* \phi_j \langle v_i | v_j \rangle = \sum_i \psi_i^* \phi_i$. Isso ilustra um dos benefícios dessa notação.

Seja V um espaço vetorial com base ortonormal $\{|v_1\rangle, \dots, |v_m\rangle\}$. Nesse espaço temos o operador linear identidade, que satisfaz:

$$I = \sum_i |v_i\rangle \langle v_i|, \quad (1.12)$$

(para uma demonstração desse fato veja [3]). Por exemplo, a matriz identidade de \mathbb{C}^2 , com a base ortonormal $\{|0\rangle, |1\rangle\}$ será expressa por $I = |0\rangle \langle 0| + |1\rangle \langle 1|$.

Seja W um subespaço de dimensão l de um espaço vetorial V com dimensão v e considere $\{|1\rangle, \dots, |l\rangle\}$ uma base ortonormal para W . Definimos um **projektor** sobre W , como sendo o seguinte operador linear:

$$P = \sum_{i=1}^l |i\rangle \langle i|.$$

Definição 7 *Sejam V um espaço vetorial e $U : V \rightarrow V$ uma transformação linear, U será chamada de transformação linear unitária se*

$$|U|v\rangle| = ||v\rangle|, \quad \forall |v\rangle \in V.$$

Esta definição é equivalente a verificar se a matriz que representa U satisfaz: $U^\dagger U = I$, onde $U^\dagger = [U^t]^*$, com t indicando a transposição matricial e $*$ indicando a conjugação complexa. Nesse texto, não faremos distinção entre a transformação linear U e a matriz que a representa, a menos que seja explicitada uma menção ao contrário.

1.5.1 Produto tensorial

No próximo capítulo precisaremos da definição de produto tensorial, pois será através dessa operação que descreveremos um sistema composto por mais de um subsistema. Esse assunto será tratado de uma forma bem simplificada.

O produto tensorial de dois vetores

$$|\psi\rangle = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_m \end{bmatrix} \text{ e } |\phi\rangle = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix}, \quad (1.13)$$

denotado por $|\psi\rangle \otimes |\phi\rangle$, é um outro vetor $|\alpha\rangle$ definido por:

$$|\alpha\rangle = |\psi\rangle \otimes |\phi\rangle = \begin{bmatrix} \psi_1\phi_1 \\ \psi_1\phi_2 \\ \vdots \\ \psi_1\phi_n \\ \psi_2\phi_1 \\ \psi_2\phi_2 \\ \vdots \\ \psi_2\phi_n \\ \vdots \\ \psi_m\phi_1 \\ \psi_m\phi_2 \\ \vdots \\ \psi_m\phi_n \end{bmatrix}, \quad (1.14)$$

onde $\psi_i\phi_j$ é o produto de números complexos. Também denotaremos $|\psi\rangle \otimes |\phi\rangle$ por $|\psi\phi\rangle$. Podemos estender a definição de produto tensorial de vetores para matrizes. Seja A uma matriz $m \times n$, e B uma matriz $p \times q$.

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{bmatrix}_{m \times p \times n \times q}. \quad (1.15)$$

Sejam V e W espaços vetoriais de dimensões m e n respectivamente, com as seguintes bases $\{|v_1\rangle, \dots, |v_m\rangle\}$ e $\{|w_1\rangle, \dots, |w_n\rangle\}$. O espaço vetorial $V \otimes W$ é gerado por combinações lineares dos seguintes elementos $|v_i\rangle \otimes |w_j\rangle$, chamaremos $\{|v_i\rangle \otimes |w_i\rangle\}$ de base produto para $V \otimes W$.

O produto tensorial satisfaz as seguintes propriedades, para $z \in \mathbb{C}$, $|v\rangle, |v_1\rangle, |v_2\rangle \in \mathbb{C}^m$ e $|w\rangle, |w_1\rangle, |w_2\rangle \in \mathbb{C}^n$:

1. $z(|v\rangle \otimes |w\rangle) = (z|v\rangle) \otimes |w\rangle = |v\rangle \otimes (z|w\rangle)$.
2. $(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle$.

$$3. |v\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle.$$

Sejam A e B transformações lineares agindo respectivamente nos seguintes espaços vetoriais V e W . O operador linear definido por $A \otimes B$ agirá no espaço $V \otimes W$ da seguinte maneira: $A \otimes B(|v\rangle \otimes |w\rangle) = A|v\rangle \otimes B|w\rangle$, estendido por linearidade. O produto escalar em $V \otimes W$ é definido na base produto como, $\langle v_i w_j | v_k w_l \rangle = \langle v_i | v_k \rangle \langle w_j | w_l \rangle$. Para maiores detalhes veja [3].

Capítulo 2

Um breve relato quântico

Neste capítulo são apresentados conhecimentos básicos necessários sobre mecânica e computação quântica para a compreensão da parte quântica do algoritmo de Shor.

2.1 Comentários sobre mecânica quântica

Por muito tempo a física Newtoniana descreveu bem os fenômenos físicos observáveis. Mas, a partir do primeiro quarto do século XIX alguns resultados discordavam do previsto pela teoria existente. Na tentativa de explicar essas discordâncias nasceu a mecânica quântica. Grosseiramente falando, essa teoria explica os fenômenos físicos na escala atômica. Normalmente atribui-se à mecânica quântica apenas tais fenômenos, porque existe uma teoria consistente para descrever os fenômenos do dia-a-dia: a mecânica clássica. Na maioria dos casos, a mecânica clássica pode ser obtida da mecânica quântica. Porém, como a mecânica quântica não é muito intuitiva, nos fenômenos que a clássica explica é preferível trabalhar com esta teoria.

A mecânica quântica torna-se mais palpável quando descrevemos seus fenômenos através de uma linguagem matemática. Toda teoria possui seus pilares, os da mecânica quântica são os seguintes postulados:

Postulado 1 *A qualquer sistema físico isolado existe associado um espaço vetorial complexo com produto interno, conhecido como espaço de estados do sistema. O sistema é completamente descrito pelo seu vetor de estado, um vetor unitário no espaço de estados.*

Postulado 2 *A evolução de um sistema quântico fechado é descrita por uma transformação linear unitária. Ou seja, o estado $|\psi\rangle$ de um sistema em um tempo t_1 está relacionado ao estado $|\psi'\rangle$ do sistema em t_2 por um operador linear unitário U que depende somente de t_1 e t_2 :*

$$|\psi'\rangle = U |\psi\rangle.$$

Postulado 3 *O espaço de estados de um sistema físico composto é o produto tensorial dos espaços de estados dos sistemas individuais. Se os sistemas forem numerados de 1 até n , e o sistema i for preparado no estado $|\psi_i\rangle$, decorre que o estado do sistema composto será $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle$.*

Postulado 4 *Seja $|\psi\rangle$ o estado do sistema, onde $|\psi\rangle \in V$. Sejam V_i subespaços vetoriais de V tais que $V = \bigoplus V_i$. Uma medição com possíveis respostas $\{i\}$, aplicada ao sistema é descrita pelos seus projetores ortogonais P_i , nos respectivos subespaços V_i . A probabilidade de obter o resultado i é dada por $p(i) = \langle \psi | P_i | \psi \rangle$. Após a medição o sistema passa a ser descrito pelo estado $|\psi_i\rangle = \frac{P_i |\psi\rangle}{|P_i |\psi\rangle|}$.*

O postulado 1 descreve a arena para a mecânica quântica, nos ensinando como um sistema deve ser descrito. O postulado 2 nos diz que a evolução de um sistema é feita através de transformações lineares unitárias. Com isso se soubermos em um tempo t_1 o estado de um sistema, poderemos descobrir qual era o seu estado em um tempo t_0 anterior. O postulado 3 nos diz como descrever um sistema composto por mais de um subsistema. O postulado 4 permite extrair informações sobre o sistema que estamos trabalhando.

Vamos explorar um pouco o postulado 4, pois ele será muito utilizado nesse trabalho. Suponha que temos um sistema descrito por $|\psi\rangle = a|0\rangle +$

$b|1\rangle$, onde o espaço de estados está sendo considerado com a seguinte base $\{|0\rangle, |1\rangle\}$ ortonormal (conhecida como base computacional, a motivação para esse nome será dada mais adiante nesse texto). Faremos uma medição nesse sistema, ela terá dois possíveis resultados, definidos pelos projetores $P_0 = |0\rangle\langle 0|$ e $P_1 = |1\rangle\langle 1|$. A probabilidade de obter o resultado 0 na medição é: $p(0) = \langle \psi | P_0 | \psi \rangle = |a|^2$. Analogamente, a probabilidade de obter o resultado 1 é $p(1) = |b|^2$. O estado do sistema após a medição em cada caso será:

$$\frac{P_0 |\psi\rangle}{|a|} = \frac{a |0\rangle}{|a|}$$

$$\frac{P_1 |\psi\rangle}{|b|} = \frac{b |1\rangle}{|b|}.$$

Note que após a medição não temos mais informação sobre o resultado que não foi obtido na medição. Perdemos toda a informação a respeito dele. Esse é um dos motivos porque é tão complexo trabalhar com fenômenos quânticos. Pois, para termos informações a seu respeito temos que fazer medições, porém quando medimos perdemos informações potenciais.

2.2 Computação quântica

Atualmente, o computador se tornou algo extremamente importante em nossa vida. Ele está presente em todos os lugares. O computador que temos em casa é um objeto físico, o qual pode ter as suas ações traduzidas em linguagem matemática. Na computação atual o **bit** é o conceito fundamental, e assume apenas os estados 0 ou 1.

Na computação quântica temos o análogo ao bit na computação clássica, ele será chamado de bit quântico ou q-bit.

Definição 8 *Seja H_2 com a base ortonormal $\{|0\rangle, |1\rangle\}$. Um q-bit ou bit quântico $|\phi\rangle$ é um vetor unitário em H_2 , isto é,*

$$|\phi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle,$$

com $\alpha_0, \alpha_1 \in \mathbb{C}$ e $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Chamamos o coeficiente complexo α_j de amplitude do estado $|j\rangle$, para $j = 0, 1$.

Na computação clássica o bit está no estado 0 ou 1 e podemos descobrir facilmente em qual estado ele se encontra. Mas, o q-bit encontra-se como uma combinação linear dos estados $|0\rangle$ e $|1\rangle$. Se quisermos determinar os valores de α e β do q-bit $|\phi\rangle$ não conseguiremos. Pelo postulado 4, quando examinamos um q-bit na tentativa de encontrar os valores de α_0 e α_1 temos acesso apenas ao estado $|0\rangle$ com probabilidade $|\alpha_0|^2$ e ao estado $|1\rangle$ com probabilidade $|\alpha_1|^2$. Com isso, quando medimos um q-bit perdemos algumas informações que ele armazena. Pois, após a medição o q-bit estará em um dos estados $|0\rangle$ ou $|1\rangle$. Uma maneira de tentar obter α_0 e α_1 seria fazer várias cópias de $|\psi\rangle$ e medí-las. Com isso teríamos um valor aproximado para α_0 e α_1 . Mas, o processo de copiar um estado quântico não é permitido pelo teorema da não-clonagem que será apresentado na seção 2.3. O q-bit pode existir em um estado contínuo entre $|0\rangle$ e $|1\rangle$, até que ele seja observado. Por exemplo, um q-bit pode estar no estado $\frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$, quando fizermos uma medição com os projetores $M_0 = |0\rangle\langle 0|$ e $M_1 = |1\rangle\langle 1|$ teremos probabilidade $\frac{1}{3}$ de encontrar 0 e $\frac{2}{3}$ de encontrar 1.

Agora falaremos um pouco sobre sistemas que são descritos por mais de um q-bit. Se tivermos dois bits clássicos os possíveis estados seriam 00, 01, 10 ou 11. Um sistema com um par de q-bits é descrito como um vetor unitário $|\phi\rangle \in H_4$, onde H_4 é considerado com a base ortonormal $B_4 = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Desse modo,

$$|\phi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$

onde $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$. Cada α_{ij} , com $i, j \in \{0, 1\}$ é chamado de amplitude do estado $|ij\rangle$. Podemos representar o estado $|00\rangle$ pelo seu número correspondente na base binária, isto é, $|0\rangle$. Fazendo isso para os outros vetores da base temos que B_4 passa a ser $\{|0\rangle, |1\rangle, |2\rangle, |3\rangle\}$ e $|\phi\rangle$ pode ser reescrito como $\alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle + \alpha_3|3\rangle$.

Enunciaremos como descrever um sistema com n q-bits.

Definição 9 *Seja $|\phi\rangle$ o estado de um sistema com n q-bits, $|\phi\rangle$ é um vetor unitário em H_{2^n} com uma base ortonormal $B_{2^n} = \{|x\rangle; x \in \{0, 1\}^n\}$, o*

significado de $x \in \{0, 1\}^n$ é que x é uma sequência de n algarismos, onde esses algarismos são 0 ou 1. Desse modo,

$$|\phi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle, \quad (2.1)$$

onde

$$\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1, \quad (2.2)$$

α_x é conhecido como amplitude do estado da base $|x\rangle$.

Se trocarmos x pelo seu valor numérico na base binária as expressões (2.1) e (2.2) tornam-se:

$$|\phi\rangle = \sum_{x=0}^{2^n-1} \alpha_x |x\rangle \text{ e } \sum_{x=0}^{2^n-1} |\alpha_x|^2 = 1.$$

Daqui para frente, os espaços vetoriais serão sempre de dimensão 2^n .

2.3 Portas Lógicas

Na computação clássica as portas lógicas são funções binárias com entradas binárias, isto é, $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$. Nesta computação existem apenas duas portas lógicas reversíveis operando sobre um bit: a porta identidade e a negação. Na computação quântica as portas lógicas de um q-bit são transformações lineares unitárias em H_2 . A necessidade de ser uma transformação linear unitária é devido ao postulado 2.

Uma porta lógica que será de muito interesse nesse trabalho é a porta Hadamard,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Essa porta transforma $|0\rangle$ em $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ e $|1\rangle$ em $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

Analogamente, as portas lógicas que atuam em n q-bits são transformações lineares unitárias de \mathbb{C}^{2^n} em \mathbb{C}^{2^n} . Uma porta lógica quântica de 2-bits é o NÃO-controlado ou porta CNOT. Essa porta é um operador de \mathbb{C}^4 em \mathbb{C}^4 .

