DSC/CEEI/UFCG

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática

DSC

Departamento de Sistemas e Computação

Av. Aprígio Veloso, 882 — Bodocongó
Caixa Postal 10.106
58.109-970 — Campina Grande — PB — Brasil
Fone: 3310-1122 — Fax: 310-1273
e_mail: dsc@dsc.ufcg.edu.br;
http://www.dsc.ufcg.edu.br

Relatório Técnico

N° DSC 001/2008

ALGORITMOS QUÂNTICOS

Elloá B. Guedes da Costa
Bernardo Lula Júnior

UFCG/CEEI/DSC/IQUANTA
elloa@dsc.ufcg.edu.br
lula@dsc.ufcg.edu.br

30 páginas

Março 2008

Algoritmos Quânticos

Elloá B. Guedes da Costa, Bernardo Lula Júnior

Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Instituto de Estudos em Computação e Informação Quânticas
elloa@dsc.ufcg.edu.br,lula@dsc.ufcg.edu.br

RESUMO

Com a miniaturização do hardware computacional, a Lei de Moore prevê que em 2020 estaremos armazenando um bit por átomo. Para que tal previsão se confirme, é necessário entender como tirar proveito das propriedades quânticas presentes nestas partículas. Decorrente desta necessidade, em 1985, Deutsch propôs uma Máquina de Turing Quântica, a qual utiliza fundamentos da mecânica quântica e é análoga à Máquina de Turing Clássica. Esta proposição levou à concepção moderna do computador quântico. Associado ao computador quântico temos algoritmos de mesma natureza. O grupo fundamental de problemas estudado na computação quântica são os problemas do subgrupo oculto, pois muitos problemas são redutíveis à eles e também são ferramentas para mostrar como algoritmos quânticos podem resolver problemas de forma mais econômica, em termos de consulta ao oráculo, que os melhores algoritmos clássicos existentes. O presente relatório técnico visa mostrar tais problemas e como resolvê-los, utilizando a computação clássica e quântica, com maior foco nesta última, utilizando a linguagem de circuitos. Este relatório é baseado nas "Quintas Quânticas", seminários semanais realizados no IQuanta - Instituto de Estudos de Computação e Informação Quânticas na Universidade Federal de Campina Grande.

Palavras-chave: Computação Quântica, Algoritmos Quânticos, Circuitos Quânticos

ABSTRACT

This report intends to be an initial reference for Computer Science's students about Quantum Algorithms, introducing the subject in a simple, clear and objective form. For this, it contains a brief introduction and motivation, showing its importance and relevance. Then, will be shown 7 problems that can be solved by quantum algorithms in a more economic way than their best classical solution, also shown here. As being an extensive theme, enough references are given so that the reader can go deeper and learn more about quantum algorithms.

Keywords: Quantum Computing, Quantum Algorithms, Quantum Circuits

Contents

1 Introdução

A tese forte de Church-Turing afirma que "Qualquer processo algorítmico pode ser simulado de forma eficiente usando uma máquina de Turing probabilística". Mas, num momento futuro, poderia aparecer algum outro modelo de computação capaz de resolver problemas mais eficientemente do que o modelo Turing? Essa pergunta motivou David Deutsch, em 1985, a se questionar se as leis da físicas poderiam ser usadas para derivar uma versão ainda mais forte desta tese.

Como as leis físicas são em última análise quânticas, Deutsch foi naturalmente levado a considerar tais aparatos com base nos princípios da mecânica quântica. Esses aparatos, análogos quânticos das máquinas de Turing, levaram à concepção moderna de um computador quântico [NL05].

Inerentes ao computador quântico, podemos também visualizar algoritmos de mesma natureza. Neste trabalho iremos apresentar os algoritmos quânticos que resolvem os problemas do subgrupo oculto. Para tanto, iremos inicialmente caracterizar este subgrupo e os problemas associados. Para cada problema, iremos apresentar a sua solução clássica mais eficiente, o algoritmo quântico que o resolve e um comparativo assintótico, em termos de consultas ao oráculo, entre as duas soluções propostas. Para facilitar a compreensão, utilizaremos a linguagem de circuitos para o entendimento dos algoritmos quânticos.