Ela têm dois q-bits de entrada, conhecidos como q-bit de controle e q-bit alvo. Descreveremos a ação da porta CNOT na base computacional do seguinte modo: se o q-bit de controle estiver no estado $|0\rangle$ nada acontece com o q-bit alvo. Se o q-bit de controle for o estado $|1\rangle$, o q-bit alvo troca de estado, onde o q-bit de controle é o primeiro q-bit. Em símbolos:

$$|00\rangle \mapsto |00\rangle, |01\rangle \mapsto |01\rangle, |10\rangle \mapsto |11\rangle, |11\rangle \mapsto |10\rangle.$$

Também podemos descrever a porta CNOT da seguinte maneira: $|A\rangle \otimes |B\rangle \mapsto |A\rangle \otimes |B + A \pmod{2}\rangle$. A matriz unitária que representa essa porta é:

$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Pelo fato de que todas as portas lógicas da computação quântica serem transformações lineares unitárias essa computação é reversível, ao contrário da computação clássica que é usualmente irreversível [3].

Agora responderemos porque não é possível copiar qualquer q-bit desconhecido. Suponha que temos uma máquina que realiza tal operação. Seja A o compartimento da máquina que armazena o q-bit desconhecido $|\phi\rangle$ que desejamos copiar e B o compartimento que irá armazenar a cópia de $|\phi\rangle$, o estado inicial do compartimento B será $|t\rangle$. O estado inicial da máquina é:

$$|\phi\rangle \otimes |t\rangle.$$

A evolução do sistema será dada por uma matriz unitária U ,

$$|\phi\rangle \otimes |t\rangle \xrightarrow{U} U(|\phi\rangle \otimes |t\rangle) = |\phi\rangle \otimes |\phi\rangle.$$

Suponha que sejam feitas cópias de dois estados, $|\phi\rangle$ e $|\psi\rangle$. Teremos:

$$U(|\phi\rangle \otimes |t\rangle) = |\phi\rangle \otimes |\phi\rangle \tag{2.3}$$

$$U(|\psi\rangle \otimes |t\rangle) = |\psi\rangle \otimes |\psi\rangle. \tag{2.4}$$

De (2.3) e (2.4) segue que,

$$\begin{aligned} (U(|\phi\rangle \otimes |t\rangle), U(|\psi\rangle \otimes |t\rangle)) &= (|\phi\rangle \otimes |\phi\rangle, |\psi\rangle \otimes |\psi\rangle) \\ (U(|\phi\rangle \otimes |t\rangle), U(|\psi\rangle \otimes |t\rangle)) &= (|\phi\rangle, |\psi\rangle)^2. \end{aligned} \quad (2.5)$$

Por hipótese, U é unitária obedecendo: $(U|a\rangle, U|b\rangle) = (|a\rangle, |b\rangle)$ logo, (2.5) torna-se:

$$\begin{aligned} (|\phi\rangle \otimes |t\rangle, |\psi\rangle \otimes |t\rangle) &= (|\phi\rangle, |\psi\rangle)^2 \\ \langle\phi| \psi\rangle &= (\langle\phi| \psi\rangle)^2. \end{aligned}$$

Porém, a equação $y = y^2$ possui apenas duas soluções, $y = 0$ ou $y = 1$. Logo, só temos duas possibilidades $|\phi\rangle = |\psi\rangle$ ou $|\phi\rangle$ e $|\psi\rangle$ serem ortogonais. Portanto, conseguimos copiar $|\phi\rangle$ apenas se $|t\rangle$ é múltiplo ou ortogonal a $|\phi\rangle$. Esse resultado é conhecido como **teorema da não clonagem**, visto que não é sempre que conseguimos copiar um q-bit [14].

2.4 Circuitos Quânticos

Os circuitos quânticos são compostos por **portas lógicas e fios**. Os fios não são necessariamente os objetos físicos a que estamos acostumados, eles apenas representam o transporte do q-bit através do circuito. O fio pode representar um fóton se movendo de um local para outro no espaço. O circuito deve ser lido da esquerda para a direita. O circuito (2.6) é o da porta CNOT

$$\begin{array}{c} |A\rangle \\ |B\rangle \end{array} \begin{array}{c} \text{---} \bullet \text{---} \\ \text{---} \oplus \text{---} \end{array} \begin{array}{c} |A\rangle \\ |A + B \pmod 2\rangle \end{array} \quad (2.6)$$

Se o q-bit de controle for um estado da base computacional a porta CNOT pode ser utilizada para copiá-lo para o q-bit alvo, desde que o q-bit alvo esteja no estado $|0\rangle$. Observe o circuito (2.7)

$$\begin{array}{c} |c\rangle \\ |0\rangle \end{array} \begin{array}{c} \text{---} \bullet \text{---} \\ \text{---} \oplus \text{---} \end{array} \begin{array}{c} |c\rangle \\ |c\rangle \end{array} \quad (2.7)$$

Um exemplo de um circuito maior é o (2.8), a tarefa dele é trocar o estado da primeira linha com o da segunda, e por isto é chamado de **swap**,

$$\begin{array}{c} \text{---} \\ \bullet \\ \text{---} \\ \oplus \\ \text{---} \\ \bullet \\ \text{---} \\ \oplus \\ \text{---} \end{array} \equiv \begin{array}{c} \text{---} \\ \times \\ \text{---} \\ \times \\ \text{---} \end{array} . \tag{2.8}$$

Verificaremos se o circuito (2.8) realiza a tarefa prometida. Suponha que o estado $|a\rangle \otimes |b\rangle$ seja um produto tensorial de estados da base computacional temos:

$$\begin{aligned}
 |a\rangle \otimes |b\rangle &\mapsto |a\rangle \otimes |a+b \pmod{2}\rangle \\
 &\mapsto |a+(a+b) \pmod{2}\rangle \otimes |a+b \pmod{2}\rangle \\
 &\mapsto |b\rangle \otimes |a+b+b \pmod{2}\rangle = |b\rangle \otimes |a\rangle = |ba\rangle .
 \end{aligned}$$

Apresentaremos uma operação que será muito útil nos próximos capítulos. Se tivermos uma porta lógica U atuando em n -q-bits, definiremos a porta U -controlada como uma operação que envolverá $n+1$ q-bits. O primeiro é o q-bit de controle, os n restantes são os alvos, isto é, são os q-bits nos quais a porta U atua. Se o q-bit de controle for o 0 a porta U não será aplicada aos q-bit alvo, se for 1 a porta será aplicada. O circuito (2.9) representa um circuito quântico para a porta U -controlada,

$$\begin{array}{c} \text{---} \\ \bullet \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \quad U \tag{2.9}$$

Observe que a porta U -controlada é apenas uma porta CNOT generalizada para muitos q-bits.

Considere o seguinte circuito,

$$|0\rangle \text{---} \boxed{H} \text{---} \boxed{\diagup} \tag{2.10}$$

o símbolo $\boxed{\diagup}$ é para indicar que nesse estágio do circuito será feita uma medição na base computacional. Analisaremos as ações realizadas no circuito

(2.10): iniciado no estado $|0\rangle$, após a atuação da porta Hadamard o estado passará a ser descrito por $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. No final será feita uma medição na base computacional, com o seguinte conjunto de projetores $|0\rangle\langle 0|$ e $|1\rangle\langle 1|$. Pelo postulado 4, temos probabilidade $\frac{1}{2}$ de após a medição o sistema ir para o estado $|0\rangle$, e probabilidade $\frac{1}{2}$ de após a medição o sistema ir para o estado $|1\rangle$.

Capítulo 3

Transformada de Fourier Quântica

A transformada de Fourier discreta (TFD) é uma transformação linear de \mathbb{C}^N em \mathbb{C}^N que leva o vetor (x_0, \dots, x_{N-1}) em (y_0, \dots, y_{N-1}) , onde os y_k são dados por:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{\frac{2\pi i j k}{N}}. \quad (3.1)$$

Vamos fazer alguns casos particulares para TFD:

para $N = 2$, teremos $[\frac{1}{\sqrt{2}}(x_0 + x_1, x_0 + x_1 e^{\pi i})]$;

para $N = 3$, teremos $[\frac{1}{\sqrt{3}}(x_0 + x_1 + x_2, x_0 + x_1 e^{\frac{2\pi i}{3}} + x_2 e^{\frac{4\pi i}{3}}, x_0 + x_1 e^{\frac{4\pi i}{3}} + x_2 e^{\frac{8\pi i}{3}})]$;

Como a TFD é um operador linear, podemos representá-lo em uma base de \mathbb{C}^N , por uma matriz de transformação. Faremos essa operação para a base canônica tendo, assim, a seguinte matriz:

$$\frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{N-2} & \dots & \omega \end{pmatrix},$$

onde $\omega = e^{\frac{2\pi i}{N}}$.

Sabemos da álgebra linear que um operador é unitário se, e somente se, a matriz que o representa possui seus vetores linhas ou colunas unitários e ortogonais. Utilizaremos esse fato para provar que a TFD é unitária. O vetor que está na j -ésima coluna é $\frac{1}{\sqrt{N}} \left(1, e^{\frac{2\pi i j 1}{N}}, \dots, e^{\frac{2\pi i j (N-1)}{N}} \right)$, fazendo o produto escalar com o que está na l -ésima coluna, obtemos

$$\frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{2\pi i k (l-j)}{N}}. \quad (3.2)$$

Temos dois casos:

1) $l = j$,

$$\frac{1}{N} \sum_{k=0}^{N-1} 1 = 1, \text{ isto é, o vetor coluna tem norma } 1.$$

2) $l \neq j$,

$$\frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{2\pi i k (l-j)}{N}}. \quad (3.3)$$

A expressão acima é a soma dos termos de uma progressão geométrica (p. g.) que possui como primeiro termo e razão, respectivamente, $\frac{1}{N}$ e $e^{\frac{2\pi i (l-j)}{N}}$. Usando a fórmula para a soma dos termos de uma p.g. finita temos que (3.3)

é igual a: $\frac{\frac{1}{N} - \frac{1}{N} \left(e^{\frac{2\pi i (l-j)}{N}} \right)^N}{1 - e^{\frac{2\pi i (l-j)}{N}}}$. Sabemos que,

$$e^{\frac{2\pi i (l-j)N}{N}} = e^{2\pi i (l-j)} = \cos(2\pi(l-j)) + i \operatorname{sen}(2\pi(l-j)) = 1 + 0i = 1.$$

Portanto, a expressão (3.3) é nula. Logo, os vetores de colunas diferentes são ortogonais. Concluimos que a TFD é unitária.

Podemos reescrever a TFD na notação utilizada pela mecânica quântica, apresentada na seção 1.5. O vetor $(x_0, x_1, \dots, x_{N-1}) \in \mathbb{C}^N$ com uma base ortonormal passa a ser escrito como $\sum_{j=0}^{N-1} x_j |j\rangle$. Aplicando uma TFD a este vetor obteremos $\sum_{k=0}^{N-1} y_k |k\rangle$, onde os y_k são dados pela expressão (3.1). A TFD agindo em um espaço complexo de dimensão finita com a notação da mecânica quântica é o que chamamos de **transformada quântica de Fourier (TQF)**. Observe que a TQF é unitária, logo podemos implementá-la em um computador quântico.

Em notação binária o número $j = j_1 j_2 \dots j_n = j_1 2^{n-1} + j_2 2^{n-2} + \dots + j_n 2^0$, com $j_k = 0$ ou 1 . Vamos introduzir uma notação para as frações binárias.

Representaremos $\frac{j_1}{2} + \frac{j_2}{4} + \dots + \frac{j_m}{2^m}$ por $0.j_1j_2\dots j_m$, com $j_k = 0$ ou 1 . Tendo em vista que nosso intuito é fazer computação, analisaremos a TQF para $N = 2^n$, ou seja, para uma base $\{|0\rangle, \dots, |2^n - 1\rangle\}$,

$$|j\rangle \mapsto \frac{1}{2^{\frac{n}{2}}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} |k\rangle.$$

Escrevendo o número k em sua representação binária temos:

$$\frac{1}{2^{\frac{n}{2}}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 e^{2\pi i j (\sum_{l=1}^n k_l 2^{-l})} |k_1 \dots k_n\rangle.$$

Podemos utilizar a notação padrão de produto tensorial para reescrever a expressão acima:

$$\frac{1}{2^{\frac{n}{2}}} \sum_{k_1=0}^1 \dots \sum_{k_n=0}^1 \bigotimes_{l=1}^n e^{2\pi i j k_l 2^{-l}} |k_l\rangle = \frac{1}{2^{\frac{n}{2}}} \bigotimes_{l=1}^n [\sum_{k_l=0}^1 e^{2\pi i j k_l 2^{-l}} |k_l\rangle] = \frac{1}{2^{\frac{n}{2}}} \bigotimes_{l=1}^n [|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle].$$

Escrevendo por extenso a expressão acima temos:

$$\frac{(|0\rangle + e^{2\pi i 0.j_n} |1\rangle)(|0\rangle + e^{2\pi i 0.j_{n-1}j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0.j_1j_2\dots j_n} |1\rangle)}{2^{\frac{n}{2}}}. \quad (3.4)$$

Podemos implementar a transformada de Fourier quântica para qualquer valor de N , mas quando $N = 2^n$ é mais fácil construir o circuito. O circuito da figura 3.1 é a implementação da TQF para o caso $N = 2^n$, sua construção é facilitada pela expressão (3.4).

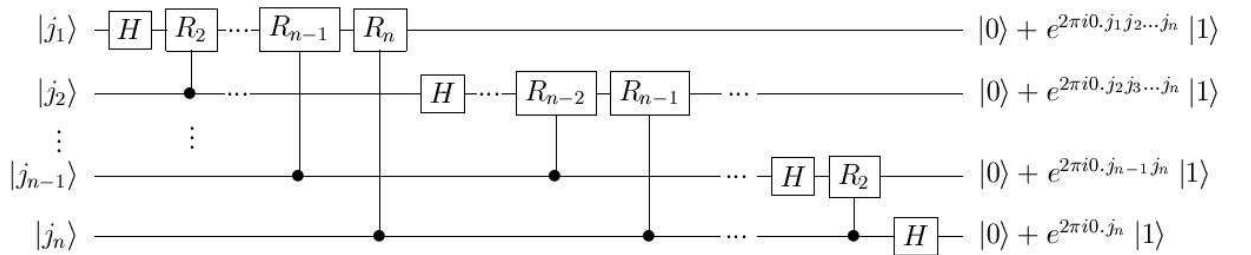


Figura 3.1: Circuito para implementar a transformada de Fourier.

O circuito é constituído de portas Hadamard e R_k controlada. A ação da porta R_k é representada pela seguinte matriz:

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}.$$

Verificaremos se o circuito acima implementa a TQF: iniciaremos com o estado $|j\rangle = |j_1 j_2 \dots j_n\rangle$. Aplicando a porta Hadamard ao primeiro q-bit temos:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_1} |1\rangle) |j_2 j_3 \dots j_n\rangle.$$

Note que se $j_1 = 0$, $e^{2\pi i 0 \cdot j_1} = 1$. No outro caso, $e^{2\pi i 0 \cdot j_1} = -1$. Em seguida aplicaremos a porta R_2 controlada pelo q-bit j_2 ao primeiro q-bit do estado acima obtendo:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2} |1\rangle) |j_2 j_3 \dots j_n\rangle.$$

Observe que ao primeiro q-bit ainda será aplicada a seguinte sequência de portas controladas: R_3, R_4, \dots, R_n . Cada porta R_k controlada multiplica o q-bit $|1\rangle$ por $e^{\frac{2\pi i}{2^k}}$ se $j_k = 1$. Caso contrário, essa porta não fará nada. Podemos reescrever a ação da porta R_k controlada como sendo multiplicar $|1\rangle$ por $e^{2\pi i 0 \cdot \overbrace{0 \dots 0}_{k-1} j_k}$. Após aplicar a sequência de portas teremos o seguinte estado:

$$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle) |j_2 j_3 \dots j_n\rangle.$$

Agora analisaremos o segundo q-bit do circuito. Após ele passar pela porta Hadamard teremos:

$$\frac{1}{2^{\frac{1}{2}}}(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle)(|0\rangle + e^{2\pi i 0 \cdot j_2} |1\rangle) |j_3 \dots j_n\rangle.$$

Em seguida ele sofrerá a ação das portas R_k controlada, com k variando de 2 até $n - 1$. Ao final teremos o estado:

$$\frac{1}{2^{\frac{1}{2}}}(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle)(|0\rangle + e^{2\pi i 0 \cdot j_2 j_3 \dots j_n} |1\rangle) |j_3 \dots j_n\rangle.$$

Continuando desta forma, chegaremos ao estado:

$$\frac{1}{2^{\frac{n}{2}}}(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle)(|0\rangle + e^{2\pi i 0 \cdot j_2 j_3 \dots j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle).$$

Se o circuito implementasse a TQF, ao final ele teria de retornar o estado da expressão (3.4). O estado acima se parece com da expressão a menos da ordem dos fatores. Para corrigir esse detalhe basta acrescentar algumas portas swap (isto é, portas de troca) ao circuito, como no ilustrado na figura 3.2, para o caso quando n é ímpar,

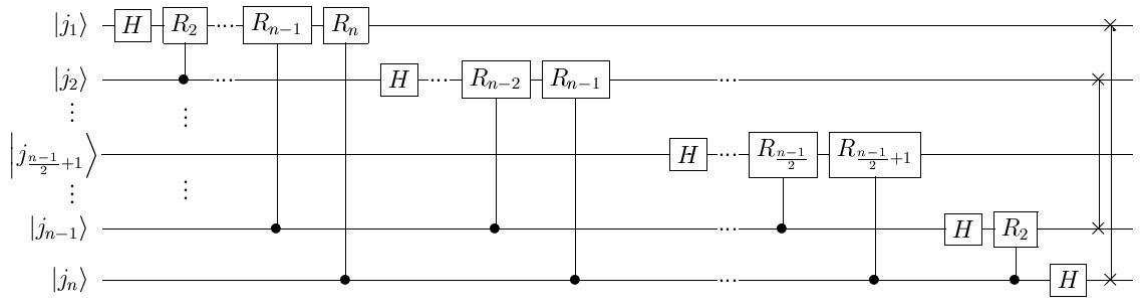


Figura 3.2: Circuito da transformada de Fourier para n ímpar.

Vamos fazer um exemplo da TQF e do seu circuito para o caso $n = 3$. Logo, a TQF é um operador linear de \mathbb{C}^8 em \mathbb{C}^8 , e a matriz que o representa na base $\{|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle\}$ é:

$$\frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix},$$

onde $\omega = e^{\frac{2\pi i}{8}}$.

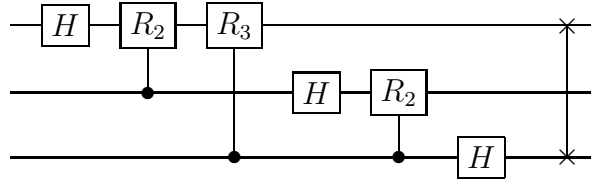
Aplicando a TQF as coordenadas do vetor $|000\rangle$ temos:

$$\frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

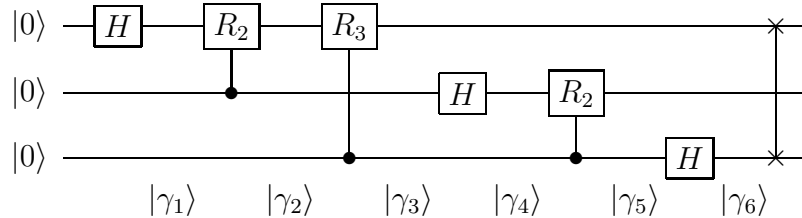
Escrevendo o vetor resultante acima em termos da base temos:

$$\frac{1}{\sqrt{8}} (|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |110\rangle + |101\rangle + |111\rangle). \quad (3.5)$$

O circuito abaixo é o que implementa a TQF em \mathbb{C}^8 :



Agora, vamos calcular a TQF em $|000\rangle$ através do circuito:



Temos a seguinte sequência de estados:

$$\begin{aligned} |\gamma_1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) |00\rangle \\ |\gamma_2\rangle &= |\gamma_3\rangle = |\gamma_1\rangle, \text{ isto acontece porque o q-bit de controle das portas controladas é } 0 \\ |\gamma_4\rangle &= \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) |0\rangle \\ |\gamma_5\rangle &= |\gamma_4\rangle \\ |\gamma_6\rangle &= \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) \\ &= \frac{1}{\sqrt{8}}(|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |110\rangle + |101\rangle + |111\rangle). \end{aligned}$$

Observe que já temos o resultado da equação (3.5), neste caso, a porta de troca atuará como a identidade.

Sabemos que a TQF é um isomorfismo linear, logo possui uma inversa, a TQF^\dagger (transformada quântica de Fourier inversa). Vamos escrever qual é a matriz da TQF^\dagger em uma base ortonormal. Como a TQF é unitária, a matriz da TQF^\dagger é igual a da TQF transposta e conjugada, como pode ser vista abaixo,

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \tau & \tau^2 & \cdots & \tau^{N-1} \\ 1 & \tau^2 & \tau^4 & \cdots & \tau^{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \tau^{N-1} & \tau^{N-2} & \cdots & \tau \end{pmatrix},$$

onde $\tau = e^{\frac{-2\pi i}{N}}$.

Logo a TQF^\dagger é uma transformação linear de \mathbb{C}^N em \mathbb{C}^N , que age nos vetores escritos em uma base ortonormal da seguinte maneira:

$$\sum_{k=0}^{N-1} a_k |k\rangle \mapsto \sum_{j=0}^{N-1} b_j |j\rangle \quad (3.6)$$

onde os $b_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{\frac{-2\pi i j k}{N}}$. Essa transformação será de grande importância quando formos discutir o algoritmo de Shor.

Para construir o circuito da TQF^\dagger devemos lembrar que ele tem que desfazer a ação da TQF. Logo, a primeira porta desse circuito tem que ser a inversa da última porta que operou no circuito da TQF, isto é, a inversa da porta swap, que é ela mesma. O mesmo ocorre para as portas seguintes que são swap. A próxima porta é a inversa da Hadamard, que também é a própria. Em seguida teremos a inversa da R_2 controlada, que será denotada por R_2^\dagger . Procedendo desta forma temos o circuito da figura 3.3.

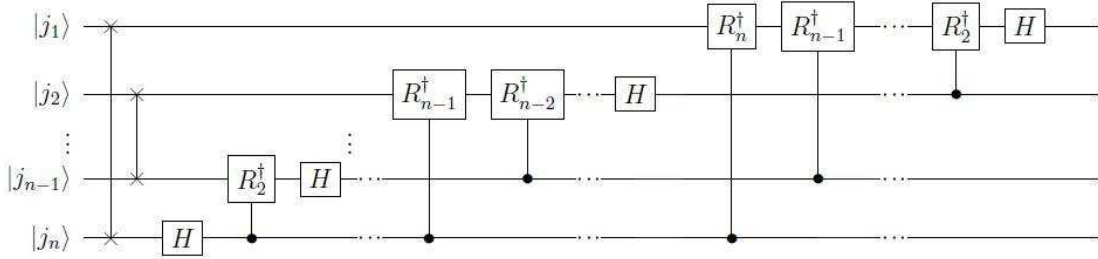


Figura 3.3: Circuito para implementar a transformada de Fourier inversa

3.1 Complexidade da Transformada Quântica de Fourier

Na computação clássica vimos alguns princípios que norteavam a contagem da complexidade de um algoritmo. A idéia principal era contar como os loops aumentavam com o tamanho da entrada. Na computação quântica é mais conveniente relacionar a complexidade do algoritmo a quantidade de portas do circuito que o implementa.

Definição 10 *A complexidade de um algoritmo quântico é o número de portas de um e dois q-bits que o circuito possui.*

Como a complexidade da TQF influenciará na complexidade do algoritmo de Shor, faremos sua análise. A complexidade da TQF^\dagger e da TQF são iguais, por isso, faremos somente a análise para a última. A complexidade da TQF é dada pela quantidade de portas necessárias no circuito 3.1. Como vimos, se o circuito age em n q-bits, na primeira linha temos n portas, na segunda $n - 1$ portas, prosseguindo desta forma, diminuindo uma porta a cada vez que passamos da linha j para a linha $j + 1$, até chegarmos a n -ésima linha que tem uma única porta. Somando as portas temos $n + (n - 1) + (n - 2) + \dots + 2 + 1 = \frac{n \cdot (n + 1)}{2}$. Além dessas portas, existem as portas swap, que são no máximo $\frac{n}{2}$. Cada porta swap é constituída de 3 portas CNOT. Logo, a quantidade de portas são: $\frac{n \cdot (n + 1)}{2} + 3 \cdot \frac{n}{2}$ que é $O(n^2)$. Portanto, a TQF é eficiente.

Você deve estar se perguntando porque é dada tanta importância a TQF. É porque o melhor algoritmo clássico que implementa a TFD é o da transformada de Fourier rápida que é $O(n2^n)$ (para maiores detalhes veja [10]).

Capítulo 4

Algoritmo de Shor

Neste capítulo será apresentado um algoritmo para encontrar a ordem r de um número $x \in \mathbb{Z}_N^*$, conhecido como algoritmo de Shor. Daqui para frente usaremos x para denotar os elementos de \mathbb{Z}_N^* , ou \mathbb{Z} ficando a distinção no contexto; só é bom lembrar que se $x \in \mathbb{Z}_N^*$ estaremos sempre trabalhando com o resto que se deixa na divisão por N . O algoritmo de Shor é importante porque se soubermos calcular a ordem de $x \in \mathbb{Z}_N^*$ conseguiremos fatorar N , (este fato será demonstrado no capítulo 5). Antes de apresentar o algoritmo, conheceremos um de seus constituintes, a transformação linear $U : \mathbb{C}^N \rightarrow \mathbb{C}^N$, definida em uma base ortonormal por:

$$U |k\rangle = |xk \pmod N\rangle. \quad (4.1)$$

Às vezes, denotaremos U por x . A transformação linear U será uma das portas quando implementarmos o algoritmo, logo temos que mostrar que ela é uma transformação unitária. Antes, iremos ver um lema que será útil para essa demonstração.

Lema 3 *Seja B um isomorfismo linear de $\mathbb{C}^m \rightarrow \mathbb{C}^m$. Se a ação de B em uma base ortonormal for uma permutação da mesma, então B é unitária.*

DEMONSTRAÇÃO: Seja $\{|0\rangle, \dots, |m-1\rangle\}$ uma base ortonormal para \mathbb{C}^m .

$$B |i\rangle = |\sigma(i)\rangle,$$

onde $\sigma(i)$ é uma permutação de i .

Pela álgebra linear sabemos que um operador linear A é unitário se, e somente se, $|A|v\rangle| = 1, \forall |v\rangle$ tal que $||v\rangle| = 1$, para demonstração deste fato veja [7].

Seja $|s\rangle$ um vetor unitário de \mathbb{C}^m , então $|s\rangle = \sum_{i=0}^{m-1} \alpha_i |i\rangle$ e $||s\rangle|^2 = \sum_{i=0}^{m-1} \alpha_i^* \alpha_i = 1$. Calculando a norma de $B|s\rangle$, temos:

$$\begin{aligned} |B|s\rangle|^2 &= \langle s|B^\dagger B|s\rangle = \sum_{ij} \alpha_i^* \alpha_j \langle i|B^\dagger B|j\rangle \\ &= \sum_{ij} \alpha_i^* \alpha_j \langle \sigma(i)|\sigma(j)\rangle = \sum_{ij} \alpha_i^* \alpha_j \delta_{ij} \\ &= \sum_{i=0}^{m-1} \alpha_i^* \alpha_i = 1. \end{aligned}$$

Portanto, B é unitária. ■

Como $x \in \mathbb{Z}_N^*$ existe $y \in \mathbb{Z}_N^*$ tal que $xy \equiv 1 \pmod{N}$. Definiremos o operador linear $A : \mathbb{C}^N \rightarrow \mathbb{C}^N$ em uma base ortonormal, pela seguinte ação:

$$A|k\rangle = |yk \pmod{N}\rangle.$$

Verificaremos que A é o operador inverso de U ,

$$U(A|k\rangle) = U|yk \pmod{N}\rangle = |xyk \pmod{N}\rangle = |k\rangle = A(U|k\rangle).$$

Desse modo, U é um isomorfismo de \mathbb{C}^N .

Seja $\{|0\rangle, \dots, |N-1\rangle\}$ uma base ortonormal de \mathbb{C}^N . Veremos que os vetores $|xy_i \pmod{N}\rangle$, com $y_i \in \{0, \dots, N-1\}$ são uma permutação do conjunto $|y_i \pmod{N}\rangle$. Para verificar este fato, basta que $xy_i \not\equiv xy_k \pmod{N}$ se $i \neq k$. Suponha que existam y_i e y_j tais que,

$$xy_i \equiv xy_j \pmod{N}.$$

Como $x \in \mathbb{Z}_N^*$, temos:

$$\begin{aligned} yxy_i &\equiv yxy_k \pmod{N} \\ y_i &\equiv y_k \pmod{N}. \end{aligned}$$

Absurdo, pois o conjunto $\{0, \dots, N - 1\}$ é formado por elementos distintos. Portanto, a ação de U é apenas uma permutação da base inicial. Logo, pelo lema 3, U é unitária.

Para a construção do algoritmo é necessário sabermos implementar as portas U^{2^j} este problema será resolvido no apêndice. Daqui para frente assumiremos que sabemos construir tais portas.