2 Problemas do Subgrupo Oculto

O problema do subgrupo oculto é enunciado como se segue:

> Seja f uma função de um grupo G gerado finitamente em um conjunto finito X, tal que f é constante nos espaços-quociente de um subgrupo K e distinta em cada um desses quocientes. Dada uma caixa-preta que realiza a transformação unitária $U\left|g\right\rangle\left|h\right\rangle \rightarrow \left|g\right\rangle\left|h \oplus f(g)\right\rangle$, para $g \in G$, $h \in X$, e $\oplus$ uma operação binária apropriada sobre X, encontre o conjunto gerador de K [NL05].

São exemplos de problemas do subgrupo oculto: o problema de Deutsch, a busca de período, o problema de Simon, a busca da ordem e o logaritmo discreto. Tais problemas serão abordados aqui neste trabalho.

Uma das principais razões que motiva o estudo dos problemas do subgrupo oculto (PSO) é o fato de muitos outros problemas serem redutíveis à este conjunto de problemas e a inexistência de algoritmos clássicos eficientes que os resolvam [Lom04]. A razão principal para esta ineficiência é que classicamente temos que executar a avaliação da função f seqüencialmente. Quanticamente existe a possibilidade de avaliação simultânea através do paralelismo quântico [Por07].

3 Problema de Deutsch

O problema de Deutsch-Jozsa é uma generalização do problema de Deutsch e é descrito como segue:

Seja $f : \{0,1\} \rightarrow \{0,1\}$ uma função booleana desconhecida. Observemos as configurações possíveis que f pode ter:

Função	$x = 0$	$x = 1$
f_0	0	0
f_1	0	1
f_2	1	0
f_3	1	1

Dizemos que f_0 e f_3 são funções constantes pois $f_i(0) = f_i(1)$, ou seja, aplicada na base binária produzem imagens iquais ainda que seus parâmetros de entrada sejam diferentes. Quando isso não acontece, temos que $f_i(0) \neq f_i(1)$ e a esse conjunto de funções denominamos balanceadas. Observe que f_1 e f_2 são classificadas como sendo deste último tipo citado.

Uma forma mais simples de enxergar esta classificação de função binária é usando a soma módulo 2 com os valores de $f(0)$ e $f(1)$:

$$f(0) \oplus f(1) = \begin{cases} 0, & \text{se } f \text{ é constante} \\ 1, & \text{se } f \text{ é balanceada} \end{cases}$$

onde $\oplus$ é o símbolo da soma módulo 2. A soma módulo 2 é o resto da divisão por 2 da soma de dois números. Os resultados possíveis para ela são: $0 \oplus 0 = 1 \oplus 1 = 0$ e $1 \oplus 0 = 0 \oplus 1 = 1$ [dLJ05].

3.1 Solução Clássica

Na computação clássica, essa avaliação que leva à classificação da função é feita da seguinte forma:

1. Obter $f(0)$
2. Obter $f(1)$
3. Comparar os valores a fim de obter a classificação

O custo desse algoritmo para um caso mais geral com n entradas é de $\Theta(n)$. Para entender este custo, basta perceber que é necessário percorrer uma vez todos os n valores.

Em 1985, Deutsch, utilizando operações quânticas, mostrou que é possível determinar se uma função é constante ou balanceada através de uma única consulta a um oráculo [Deu85], fazendo então com que o custo dessa operação seja $O(1)$. Este oráculo por sua vez, faz uso de superposição e da interferência quânticas para determinar o resultado.

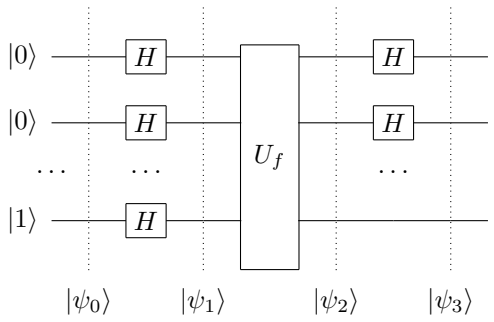
3.2 Algoritmo de Deutsch-Jozsa

O algoritmo de Deutsch, proposto em 1985, trata da classificação de funções binárias de aridade 1. A extensão proposta por Deutsch-Jozsa lida com funções de aridade qualquer, ou seja, aquelas definidas por $f : \{0,1\}^n \to \{0,1\}$ e tem por objetivo o mesmo do algoritmo de Deutsch, determinar se uma função binária é constante ou balanceada com um gasto mínimo.O problema de Deutsch-Jozsa é, na verdade, uma generalização do problema Deutsch [NL05].