4.1 Algoritmo de Shor para uma ordem igual a uma potência de 2

Por simplicidade, primeiro descreveremos o algoritmo no caso mais fácil, para $r = 2^s$, isto é, quando r pode ser expresso com exatamente $s + 1$ bits. Nas próximas seções será apresentado o caso geral.

No circuito da figura 4.1, $t = \lfloor \log_2 N^2 \rfloor + 1$. O primeiro registrador possui t q-bits todos no estado $|0\rangle$ e o segundo registrador é dado por $\left| \underbrace{0 \dots 0}_{L-1} 1 \right\rangle = |1\rangle$, onde $L = \lfloor \log_2 N + 1 \rfloor$.

O circuito da figura 4.1 implementa o algoritmo. Vamos escrever quais são os estados apresentados neste circuito. O estado inicial é:

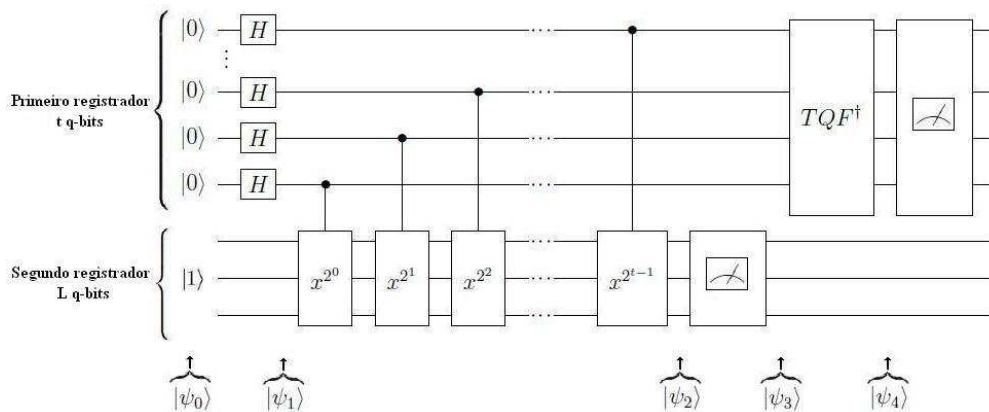


Figura 4.1: Circuito da busca de ordem.

$$|\psi_0\rangle = \underbrace{|0\dots 00\rangle}_t |1\rangle.$$

Em seguida aplicamos as portas Hadamard aos t primeiros q-bits, produzindo o estado:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |1\rangle.$$

Aplicando a sequência de portas x^{2^j} controladas a $|\psi_1\rangle$, temos:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |x^j\rangle. \quad (4.2)$$

O estado acima possui na sua soma todas as potências de x que vão de 0 até $2^t - 1$. Não é difícil notar que na soma acima estão presentes os termos,

$$|0\rangle |1\rangle, |r\rangle |1\rangle, |2r\rangle |1\rangle, \dots, \left| \left(\frac{2^t}{r} - 1 \right) r \right\rangle |1\rangle.$$

Observe que este resultado acontece porque a potência modular de um número é uma função periódica de período r . Todos os estados da equação (4.2) estão calculados, mas não conseguimos acessá-los. Pois, para sabermos qual é o estado temos que fazer uma medição. Porém, todos os estados são equiprováveis com probabilidade igual a $\frac{1}{2^t}$. No momento, estamos no estágio do algoritmo quântico em que temos a resposta, mas ainda não fizemos a pergunta certa a ele. Utilizando o fato que x^j é periódica vamos rearranjar o estado $|\psi_2\rangle$ para facilitar os cálculos,

$$|\psi_2\rangle = \frac{1}{2^{\frac{t}{2}}} \sum_{b=0}^{r-1} \left(\sum_{a=0}^{\frac{2^t}{r}-1} |ar + b\rangle \right) |x^b\rangle.$$

Na medição no segundo registrador, qualquer um dos estados $|x^0\rangle, |x^1\rangle, \dots, |x^{r-1}\rangle$ pode ser obtido com igual probabilidade. Suponha que obtemos o resultado $|x^{b_1}\rangle$. Neste caso o estado seguinte passa a ser:

$$|\psi_3\rangle = \sqrt{\frac{r}{2^t}} \left(\sum_{a=0}^{\frac{2^t}{r}-1} |ar + b_1\rangle \right) |x^{b_1}\rangle.$$

Note que se tivéssemos obtido x^{b_2} onde $b_2 \neq b_1$ o estado do termo entre parênteses teria o mesmo formato, isto é, $|ar + b_2\rangle$, ele continuaria carregando a informação que queremos r . Porém, ainda temos algumas informações que atrapalham a nossa descoberta, como por exemplo, o valor de a que é uma variável e não uma constante. Mas, esses problemas serão solucionados com a aplicação da TFQ[†] ao primeiro registrador, fazendo essa operação obtemos:

$$\begin{aligned} |\psi_4\rangle &= \sqrt{\frac{r}{2^t}} \left(\sum_{a=0}^{\frac{2^t}{r}-1} \frac{1}{2^{\frac{t}{2}}} \sum_{k=0}^{2^t-1} e^{\frac{-2\pi i(ar+b_1)j}{2^t}} |j\rangle \right) |x^{b_1}\rangle \\ &= \frac{1}{\sqrt{r}} \left(\sum_{j=0}^{2^t-1} \left[\frac{1}{\frac{2^t}{r}} \sum_{a=0}^{\frac{2^t}{r}-1} e^{\frac{-2\pi ija}{\frac{2^t}{r}}} \right] e^{\frac{-2\pi ijb_1}{2^t}} |j\rangle \right) |x^{b_1}\rangle. \end{aligned}$$

Utilizando o fato de que o termo entre colchetes é a soma de uma p.g., como na expressão (3.2) ele é diferente de zero se, e somente se, $j = \frac{k2^t}{r}$ com $k = 0, 1, \dots, r-1$. Se j assume tais valores, a expressão entre colchetes é igual a 1. E, o estado acima pode ser escrito como:

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \left(\sum_{k=0}^{r-1} e^{-2\pi i \frac{k}{r} b_1} \left| \frac{k2^t}{r} \right\rangle \right) |x^{b_1}\rangle. \quad (4.3)$$

Agora é só fazer uma medição na base computacional. Se o resultado for $|0\rangle$, quando $k = 0$, nada poderemos dizer. Mas, se o resultado é $\left| \frac{k2^t}{r} \right\rangle$, para algum $k > 0$, dividimos por 2^t , e temos duas possibilidades: $\text{mdc}(k, r) = 1$ ou $\text{mdc}(k, r) \neq 1$. No primeiro caso, selecionamos o denominador e fazemos $x^r \equiv 1 \pmod{N}$, com isso temos a ordem desejada. No segundo, o denominador será um d que é um fator de r , quando fizermos $x^d \not\equiv 1 \pmod{N}$, nesse caso o algoritmo falha.

Vamos voltar um pouco ao estado $|\psi_2\rangle$ e analisar a medição feita no segundo registrador. Se o resultado fosse x^b , tal que $b \neq b_1$ teríamos que o estado $|\psi_4\rangle$ no circuito da figura 4.1 seria:

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \left(\sum_{k=0}^{r-1} e^{-2\pi i \frac{k}{r} b} \left| \frac{k2^t}{r} \right\rangle \right) |x^b\rangle.$$

Observe que quando fizermos uma medição no estado $|\psi_4\rangle$ no primeiro registrador cada $\left|\frac{k2^t}{r}\right\rangle$ continua tendo probabilidade $\frac{1}{r}$ de ser encontrado e teremos os mesmos resultados independente do valor de b obtido. Quando fizemos a medição no segundo registrador não utilizamos seu resultado diretamente. É como se tivéssemos feito a medição e não olhado o resultado. O fato de não sabermos qual foi o resultado nessa passagem não interferiu no resultado da medição posterior do primeiro registrador. Logo, esse passo da medição no segundo registrador pode ser omitido. Sendo isto que iremos fazer no caso geral, na seção 4.2.

4.2 Caso Geral

O caso geral é mais complicado que o anterior não só pela complexidade das equações, mas também pelo método de extrair r da resposta retornada pelo algoritmo, como ficará claro mais na frente. O circuito da figura 4.2 implementa o algoritmo para o caso geral, nele o valor de t é $\lceil \log_2 N^2 \rceil + 1$ e $L = \lceil \log_2 N + 1 \rceil$. O primeiro registrador possui t q-bits todos no estado $|0\rangle$ e o segundo registrador é dado por $\left|\underbrace{0\dots 0}_{L-1}1\right\rangle = |1\rangle$. A sequência de estados no circuito da figura 4.2 é,

$$\begin{aligned} |\psi_0\rangle &= |0\rangle |1\rangle \\ |\psi_1\rangle &= \frac{1}{2^{\frac{t}{2}}} \sum_{j=0}^{2^t-1} |j\rangle |1\rangle \\ |\psi_2\rangle &= \frac{1}{2^{\frac{t}{2}}} \sum_{j=0}^{2^t-1} |j\rangle |x^j\rangle \\ |\psi_3\rangle &= \frac{1}{2^t} \sum_{j,k}^{2^t-1} e^{\frac{-2\pi i k j}{2^t}} |k\rangle |x^j\rangle \\ |\psi_4\rangle &= |c\rangle |\alpha_c\rangle, \end{aligned}$$

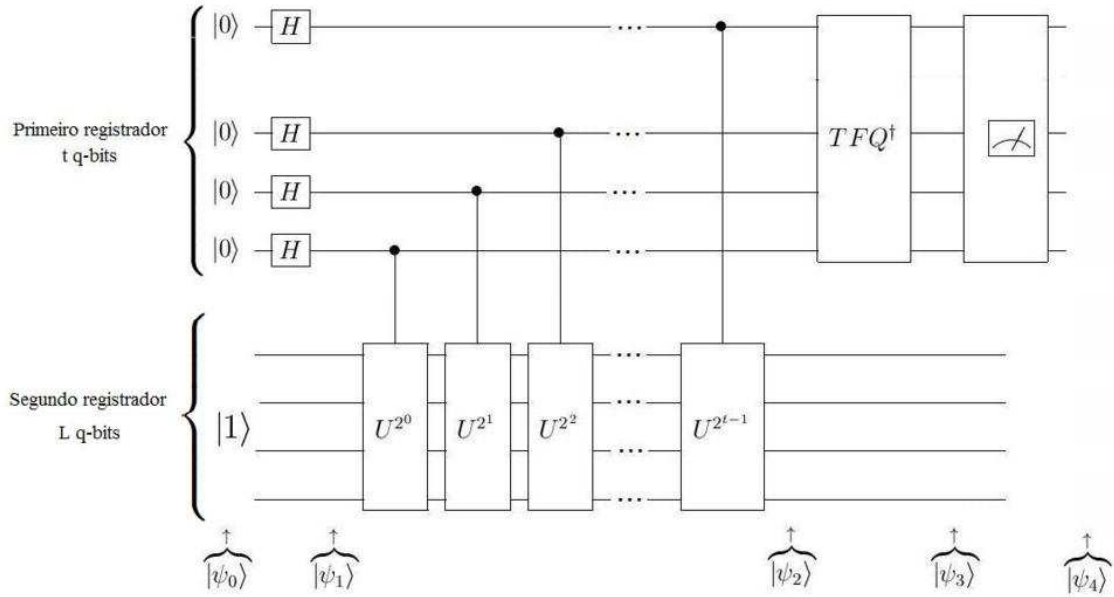


Figura 4.2: Circuito da busca de ordem no caso geral.

onde $|c\rangle \in \{|0\rangle, |1\rangle, \dots, |2^{t-1}\rangle\}$ e $|\alpha_c\rangle$ vem da decomposição de $|\psi_3\rangle$ com respeito à base computacional do primeiro registrador. O estado do primeiro registrador após a medição é o estado $|c\rangle$.

Para encontrarmos r com alta probabilidade, temos que fazer $\frac{c}{2^t}$, porém $2^t > r$. Precisamos encontrar uma maneira de obtermos r de $\frac{c}{2^t}$. Felizmente, isto não é difícil. O algoritmo de frações contínuas apresentado na seção 1.4 resolverá este problema. Para vermos isto, iremos calcular qual a probabilidade de obtermos um resultado $|c\rangle$ após a medição no primeiro registrador. Começaremos calculando a probabilidade $P(c, x^k)$ de obtermos um estado $|c\rangle |x^k\rangle$ se fizermos uma medição nos dois registradores na base computacional. Essa probabilidade é dada por:

$$P(c, x^k) = \left| \frac{1}{q} \sum_{j: x^j \equiv x^k \pmod{N}} e^{-\frac{2\pi i j c}{q}} \right|^2,$$

em que a soma é feita sobre todos os j tais que $x^j \equiv x^k \pmod{N}$, e $q = 2^t$. A ordem de x é r , logo j pode ser escrito como $j = br + k$ e o somatório pode

ser calculado da seguinte maneira:

$$\left| \frac{1}{q} \sum_{b=0}^{\frac{q-\beta}{r}} e^{\frac{-2\pi ic(br+k)}{q}} \right|^2,$$

em que β é o resto da divisão de q por r . Como o termo $e^{\frac{-2\pi ick}{q}}$ está em todos os termos do somatório podemos fatorá-lo. Além disso, seu módulo é 1, logo a expressão acima torna-se:

$$\left| \frac{1}{q} \sum_{b=0}^{\frac{q-\beta}{r}} e^{\frac{-2\pi ibrc}{q}} \right|^2, \quad (4.4)$$

Podemos utilizar o algoritmo da divisão de Euclides para dividirmos rc por q , isto é, $rc = fq + \text{resto}$. Certamente o leitor está familiarizado com o resto como um elemento do conjunto $\{0, 1, \dots, q-1\}$. Mas, não precisa ser sempre assim. Por exemplo, quando dividimos 29 por 5 podemos ter, $29 = 4.6 + 5$ ou $29 = 5.6 - 1$. Daqui para frente nesta seção toda vez que tivermos l e m e utilizarmos o algoritmo da divisão para escrever $l = m \cdot n + p$, se $p > \frac{m}{2}$ iremos reescrever essa divisão como $l = (n+1)m + p'$, em que $-\frac{m}{2} < p' < 0$. E denotaremos p' por $\{p'\}_m$, lembrando que $|\{p'\}_m| \leq \frac{m}{2}$. Logo, $rc = fq + \{rc\}_q$, em que $|\{rc\}_q| \leq \frac{q}{2}$. Pelo fato de $e^{\frac{-2\pi ifq}{q}} = e^{-2\pi if} = 1$ a expressão (4.4) é igual à:

$$\left| \frac{1}{q} \sum_{b=0}^{\frac{q-\beta}{r}} e^{\frac{-2\pi ib\{rc\}_q}{q}} \right|^2.$$

Sabemos que,

$$P(c, x^k) = \left| \frac{1}{q} \left[1 + \sum_{b=1}^{\frac{q-\beta}{r}} e^{\frac{-2\pi ib\{rc\}_q}{q}} \right] \right|^2 \geq \left| \frac{1}{q} \sum_{b=1}^{\frac{q-\beta}{r}} e^{\frac{-2\pi ib\{rc\}_q}{q}} \right| - \frac{1}{q} \right|^2. \quad (4.5)$$

Analisaremos a seguinte expressão,

$$\left| \frac{1}{q} \sum_{b=1}^{\frac{q-\beta}{r}} e^{\frac{-2\pi ib\{rc\}_q}{q}} \right|, \quad (4.6)$$

que é apenas o somatório de uma p.g., logo (4.6) torna-se:

$$\begin{aligned} & \left| \frac{1}{q} \left(\frac{e^{\frac{-2\pi i\{rc\}_q}{q}} - e^{\frac{-2\pi i(\alpha+1)\{rc\}_q}{q}}}{1 - e^{\frac{-2\pi i\{rc\}_q}{q}}} \right) \right| \\ &= \left| \frac{1}{q} \frac{e^{\frac{-2\pi i\{rc\}_q}{q}} \left(1 - e^{\frac{-2\pi i\alpha\{rc\}_q}{q}} \right)}{1 - e^{\frac{-2\pi i\{rc\}_q}{q}}} \right| \\ &= \left| \frac{1}{q} \left(\frac{1 - e^{\frac{-2\pi i\alpha\{rc\}_q}{q}}}{1 - e^{\frac{-2\pi i\{rc\}_q}{q}}} \right) \right|, \end{aligned}$$

nas equações acima $\alpha = \frac{q-\beta}{r}$, essa substituição foi feita apenas para facilitar a escrita.

Lema 4 Se $\theta \in [-\pi; \pi]$, então $|1 - e^{i\theta}| \geq \frac{2|\theta|}{\pi}$.

DEMONSTRAÇÃO: $|1 - e^{i\theta}| = |1 - \cos\theta - i\sin\theta| = \left((1 - \cos\theta)^2 + \sin^2\theta \right)^{\frac{1}{2}} = [2(1 - \cos\theta)]^{\frac{1}{2}}$. Da trigonometria sabemos que: $1 - \cos\theta = 2\sin^2\frac{\theta}{2}$. Então:

$$|1 - e^{i\theta}| = 2 \left| \sin\frac{\theta}{2} \right|.$$

Vamos verificar qual das seguintes funções é a maior $2 \left| \sin\frac{\theta}{2} \right|$ ou $\frac{2|\theta|}{\pi}$ para $-\pi \leq \theta \leq \pi$.

Como cada função é par basta analisar no intervalo $[0; \pi]$. As duas funções coincidem nos extremos e $2 \left| \sin\frac{\theta}{2} \right|$ tem concavidade para baixo neste intervalo, logo $|1 - e^{i\theta}|$ é a maior função no intervalo em questão. O gráfico da figura 4.3 ilustra a diferença das duas funções. ■

Pela demonstração do lema 4 temos que $|1 - e^{i\theta}| = 2 \left| \sin\frac{\theta}{2} \right|$, mas $|\sin x| \leq |x|$, então:

$$\left| \frac{1}{1 - e^{\frac{-2\pi i\{rc\}_q}{q}}} \right| = \frac{1}{2} \left| \frac{1}{\sin\left(\frac{-\pi\{rc\}_q}{q}\right)} \right| \quad (4.7)$$

$$\geq \frac{1}{\frac{2\pi\{rc\}_q}{q}}. \quad (4.8)$$

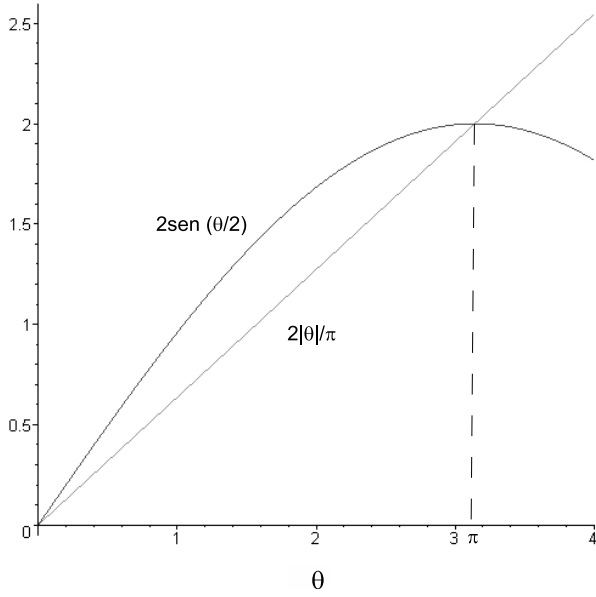


Figura 4.3: Ilustração da prova do Lema 4

Façamos uma análise da expressão:

$$\left| 1 - e^{\frac{-2\pi i \alpha \{rc\}_q}{q}} \right|,$$

lembrando que $\alpha = \frac{q-\beta}{r}$ e supondo $|\{rc\}_q| \leq \frac{r}{2}$ temos que,

$$\left| \frac{2\pi \{rc\}_q q - \beta}{q r} \right| \leq \pi.$$

Pelo lema 4, concluímos que,

$$\left| 1 - e^{\frac{-2\pi i (q-\beta) \{rc\}_q}{qr}} \right| \geq \frac{4 \{rc\}_q q - \beta}{q r}. \quad (4.9)$$

Pelas equações (4.8) e (4.9), chegamos à:

$$\left| \frac{1}{q} \sum_{b=1}^{\frac{q-\beta}{r}} e^{\frac{-2\pi i b \{rc\}_q}{q}} \right| \geq \frac{1}{q} \left(\frac{4 \{rc\}_q (q-\beta)}{qr} \right) = \frac{2(q-\beta)}{\pi q r}. \quad (4.10)$$

Então

$$\left| \left| \frac{1}{q} \sum_{b=1}^{\frac{q-\beta}{r}} e^{\frac{-2\pi i b \{rc\}_q}{q}} \right| - \frac{1}{q} \right| \geq \left| \frac{2}{\pi q} \frac{q-\beta}{r} - \frac{1}{q} \right|.$$

Pelo fato de $t = \lfloor \log_2 N^2 \rfloor + 1$ temos que $N^2 < q \leq 2N^2$. Portanto, $r < N < N^2 < q \leq 2N^2$, para N grande obtemos, $\left| \frac{2}{\pi q} \frac{q-\beta}{r} - \frac{1}{q} \right| \geq \frac{1}{\pi r}$.

Com esses cálculos concluímos que:

$$\left| \frac{1}{q} \left[1 + \sum_{b=1}^{\frac{q-\beta}{r}} e^{\frac{-2\pi i b \{rc\}_q}{q}} \right] \right| \geq \frac{1}{\pi r}.$$

Portanto, se $|\{rc\}_q| \leq \frac{r}{2}$, temos

$$P(c, x^k) \geq \frac{1}{\pi^2 r^2}. \quad (4.11)$$

Pela definição de $\{rc\}_q$, sabemos que existe um d tal que, $\{rc\}_q = rc - dq$. Além disso, $|\{rc\}_q| \leq \frac{r}{2}$. Então

$$\frac{-r}{2} \leq rc - dq \leq \frac{r}{2},$$

dividindo a equação por rq e rearranjando os termos temos,

$$\left| \frac{c}{q} - \frac{d}{r} \right| \leq \frac{1}{2q}, \quad (4.12)$$

como $r < N$ e $N^2 < q \leq 2N^2$ chegamos à

$$\left| \frac{c}{q} - \frac{d}{r} \right| \leq \frac{1}{2r^2}.$$

Portanto, pelo teorema 9, $\frac{d}{r}$ é um dos convergentes de $\frac{c}{q} = \frac{c}{2r}$. Lembre-se que os convergentes retornados pelo algoritmo de frações contínuas são tais que $\text{mdc}(p_i, q_i) = 1$. Portanto, o algoritmo irá encontrar $\frac{d}{r}$ apenas se $\text{mdc}(d, r) = 1$. Caso contrário, o algoritmo irá retornar $\frac{d'}{r'}$, em que $r' | r$, nesse caso o algoritmo falha. No caso em que $\text{mdc}(d, r) = 1$, usando o algoritmo de frações contínuas podemos calcular os convergentes de $\frac{c}{2r}$. Estamos interessados nos convergentes $\frac{c}{2r}$ que tenham denominador menor que N . Quando encontrarmos esses convergentes basta selecionarmos os denominadores a_i menores do que N e calcularmos $x^{a_i} \pmod N$ e ver qual desses números deixa resto 1 na divisão por N .

Até o momento temos que, se $|\{rc\}_q| \leq \frac{r}{2}$, $P(c, x^k) \geq \frac{1}{\pi^2 r^2}$, mas o que queremos é a probabilidade de encontrarmos $|c\rangle$. Como existem r valores para $x^k \pmod N$, temos que $P(c) \geq \frac{1}{\pi^2 r}$. Visto que o algoritmo às vezes falha, tentaremos encontrar uma cota inferior para a probabilidade dele retornar a resposta correta, isto é, r . Denotaremos essa probabilidade por $P(r)$. O lema abaixo nos auxiliará a encontrar uma cota inferior para $P(r)$.

Lema 5 *No conjunto $\{0, 1, 2, \dots, q-1\}$ existem r elementos distintos satisfazendo $|\{rc\}_q| \leq \frac{r}{2}$.*

DEMONSTRAÇÃO: Como $(r-1)q < (q-1)r$ existem r múltiplos de q no intervalo $\{0, 1, \dots, (q-1)r\}$, eles podem ser escritos como iq para $i = 0, 1, \dots, r-1$. Usaremos o algoritmo da divisão de Euclides para dividi-los por r , isto é,

$$iq = c_i r + \{iq\}_r, \quad (4.13)$$

lembre-se que $|\{iq\}_r| \leq \frac{r}{2}$. A equação (4.13) determina r múltiplos de r , sendo eles $c_i r$. Utilizaremos o algoritmo da divisão de Euclides para dividi-los por q , $c_i r = iq + \{c_i r\}_q$. Comparando esta equação com a equação (4.13) concluímos que $|\{c_i r\}_q| = |\{iq\}_r| \leq \frac{r}{2}$. Portanto, temos r múltiplos de r no conjunto $\{0, 1, \dots, (q-1)r\}$ tais que $|\{rc_i\}_q| \leq \frac{r}{2}$. Com isto, concluímos que existem r valores distintos de c_i no conjunto $\{0, 1, \dots, q-1\}$ tais que $|\{rc_i\}_q| \leq \frac{r}{2}$. ■

Da demonstração do lema 5 segue que como todos os valores de i em $\{0, 1, \dots, r-1\}$ são assumidos, na desigualdade (4.12) d assumirá todos os valores de $\{0, 1, \dots, r-1\}$. Portanto, existem r valores de c em $\{0, 1, \dots, q-1\}$ tais que $|\{rc\}_q| \leq \frac{r}{2}$ e desses r c 's existem $\phi(r)$ deles que quando fazemos $rc = dq + \{rc\}_q$, $\text{mdc}(r, d) = 1$.

A probabilidade de encontrarmos r é dada por,

$$P(r) = \sum_{c=0}^{q-1} P(r|c).P(c).$$

Como queremos uma cota inferior iremos fazer a soma acima apenas para os c 's que satisfazem as duas condições seguintes simultaneamente $|\{rc_i\}_q| \leq \frac{r}{2}$

e $\text{mdc}(r, d) = 1$. Sabemos que se c satisfaz $|\{rc_i\}_q| \leq \frac{r}{2}$, $P(c) \geq \frac{1}{\pi^2 r}$. Existem $\phi(r)$ valores de c que satisfaz simultaneamente $|\{rc_i\}_q| \leq \frac{r}{2}$ e $\text{mdc}(r, d) = 1$, se encontramos um c que satisfaz essas duas condições, obtemos r . Logo,

$$\begin{aligned}
 P(r) &\geq \sum_{c:|\{rc_i\}_q|\leq\frac{r}{2} \text{ e } \text{mdc}(r,d)=1}^q P(r|c).P(c) \\
 &\geq \sum_{\substack{c:|\{rc_i\}_q|\leq\frac{r}{2} \\ \text{mdc}(r,d)=1}}^q \frac{1}{\pi^2 r} = \frac{\phi(r)}{\pi^2 r}.
 \end{aligned}$$

Mas, $\frac{\phi(r)}{r} \geq \frac{a}{\log \log r}$, onde a é uma constante, para demonstração deste fato veja [5]. Temos que $\frac{a}{\log \log r} > \frac{a}{\log \log N}$. Portanto,

$$P(r) \geq \frac{a}{\pi^2 \log \log N}. \quad (4.14)$$

Como o leitor percebeu o algoritmo de Shor é um algoritmo probabilístico. Nem sempre ele retorna a resposta desejada, existe apenas uma probabilidade da resposta ser a correta. Esse tipo de algoritmo é diferente do tipo apresentado no capítulo 1. Lá, o algoritmo era um método prático para resolver uma tarefa e ele sempre realizava o trabalho pretendido. Quando isso acontece dizemos que o algoritmo é determinístico. Você deve estar com a impressão que os algoritmos determinísticos são melhores que os probabilísticos, visto que eles sempre retornam a resposta correta. Mas, essa não é bem a verdade. Pois, às vezes, existem algoritmos determinísticos para realizar uma determinada tarefa, porém eles não são eficientes. Por exemplo, o algoritmo para fatorar números. Existem algoritmos determinísticos para fatorar números inteiros, entretanto eles não são eficientes. Este é um caso que se existir um probabilístico eficiente já seria muito bom. No próximo capítulo veremos um algoritmo probabilístico para a fatoração de números inteiros. Com relação a eficiência dos algoritmos probabilísticos diremos que ele é eficiente se a probabilidade de sucesso for limitada inferiormente por uma função que é inversa de um polinômio. Portanto, pela equação (4.14) o algoritmo de Shor é eficiente.

Analisaremos a complexidade do algoritmo de Shor. O circuito da figura 4.2 é constituído de t portas Hadamard, mas $t \leq 2L + 1$, portanto essa etapa é $O(L)$. As portas x^{2^j} são de complexidade menor ou igual a $O(L^3)$, pois a complexidade dessas portas é determinada pelo algoritmo de exponenciação modular, que pelo apêndice é $O(L^3)$. A transformada quântica de Fourier inversa, pelo capítulo 3, é $O(L^2)$. Além disso, temos o algoritmo de frações contínuas que é $O(L^3)$. Com isso, temos que a complexidade do algoritmo é $O(L^3)$, o que nos diz que esse algoritmo é eficiente, desde que utilizado em um computador quântico.

4.3 Exemplo

Faremos um exemplo para que possamos acompanhar mais facilmente o que está acontecendo em cada passagem. Suponha que o nosso desafio seja encontrar a ordem do elemento 2 em \mathbb{Z}_{21}^* . Como $t = \lfloor \log_2 21^2 \rfloor + 1$ o seu valor será 9. Portanto a quantidade de q-bits do primeiro registrador será 9, todos inicializados no estado $|0\rangle$. O estado do segundo registrador é $|1\rangle = |00001\rangle$. A porta $U : \mathbb{C}^{2^5} \rightarrow \mathbb{C}^{2^5}$ é descrita por $U(|y\rangle) = |2y \bmod 21\rangle$. O circuito da figura 4.4 implementa a busca de ordem para este caso.

Descreveremos quais são os estados $|\psi_i\rangle$, $0 \leq i \leq 4$ no circuito da figura 4.4.

$$\begin{aligned}
 |\psi_0\rangle &= \underbrace{|00 \dots 0\rangle}_9 |1\rangle \\
 |\psi_1\rangle &= \frac{1}{\sqrt{512}} \sum_{j=0}^{511} |j\rangle |1\rangle.
 \end{aligned}$$

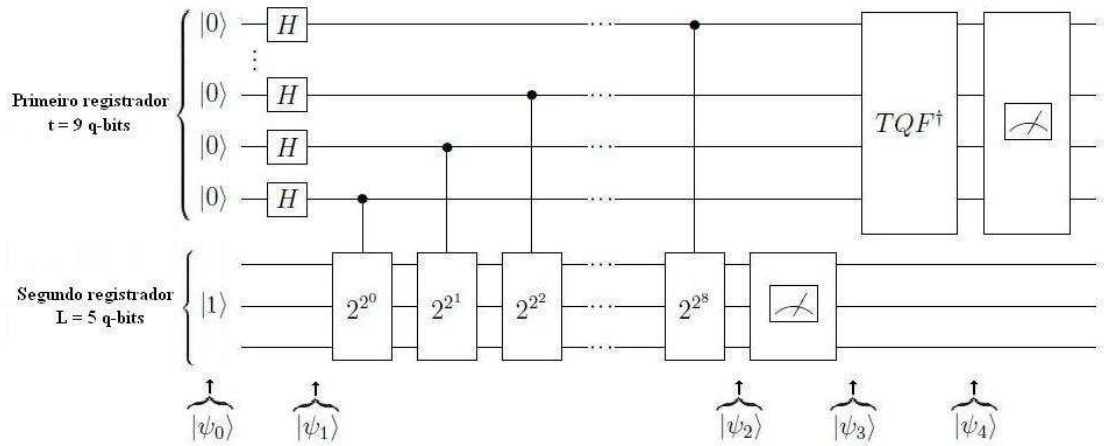


Figura 4.4: Exemplo do circuito da busca de ordem.

Após a atuação das portas controladas temos:

$$|\psi_2\rangle = \frac{1}{\sqrt{512}} \sum_{j=0}^{511} |j\rangle |2^j \pmod{21}\rangle = \frac{1}{\sqrt{512}} \left(\begin{array}{l} |0\rangle |1\rangle + |1\rangle |2\rangle + |2\rangle |4\rangle + |3\rangle |8\rangle + |4\rangle |16\rangle + |5\rangle |11\rangle + \\ |6\rangle |1\rangle + |7\rangle |2\rangle + |8\rangle |4\rangle + |9\rangle |8\rangle + |10\rangle |16\rangle + |11\rangle |11\rangle + \\ |12\rangle |1\rangle + |13\rangle |2\rangle + |14\rangle |4\rangle + |15\rangle |8\rangle + |16\rangle |16\rangle + |17\rangle |11\rangle + \\ \dots \\ |510\rangle |1\rangle + |511\rangle |2\rangle \end{array} \right).$$

Os termos do segundo registrador de cada coluna podem ser agrupados.

Deste modo, podemos reescrever os estados acima como:

$$= \frac{1}{\sqrt{512}} \left(\begin{array}{l} (|0\rangle + |6\rangle + |12\rangle + |18\rangle + |24\rangle + \dots + |510\rangle) |1\rangle + \\ (|1\rangle + |7\rangle + |13\rangle + |19\rangle + |25\rangle + \dots + |511\rangle) |2\rangle + \\ (|2\rangle + |8\rangle + |14\rangle + |20\rangle + |26\rangle + \dots + |506\rangle) |4\rangle + \\ (|3\rangle + |9\rangle + |15\rangle + |21\rangle + |27\rangle + \dots + |507\rangle) |8\rangle + \\ (|4\rangle + |10\rangle + |16\rangle + |22\rangle + |28\rangle + \dots + |508\rangle) |16\rangle + \\ (|5\rangle + |11\rangle + |17\rangle + |23\rangle + |29\rangle + \dots + |509\rangle) |11\rangle \end{array} \right).$$

Ao fazermos uma medição no segundo q-bit, temos probabilidade $\frac{1}{6}$ de encontrar quaisquer um dos seguintes estados: $|1\rangle$, $|2\rangle$, $|4\rangle$, $|8\rangle$, $|16\rangle$, $|11\rangle$.

Suponha que o resultado da medição seja $|2\rangle$. Logo,

$$|\psi_3\rangle = \frac{1}{\sqrt{86}}(|1\rangle + |7\rangle + |13\rangle + |19\rangle + |25\rangle + \dots + |511\rangle) |2\rangle = \frac{1}{\sqrt{86}} \sum_{a=0}^{85} |6a+1\rangle |2\rangle.$$

O estado $|\psi_3\rangle$ foi renormalizado. Iremos aplicar a transformada de Fourier inversa ao primeiro estado, obtendo:

$$\begin{aligned} |\psi_4\rangle &= \frac{1}{\sqrt{512}} \frac{1}{\sqrt{86}} \sum_{j=0}^{511} \left(\left[\sum_{a=0}^{85} e^{-\frac{2\pi i j(6a+1)}{512}} \right] |j\rangle \right) |2\rangle \\ |\psi_4\rangle &= \frac{1}{\sqrt{512}} \frac{1}{\sqrt{86}} \sum_{j=0}^{511} \left(\left[\sum_{a=0}^{85} e^{-\frac{2\pi i 6ja}{512}} \right] e^{-\frac{2\pi i j}{512}} |j\rangle \right) |2\rangle. \end{aligned}$$

Agora, analisaremos a probabilidade de encontrar um dado estado $|j\rangle$.

$$\begin{aligned} P(j) &= \frac{1}{512 \cdot 86} \left| \sum_{a=0}^{85} e^{-\frac{2\pi i 6ja}{512}} \right|^2 \\ &= \frac{1}{512 \cdot 86} \left| \frac{1 - e^{-\frac{2\pi i j 6 \cdot 86}{512}}}{1 - e^{-\frac{2\pi i j}{512}}} \right|^2 \\ &= \frac{1}{512 \cdot 86} \frac{1 - \cos \frac{86 \cdot 3\pi j}{128}}{1 - \cos \frac{3\pi j}{128}}. \end{aligned}$$

Pelo gráfico da figura 4.5, podemos perceber que os máximos globais da função encontram-se em torno dos seguintes valores $\frac{512k}{6}$, com $0 \leq k \leq 5$. Os quais, possuem o valor de r que estamos procurando no denominador da fração. A primeira coisa a se fazer é uma medição na base canônica. Suponha que o algoritmo retorne $j = 85$, dividindo o resultado por 512, temos $\frac{85}{512}$. Ainda não temos 6 no denominador, mas o algoritmo de frações contínuas apresentado em 1.4 resolverá nosso problema, pois ele computa os convergentes de $\frac{85}{512}$, estamos interessados nos convergentes que possuem o denominador menor que 21. Fazendo os cálculos para encontrar a fração contínua de $\frac{85}{512}$, temos:

$$\begin{aligned} \frac{85}{512} &= \frac{1}{\frac{512}{85}} \\ &= \frac{1}{6 + \frac{2}{85}} = \frac{1}{6 + \frac{1}{42 + \frac{1}{2}}} \end{aligned}$$

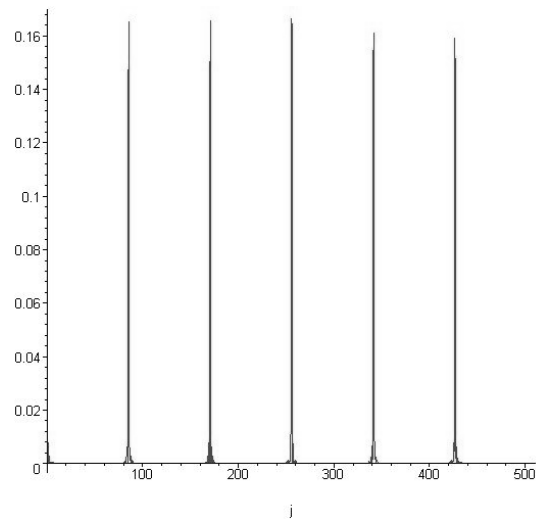


Figura 4.5: Gráfico da probabilidade de encontrar um estado j .

Logo, os convergentes são: $\frac{1}{6}$, $\frac{42}{253}$ e $\frac{85}{512}$. O de nosso interesse é $\frac{1}{6}$. Seleccionando o denominador e calculando $2^6 \equiv 1 \pmod{21}$, encontramos o valor de r . Portanto, o algoritmo retornou o resultado procurado. Mas, nem sempre é assim. Se o algoritmo tivesse retornado $j = 0$, isso não nos acrescentaria nada. Caso, o resultado encontrado fosse $j = 171$, teríamos $\frac{171}{512}$. Aplicando o algoritmo de frações contínuas, encontramos os convergentes $\frac{1}{2}$, $\frac{1}{3}$ e $\frac{171}{512}$. Calculando $2^2 \equiv 4 \pmod{21}$ e $2^3 \equiv 8 \pmod{21}$, infelizmente, não encontramos a ordem do 2, e sim 2 e 3 que são fatores da ordem, esse é o caso no qual o algoritmo falha. Se o resultado retornado for $j = 256$ teremos $\frac{256}{512} = \frac{1}{2}$. Novamente o algoritmo falha pois, o valor retornado é um fator da ordem.

Capítulo 5

Fatoração de um Número e sua Ordem

O intuito deste capítulo é mostrar que se conseguirmos resolver eficientemente em um computador o problema de encontrar a ordem de um elemento $x \in \mathbb{Z}_N^*$ automaticamente conseguiremos fatorar o número N .

Se um número a é múltiplo de b podemos escrever $a = b \cdot q$ ou equivalentemente expressar esse fato como $b|a$, que se lê b divide a . Caso contrário, falamos que b não divide a .

Apresentaremos alguns teoremas que, junto com o algoritmo da busca de ordem, nos possibilitarão construir um algoritmo quântico eficiente para fatorar um número inteiro positivo N .

Teorema 10 *Seja N um número composto com L bits, e x uma solução da congruência $x^2 \equiv 1 \pmod{N}$ no intervalo $1 < x < N - 1$. Então, um desses números $\text{mdc}(x - 1, N)$ e $\text{mdc}(x + 1, N)$ é um fator não-trivial de N que pode ser calculado com $O(L^3)$ operações.*

DEMONSTRAÇÃO: $x^2 \equiv 1 \pmod{N} \Leftrightarrow N|(x^2 - 1)$, então $N|(x + 1)(x - 1)$. Logo, N tem um fator em comum com $(x + 1)$ ou $(x - 1)$. Como $1 < x < N - 1$ esse fator é diferente de N . O algoritmo de Euclides apresentado no capítulo 1 nos permite calcular tal fator, obtendo um fator não-trivial de N , usando $O(L^3)$ operações. ■

Teorema 11 *Seja r a ordem de x em \mathbb{Z}_N^* , se existir um α tal que $x^\alpha \equiv 1 \pmod{N}$ então $r|\alpha$.*

DEMONSTRAÇÃO: Suponha que r não seja um fator de α . Pelo teorema de Euclides $\alpha = rq + r_1$, onde r_1, q são únicos e $r > r_1$. Então,

$$x^\alpha = x^{rq+r_1} = x^{rq}x^{r_1} \equiv x^{r_1} \pmod{N}.$$

Mas, $x^\alpha \equiv 1 \equiv x^{r_1} \pmod{N}$. Absurdo, pois encontramos $r_1 < r$ tal que $x^{r_1} \equiv 1 \pmod{N}$. ■

Corolário 2 *Se $x \in \mathbb{Z}_n^*$, então a ordem r de x divide $\varphi(n)$.*

DEMONSTRAÇÃO: Segue do lema acima e do teorema 2. ■

Neste capítulo algumas vezes iremos nos referir à ordem r de um elemento $x \in \mathbb{Z}_N^*$, por $ord_N x$.

Definição 11 *Se $ord_N x = \varphi(N)$ dizemos que x é uma raiz primitiva módulo N .*

Proposição 1 *Se p é um número primo e $a \in \mathbb{Z}$ é uma raiz primitiva módulo p , então a ou $a + p$ é uma raiz primitiva módulo p^2 .*

DEMONSTRAÇÃO: $a + p \equiv a \pmod{p}$, portanto $ord_p a = ord_p(a + p) = \varphi(p) = p - 1$. Sabemos que $a^{ord_p a} \equiv 1 \pmod{p^2}$, então $a^{ord_p a} \equiv 1 \pmod{p}$, pelo teorema 11

$$p - 1 | ord_{p^2} a. \tag{5.1}$$

Pelo teorema de Euler $a^{\varphi(p^2)} \equiv 1 \pmod{p^2}$, logo

$$ord_{p^2} a | \varphi(p^2) = p(p - 1). \tag{5.2}$$

Por (5.1) e (5.2) $ord_{p^2} a = p - 1$ ou $ord_{p^2} a = p(p - 1) = \varphi(p^2)$. Analogamente, $ord_{p^2}(a + p) = p - 1$ ou $ord_{p^2}(a + p) = p(p - 1) = \varphi(p^2)$.

Com isso, só nos resta mostrar que $ord_{p^2} a \neq p - 1$ ou $ord_{p^2}(a + p) \neq p - 1$. Suponha que $ord_{p^2} a = p - 1$. Portanto, $a^{p-1} \equiv 1 \pmod{p^2}$. Com isso,

$$\begin{aligned}
(a+p)^{p-1} &= a^{p-1} + \binom{p-1}{1} a^{p-2} p + \binom{p-1}{2} a^{p-3} p^2 + \dots \\
&\equiv 1 - pa^{p-2} \pmod{p^2}.
\end{aligned}$$

Devido ao fato que p^2 não divide pa^{p-2} , temos que $(a+p)^{p-1} \not\equiv 1 \pmod{p^2}$. Portanto, $\text{ord}_{p^2}(a+p) = \varphi(p^2)$. ■

Teorema 12 *Se p é um número primo ímpar e a é uma raiz primitiva módulo p^2 , então a é uma raiz primitiva módulo $p^k \forall k \in \mathbb{N}$.*

DEMONSTRAÇÃO: Sabemos que $a^{p-1} \equiv 1 \pmod{p}$ e $a^{p-1} \not\equiv 1 \pmod{p^2}$, pois a é raiz primitiva. Disso, segue que $a^{p-1} = 1 + b_1 p$, onde p não divide b_1 . Mostraremos por indução que $a^{p^{k-1}(p-1)} = 1 + b_k p^k$ e p não divide $b_k, \forall k \geq 1$. Para $k = 1$ já sabemos ser verdade. Suponhamos que $a^{p^{k-1}(p-1)} = 1 + b_k p^k$ seja verdade. Então,

$$\begin{aligned}
a^{p^k(p-1)} &= \left(a^{p^{k-1}(p-1)} \right)^p = (1 + b_k p^k)^p \\
&= 1 + \binom{p}{1} b_k p^k + \binom{p}{2} b_k^2 p^{2k} + \dots \\
&= 1 + p^{k+1}(b_k + pt),
\end{aligned}$$

para algum $t \in \mathbb{N}$. Logo, $b_{k+1} = b_k + pt$, p não divide b_{k+1} , pois por hipótese p não divide b_k .

Usando o princípio de indução provaremos que a é raiz primitiva módulo $p^k, \forall k \geq 2$. Para $k = 2$ é verdadeiro por hipótese. Suponha que a seja raiz primitiva módulo p^k .

$$a^{\text{ord}_{p^{k+1}} a} \equiv 1 \pmod{p^{k+1}} \quad (5.3)$$

$$a^{\text{ord}_{p^{k+1}} a} \equiv 1 \pmod{p^k}. \quad (5.4)$$

Logo: $\varphi(p^k) = \text{ord}_{p^k} a | \text{ord}_{p^{k+1}} a$. Mas, pelo teorema de Euler temos que $\text{ord}_{p^{k+1}} a | \varphi(p^{k+1}) = p^k(p-1)$. Assim, chegamos à:

$$p^{k-1}(p-1) | \text{ord}_{p^{k+1}} a | p^k(p-1).$$

Portanto, $ord_{p^{k+1}}a = p^{k-1}(p-1)$ ou $ord_{p^{k+1}}a = p^k(p-1) = \varphi(p^{k+1})$. Porém, $a^{p^{k-1}(p-1)} = 1 + b_k p^k$, sendo que p não divide b_k . Então $ord_{p^{k+1}}a = \varphi(p^{k+1})$. ■

Corolário 3 *Seja p um número primo ímpar e α um inteiro positivo. Resulta que $\mathbb{Z}_{p^\alpha}^*$ é cíclico.*

DEMONSTRAÇÃO: Sabemos que a ordem de $\mathbb{Z}_{p^\alpha}^*$ como grupo é $\varphi(p^\alpha)$. Pelo teorema 12 existe $a \in \mathbb{Z}_{p^\alpha}^*$ tal que $a^{\varphi(p^\alpha)} \equiv 1 \pmod{p^\alpha}$. Portanto, a é um gerador de $\mathbb{Z}_{p^\alpha}^*$, sendo $\mathbb{Z}_{p^\alpha}^*$ um grupo cíclico. ■

Lema 6 *Seja p um número primo ímpar. Seja 2^d a maior potência de 2 que divide $\varphi(p^\alpha)$. Temos que, com probabilidade exatamente igual a $\frac{1}{2}$, 2^d dividirá a ordem de um elemento x de $\mathbb{Z}_{p^\alpha}^*$ escolhido uniformemente.*

DEMONSTRAÇÃO: Temos que $\varphi(p^\alpha) = p^{\alpha-1}(p-1)$ é par, pois p é ímpar, logo $d \geq 1$. Existe um gerador g para $\mathbb{Z}_{p^\alpha}^*$, tal que qualquer elemento pode ser escrito como $g^k \pmod{p^\alpha}$ para $1 \leq k \leq \varphi(p^\alpha)$. Seja r a ordem de $g^k \pmod{p^\alpha}$, existem dois possíveis casos: k é par ou k é ímpar.

1) Se k é ímpar $g^{kr} \equiv 1 \pmod{p^\alpha}$, deduzimos que $\varphi(p^\alpha) | kr$, pois $\varphi(p^\alpha)$ é a ordem de g em $\mathbb{Z}_{p^\alpha}^*$. Assim $2^d | r$, se k é ímpar.

2) Se k é par

$$\begin{aligned} g^{k\varphi(p^\alpha)/2} &\equiv (g^{\varphi(p^\alpha)})^{k/2} \equiv 1^{k/2} \equiv 1 \pmod{p^\alpha} \\ (g^k)^{\varphi(p^\alpha)/2} &\equiv 1 \pmod{p^\alpha}, \end{aligned}$$

como r é a ordem de g^k , temos:

$$r \mid \frac{\varphi(p^\alpha)}{2} \Rightarrow 2^d \text{ não divide } r.$$

Assim $\mathbb{Z}_{p^\alpha}^*$ pode ser escrito como a união de dois conjuntos disjuntos e de mesmo tamanho.

$$\begin{aligned} A &= \{g^k \text{ tal que } k \text{ é ímpar}\} = \{g^k \text{ tal que } 2^d | r\} \text{ e} \\ B &= \{g^k \text{ tal que } k \text{ é par}\} = \{g^k \text{ tal que } 2^d \text{ não divide } r\} \end{aligned}$$

Assim com probabilidade $\frac{1}{2}$, $2^d | r$ quando r é escolhido uniformemente em $\mathbb{Z}_{p^\alpha}^*$.

O próximo teorema é importante porque ele fala sobre a existência de solução para um sistema de equações modulares.

Teorema 13 *Teorema Chinês do Resto.* [2] *Sejam m e n inteiros positivos, primos entre si. O sistema*

$$\begin{aligned}x &\equiv a \pmod{m} \\x &\equiv b \pmod{n}\end{aligned}$$

sempre tem uma única solução em \mathbb{Z}_{mn} .

Corolário 4 *Sejam m_1, \dots, m_j números naturais, dois a dois primos entre si. O sistema:*

$$\begin{aligned}x &\equiv a_1 \pmod{m_1} \\x &\equiv a_2 \pmod{m_2} \\&\vdots \\x &\equiv a_j \pmod{m_j},\end{aligned}$$

sempre tem uma única solução em $\mathbb{Z}_{m_1 \dots m_j}$.

Teorema 14 *Seja $N \in \mathbb{N}^*$ ímpar, composto e $p_1^{\alpha_1} \dots p_m^{\alpha_m}$ a sua fatoração. Seja x um número de \mathbb{Z}_N^* escolhido aleatoriamente e uniformemente, sendo r a ordem de x . Então, $P(r \text{ é par e } x^{r/2} \not\equiv -1 \pmod{N}) \geq 1 - \frac{1}{2^{m-1}}$.*

DEMONSTRAÇÃO: Vamos mostrar que $P(r \text{ é ímpar ou } x^{r/2} \equiv -1 \pmod{N}) \leq \frac{1}{2^{m-1}}$. Para cada conjunto de $x_j \in \mathbb{Z}_{p_j^{\alpha_j}}^*$ escolhido de maneira aleatória, uniforme e com os j dois a dois distintos o corolário 4 nos dá um único $x \in \mathbb{Z}_N^*$ aleatório. Podemos ao invés de escolher o elemento x , escolher os elementos x_j em $\mathbb{Z}_{p_j^{\alpha_j}}^*$ e trabalhar com o x associado à coleção x_j . Lembre-se que x satisfaz $x \equiv x_j \pmod{p_j^{\alpha_j}}$. Seja r_j a ordem de x_j em $\mathbb{Z}_{p_j^{\alpha_j}}^*$. Seja 2^{d_j} a maior potência de 2 que divide r_j e 2^d a maior potência de 2 que divide r , em que

r é a ordem de x em \mathbb{Z}_N^* . Mostraremos que para se ter r ímpar ou $x^{r/2} \equiv -1 \pmod{N}$ é necessário que d_j tome o mesmo valor para todos os valores de j . Sabemos que $x^r \equiv x_j^r \pmod{p_j^{\alpha_j}}$, mas $x^r \equiv 1 \pmod{N}$, então, $x^r \equiv 1 \pmod{p_j^{\alpha_j}}$. Logo, $x_j^r \equiv 1 \pmod{p_j^{\alpha_j}}$. Com isso, temos que $r_j | r \forall j$. Se r é ímpar, como $r_j | r$, concluímos que r_j é ímpar. Portanto, $d_j = 0, \forall j = 1, \dots, m$. Se r é par e $x^{r/2} \equiv -1 \pmod{N}$, temos que $x^{r/2} \equiv -1 \pmod{p_j^{\alpha_j}}$, como $x^{r/2} \equiv x_j^{r/2} \pmod{p_j}$, logo $x_j^{r/2} \equiv -1 \pmod{p_j}$. Portanto, r_j não divide $\frac{r}{2}$. Pelo fato que $r_j | r$ concluímos que $d_j = d \forall j$. No caso de r ser ímpar temos que r_j tem que ser ímpar $\forall j = 1, \dots, m$. Mas, pelo lema 6, a probabilidade deste evento é $\frac{1}{2^m}$. No caso de r ser par e $x^{r/2} \equiv -1 \pmod{N}$ todos os d_j têm que ser iguais, além disso, todos os r_j têm que ser par. A probabilidade de todos r_j serem par é $\frac{1}{2^m}$. Logo,

$$P(r \text{ ser par e } x^{r/2} \equiv -1 \pmod{N}) < \frac{1}{2^m}.$$

Portanto, $P(r \text{ ser ímpar ou } x^{r/2} \equiv -1 \pmod{N}) \leq \frac{1}{2^{m-1}}$. ■

Pela demonstração do teorema 14 sabemos que para a ordem r de um dado $x \in \mathbb{Z}_N^*$ ser par e $x^{\frac{r}{2}} \equiv -1 \pmod{N}$ todos os d_j têm que ser iguais. Logo, em todo \mathbb{Z}_N^* existem x tais que a ordem satisfaz r é par e $x^{\frac{r}{2}} \not\equiv -1 \pmod{N}$. Para isto basta escolhermos alguns x_j tais que r_j seja ímpar e outros tais que r_j seja par.

Os passos abaixo nos dão um algoritmo probabilístico eficiente para encontrar um fator de N , esse algoritmo é uma aplicação do algoritmo de Shor:

Etapa 1: Se N é par, retorne o fator 2, caso contrário faça a Etapa 2.

Etapa 2- Use o algoritmo apresentado na seção 1.3 para determinar se $N = a^b$ para inteiros $a \geq 1$ e $b \geq 2$ e se for o caso retorne o fator a , caso contrário passe a Etapa 3.

Etapa 3- Escolha uniformemente x no intervalo de 1 a $N-1$. Se $\text{mdc}(x, N) > 1$, retorne-o, caso contrário faça a etapa 4.

Etapa 4- Use o algoritmo de Shor para encontrar a ordem r de $x \pmod{N}$.

Etapa 5- Se r for par e $x^{\frac{r}{2}} \not\equiv -1 \pmod{N}$, calcule $\text{mdc}(x^{\frac{r}{2}} - 1, N)$ e $\text{mdc}(x^{\frac{r}{2}} + 1, N)$. Verifique qual deles é um fator não trivial e retorne-o. Caso contrário o algoritmo falha.

Analisaremos a complexidade desse algoritmo. A etapa 1 é $O(1)$, pois é um comando de leitura. O algoritmo vai ler o último dígito e ver se ele é 0 ou 1. A etapa 2 pela seção 1.3 é $O(L^4)$. As etapas 3 e 4 pelas seções 1.3, 4.2, respectivamente são $O(L^3)$. A etapa 5, realiza 3 comandos, o primeiro é de leitura, sendo $O(1)$. O segundo é calcular uma exponencial modular que é $O(L^3)$ e por último calcular o máximo divisor comum que é $O(L^3)$. Nesse caso a complexidade desse loop é a do comando de maior complexidade sendo $O(L^3)$. A complexidade do algoritmo é o da etapa de maior complexidade, portanto, $O(L^4)$.

O algoritmo acima é eficiente, todas as suas etapas podem ser rodadas em um computador clássico, exceto a etapa 4 que é o algoritmo de Shor, pois essa etapa é um algoritmo quântico. No momento, às vezes, não damos a esse algoritmo o status que ele merece porque os computadores quânticos existentes são de pequeno porte. Mas, com a evolução dos computadores quânticos quando eles forem capazes de trabalhar com o mesmo tamanho de números que os computadores clássicos atuais trabalham poderemos ver a sua importância real e ele deixará de ser apenas um algoritmo teórico.

A etapa 4 encontra com alta probabilidade a ordem de um elemento $x \in \mathbb{Z}_N^*$ escolhido uniformemente. Na Etapa 5, o teorema 14 garante que r é par e $x^{\frac{r}{2}} \not\equiv -1 \pmod{N}$ com probabilidade maior que $\frac{1}{2}$. Se isso acontece, o teorema 10 garante que temos um fator não-trivial de N . Se r não satisfaz as hipóteses do teorema 14 o algoritmo falha. Mas, se isso ocorrer temos a possibilidade de rodar o algoritmo novamente, quantas vezes precisarmos até que ele encontre um x na Etapa 3 que satisfaça o teorema 14. Pelo comentário após o teorema 14 sabemos que sempre existe x que satisfaz a etapa 5.

Faremos um exemplo para ver como funciona. Usaremos o algoritmo para fatorar o 21. O primeiro passo é ver se 21 é par, para isso basta escrever 21 na base 2 e olharmos se o último dígito é 0. Já sabemos que 21 não é par. Então, passaremos à próxima etapa, utilizaremos o algoritmo da seção 1.3 para verificar se 21 é da forma a^b . Sabemos que isso não acontece. Logo,

iremos para a etapa 3, escolher um x aleatório em $[1; 20]$. Suponha que o x escolhido seja o 2. Calculando $\text{mdc}(2, 21) = 1$, passaremos ao algoritmo de Shor, para encontrar a ordem de 2. Pela seção 4.3, sabemos que r é 6. Neste caso, r é par e $2^{\frac{r}{2}} = 2^3 \equiv 8 \not\equiv -1 \pmod{21}$. Portanto, o r encontrado satisfaz as hipóteses da etapa 5. Calculando $\text{mdc}(2^3 - 1, 21) = \text{mdc}(7, 21) = 7$ e $\text{mdc}(2^3 + 1, 21) = 3$. Aqui os dois resultados são fatores de 21. Isso, não ocorre sempre, mas basta obtermos um dos fatores.

O exemplo para fatorar 21 pode ter parecido sem muitos atrativos, pois basta sabermos a tabuada para fatorá-lo. Por isso, faremos um outro exemplo, menos trivial. Iremos fatorar 1927, você ainda pode estar achando que encontrar os fatores primos desse número é uma tarefa fácil, portanto não precisaríamos de um algoritmo quântico. Não escolhemos números maiores, pois as contas a mão começam a tornar-se impraticáveis. Utilizando o algoritmo para encontrar fatores, primeiro verificaremos se 1927 é par, como a resposta é não, faremos a etapa 2, determinar se existem $a \geq 1$ e $b \geq 2$ tais que $1927 = a^b$, novamente a resposta é não. O próximo passo é escolher $x \in [1, N - 1]$, digamos que a escolha foi 4. Como $\text{mdc}(1927, 4) = 1$, temos que fazer a próxima etapa, que é a parte quântica do algoritmo. O circuito da figura 5.1 descreve os passos para descobrirmos a ordem do elemento $4 \in \mathbb{Z}_{1927}^*$. A quantidade de q-bits no primeiro registrador iniciados no estado $|0\rangle$ é $t = \lfloor \log_2 1927^2 \rfloor + 1 = 22$ e a quantidade de q-bits do segundo registrador é $L = \lfloor \log_2 1927 + 1 \rfloor = 11$.

Os estados descritos no circuito da figura 5.1 são:

$$\begin{aligned} |\psi_0\rangle &= \underbrace{|0 \dots 0\rangle}_{22} |1\rangle, \\ |\psi_1\rangle &= \frac{1}{2^{11}} \sum_{j=0}^{2^{22}-1} |j\rangle |1\rangle, \\ |\psi_2\rangle &= \frac{1}{2^{11}} \sum_{j=0}^{2^{22}-1} |j\rangle |4^j\rangle, \end{aligned}$$

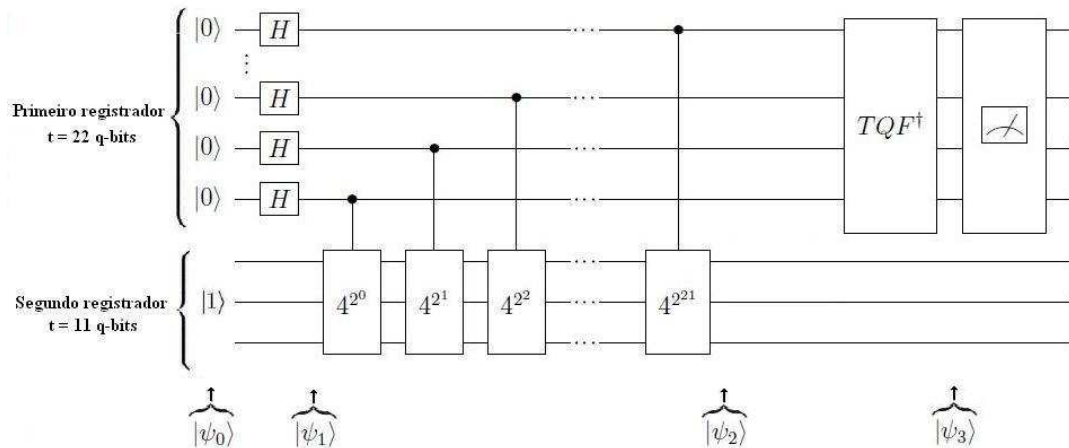


Figura 5.1: Circuito para calcular a ordem do elemento 4.

aplicando a transformada de Fourier inversa temos,

$$\frac{1}{2^{22}} \sum_{k,j} e^{-\frac{2\pi ijk}{2^{22}}} |k\rangle |4^j\rangle.$$

Ao realizarmos a medição no primeiro registrador, suponha que obtemos $k = 18236$. Fazendo $\frac{18236}{4194304}$ e aplicando o algoritmo de frações contínuas, obtemos,

$$\frac{18236}{4194304} = \frac{1}{230 + \frac{4024}{18236}}.$$

Deveríamos continuar e calcular todos os convergentes, pois na prática é isto que o computador faria. Depois ele iria selecionar os denominadores para testar se algum deles é a ordem. Não calcularemos todos os convergentes, pois será um trabalho inútil já que $4^{230} \equiv 1 \pmod{1927}$. Passando à etapa 5, temos que verificar qual é a solução de $4^{115} \equiv y \pmod{1927}$. Calculando temos, $4^{115} \equiv 1270 \pmod{1927}$, portanto $4^{115} \not\equiv -1 \pmod{1927}$. Agora é só encontrar $\text{mdc}(1270 + 1, 1927) = 1$ e $\text{mdc}(1270 - 1, 1927) = 47$. Logo, encontramos um fator de 1927 para obter outro basta dividir 1927 por 47 obtendo 41. Neste caso, como $1927 = 41 \cdot 47$, 41 e 47 são primos temos a fatoração de 1927.

O algoritmo apresentado para encontrar um fator de N pode ser modifi-

cado para retornar a fatoração completa de N . Neste trabalho não será feito isto, porque a curiosidade que nos moveu para estudar o algoritmo de Shor foi ver que sempre é possível encontrar um fator de um número natural composto N . Pois, na criptografia RSA N é o produto de dois números primos, basta encontrarmos um fator de N para conseguirmos fatorá-lo completamente. Fatorando N conseguimos quebrar o protocolo de segurança deste método criptográfico. Caso o leitor tenha ficado curioso para descobrir como é feita a criptografia RSA basta ver [2].

Apêndice A

Algoritmo quântico para calcular $x^j y \pmod N$

No capítulo 4 quando apresentamos o algoritmo da busca de ordem, vimos que era necessário calcular $x^j y \pmod N$ para $0 \leq j \leq 2^t - 1$. No momento nosso objetivo é apresentar um algoritmo que realiza essas operações e calcular o seu custo computacional. A ação da porta x^j pode ser vista como:

$$\begin{aligned} |j\rangle |y\rangle &\mapsto |j\rangle |x^j y \pmod N\rangle \\ &= |j\rangle |x^{(j_t 2^{t-1} + \dots + j_1 2^0)} y \pmod N\rangle \\ &= |j\rangle |x^{j_t 2^{t-1}} \cdot \dots \cdot x^{j_1 2^0} y \pmod N\rangle. \end{aligned}$$

O algoritmo pode ser descrito como:

Etapa 1: calcular $x^2 \pmod N$ a partir de $x \pmod N$, depois obter $x^4 \pmod N$, até $x^{2^{t-1}} \pmod N$.

Etapa 2: calcular $z = (x^{j_t 2^{t-1}} \pmod N) \dots (x^{j_1 2^0} \pmod N)$.

Etapa 3: fazer $z \cdot y \pmod N$.

A Etapa 1 faz $t - 1$ operações de quadramento, como $t \leq 2L + 1$, e cada operação de elevar ao quadrado custa $O(L^2)$, essa etapa é $O(L^3)$. Na Etapa 2 temos $t - 1$ multiplicações modulares custando $O(L^2)$, logo essa etapa é $O(L^3)$. A Etapa 3 possui apenas uma multiplicação que custa $O(L^2)$. O custo do algoritmo é o da etapa mais cara, sendo $O(L^3)$.

Além do algoritmo ser eficiente, temos que ter uma maneira de implementá-lo em um computador quântico. Essa preocupação deve-se ao fato que no computador quântico todas as operações são reversíveis, enquanto na computação clássica existem operações irreversíveis. Como veremos isso não será um problema, a solução virá do seguinte fato: todo algoritmo clássico possui um circuito clássico que o implementa e é possível construí-lo utilizando apenas portas NOT e AND.

A NOT é uma porta de um bit, sua ação é negar a entrada, podemos descrevê-la como:

$$\begin{aligned} 0 &\mapsto 1 \\ 1 &\mapsto 0. \end{aligned}$$

Claramente a porta NOT é reversível. Na computação quântica a matriz $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ atuando nos vetores da base de \mathbb{C}^2 produz a ação da porta NOT clássica. Como X é unitária ela pode ser implementada em um computador quântico. A porta X é conhecida como o NOT quântico.

A porta *AND* possui dois bits de entrada e um de saída. Sua ação é dada por:

$$\begin{aligned} 00 &\mapsto 0 \\ 01 &\mapsto 0 \\ 10 &\mapsto 0 \\ 11 &\mapsto 1. \end{aligned}$$

Essa porta é irreversível pois se a saída é 0, não sabemos se a entrada é 00, 01, 10. Temos que ter uma maneira de implementar a ação da porta AND de uma forma reversível. Esse problema será solucionado com a porta Toffoli, que pode ser pensada como uma CNOT, usando dois bits de controle no lugar de um. O circuito (A.1) é o da porta Toffoli.

$$\begin{array}{ccc} |a\rangle & \text{---} \bullet & |a\rangle \\ |b\rangle & \text{---} \bullet & |b\rangle \\ |c\rangle & \text{---} \oplus & |a \cdot b + c \pmod{2}\rangle \end{array} \quad (\text{A.1})$$

Essa porta realiza a seguinte tarefa: o valor do bit inferior é invertido se a e b valem 1. Caso contrário, nada acontece. Sua ação nos estados da base computacional é dada por:

$$\begin{array}{ll} 000 \mapsto 000 & 001 \mapsto 001 \\ 010 \mapsto 010 & 011 \mapsto 011 \\ 100 \mapsto 100 & 101 \mapsto 101 \\ 110 \mapsto 111 & 111 \mapsto 110 \end{array}$$

Se $c = 0$ temos:

$$\begin{array}{ll} 000 \mapsto 000 & 010 \mapsto 010 \\ 100 \mapsto 100 & 110 \mapsto 111 \end{array}$$

Observe que o q-bit $|c\rangle$ retorna os valores da tabela verdade da porta AND. Porém existem os dois primeiros q-bits, não podemos apagá-los porque são eles que garantem a reversibilidade. Eles serão considerados o lixo da computação. Se existe um circuito clássico que calcula uma função $f(x)$. Agora podemos implementá-lo em um computador quântico, basta substituir as portas AND pela porta Toffoli com $c = 0$. O único problema é que nessas mudanças na entrada teremos alguns q-bits extras chamados de q-bits auxiliares e na saída teremos o lixo. Os q-bits auxiliares podemos supor que eles estão no estado predefinido $|0\rangle$. E sempre que necessário usamos a porta NOT para mudar seu estado. Descreveremos a ação do circuito como:

$$(x, 0) \mapsto (f(x), g(x)),$$

onde $g(x)$ é o lixo no final da computação. Se o lixo depender de x pode ser que quando fizermos uma medição para descobrir qual o valor de $f(x)$ ele interfira. Por isso, precisamos que ele esteja em um estado bem definido. Mostraremos que é possível deixar o lixo no estado $|0\rangle$. Para isso, aumentaremos um registro no circuito. A quantidade de q-bits nesse registro é a mesma necessária para armazenar $f(x)$. Esses q-bits no início estarão todos no estado $|0\rangle$, denotaremos esse registro extra por 0.

Assim, a entrada no circuito passa a ser descrita por $(x, 0, 0)$, a necessidade desse espaço extra será explicado posteriormente. Após a atuação das

portas que calcula f , o estado do circuito será descrito por $(f(x), g(x), 0)$. O próximo passo é usar portas CNOTs para copiar o conteúdo de $f(x)$ para o registro 0. Note que essa operação é feita q-bit a q-bit. Se for necessário n q-bits para armazenar o valor de $f(x)$, também será preciso n portas CNOTs. No final teremos $(f(x), g(x), f(x))$.

O próximo estágio é passar esse estado pelo circuito que calcula $f(x)$ invertido. Após essa operação teremos $(x, 0, f(x))$. O último registro não é afetado, pois no cálculo de $f(x)$ ele não foi utilizado. Essa técnica usada no final para inverter o circuito é conhecida como decomputação. Com essas modificações a saída da computação está em um estado bem definido.

Analisaremos o custo de um circuito irreversível implementado em computação reversível. Teremos a mesma quantidade de portas do circuito original multiplicado por 2 por causa da decomputação, adicionado das portas CNOT e NOT, as quais possuem uma quantidade linear ao número de q-bits do circuito. Portanto, esse processo não aumenta a complexidade do algoritmo.

Referências Bibliográficas

- [1] Avritzer, D.; Bueno, H. P.; Faria, M. C.; Fernandes, A. M. V.; Ferreira, M. C. C.; Soares, E. F. Fundamentos de Álgebra. UFMG, 2005.
- [2] Coutinho, S. C. Números Inteiros e Criptografia RSA. IMPA, 2003.
- [3] Nielsen, M. A. e Chuang, I. L. Computação Quântica e Informação Quântica. Artmed Editora S. A., 2005.
- [4] Bueno, P. H. Álgebra Linear Um segundo Curso. SBM, 2006.
- [5] Brochero, F. B.; Moreira, C. G. T. A.; Saldanha, N. C.; Tengan, E. Teoria dos Números. www.icmc.usp.br/~etengan/livro.pdf, visitado em 20/04/2010.
- [6] Santos, J. P. O. Introdução à Teoria dos Números. IMPA, 2005.
- [7] Lang, S. Linear Algebra. Addison-Wesley, 1968.
- [8] Shor, W. P. Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. arxiv:quant-ph/9508027v2 25 jan 1996.
- [9] Portugal, R.; Lavor, C. C.; Carvalho, L. M.; Maculan, N. Uma Introdução à Computação Quântica. Sociedade Brasileira de Matemática Aplicada e Computacional, 2004.
- [10] Press, W. H. Numerical Recipes in Fortran. Press Syndicate of the University of Cambridge, 1994.

- [11] Ziviani, N. *Projetos de Algoritmos e Estrutura de Dados*. Campinas, 1986.
- [12] Ekert, A.; Josza R. Quantum Computation and Shor's Factoring Algorithm. *Rev. Mod. Phys.*, Vol. 68 Número 3, Julho 1996.
- [13] Vandersypen, L. M. K. et al Experimental Realization of Shor's Quantum Factoring Algorithm Using Nuclear Magnetic Resonance. *Nature* Vol 414, 20/27 December 2001, 883-887.
- [14] Wootters, W. K.; Zurek, W. H. A Single Quantum Cannot Be Cloned. *Nature* Vol 299, 1985, 802-803.