A demonstração a seguir foi organizada comtemplando informações de [CEMM96], [PCG06] e [Joz96].

3.2.1 Circuito Resolvedor do Algoritmo de Deutsch-Jozsa

Para entendermos como o algoritmo do nosso estudo alcança esse objetivo, vamos partir do circuito que o resolve proposto pelos seus idealizadores:



Circuito Resolvedor do Algoritmo de Deutsch-Jozsa

Temos $|0\rangle^{\otimes n}$ e $|1\rangle$ como entradas do circuito. Logo, $|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$ é o que obtemos da primeira interceptação antes da entrada no circuito e que podemos chamar de estado inicial.

Apliquemos então à entrada portas de Hadamard:

$$|\psi_1\rangle = (H^{\otimes n} \otimes H)(|0\rangle^{\otimes n} |1\rangle)$$

Temos que:

$$H\left|1\right\rangle = \left(\frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}\right)$$

e que

$$H^{\otimes n}\left|0\right\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}\left|x\right\rangle.$$

Observe que $\sum_{x=0}^{2^n-1}\left|x\right\rangle$ utiliza a base decimal para representar o qubit. Essa notação será usada ao longo deste trabalho.

Assim:

$$\left|\psi_1\right\rangle = \Big(\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}\left|x\right\rangle\Big)\otimes\Big(\frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}\Big)$$

$$\left|\psi_1\right\rangle = \frac{1}{2^{\frac{n}{2}}\sqrt{2}}\sum_{x=0}^{2^n-1}\big(\left|x\right\rangle\left|0\right\rangle - \left|x\right\rangle\left|1\right\rangle\big)$$

Agora iremos aplicar a porta U_f aos qubits em $\left|\psi_1\right\rangle$. A atuação de U_f é da forma:

$$U_f\left|x\right\rangle\left|y\right\rangle \rightarrow \left|x\right\rangle\left|y\oplus f(x)\right\rangle$$

Logo, a aplicação do oráculo ao estado $\left|\psi_1\right\rangle$ é como se segue:

$$\left|\psi_2\right\rangle = U_f\left|\psi_1\right\rangle = U_f\left\{\frac{1}{2^{n/2}\sqrt{2}}\Big[\sum_{x=0}^{2^n-1}\big(\left|x\right\rangle\left|0\right\rangle - \left|x\right\rangle\left|1\right\rangle\big)\Big]\right\}$$

Como U_f é linear, temos:

$$\left|\psi_2\right\rangle = \frac{1}{2^{n/2}\sqrt{2}}\left\{U_f\Big[\sum_{x=0}^{2^n-1}\big(\left|x\right\rangle\left|0\right\rangle\big)\Big] - U_f\Big[\sum_{y=0}^{2^n-1}\big(\left|y\right\rangle\left|1\right\rangle\big)\Big]\right\}$$

$$\left|\psi_2\right\rangle = \frac{1}{2^{n/2}\sqrt{2}}\left[\big(\sum_{x=0}^{2^n-1}\big(\left|x\right\rangle\left|f(x)\right\rangle\big) - \Big[\sum_{y=0}^{2^n-1}\big(\left|y\right\rangle\left|\overline{f(y)}\right\rangle\big)\right]\right]$$

Note que $f(y)$ aparece invertido, isso se dá porque no alvo (2° qubit), temos:

$$1\oplus f(y) = \begin{cases} Se\ \ f(y)=0 \Rightarrow 1\oplus f(y) = 1\oplus 0 = 1 = \overline{f(y)} \\ Se\ \ f(y)=1 \Rightarrow 1\oplus f(y) = 1\oplus 1 = 0 = \overline{f(y)} \end{cases}$$

Vamos agora fazer uma análise mais aprofundada sobre o estado $\left|\psi_2\right\rangle$ e como dele podemos chegar ao que se propõe o algoritmo de Deutsch-Jozsa.

Analisemos separadamente as possíveis classificações para f.

1. f é constante

   (a) $f(x)=0$, $\forall x$ em $\{0,1\}^n$

$$\left|\psi_2\right\rangle = \frac{1}{2^{n/2}\sqrt{2}}\big(\sum_{x=0}^{2^n-1}\left|x\right\rangle\left|0\right\rangle - \sum_{y=0}^{2^n-1}\left|y\right\rangle\left|\overline{0}\right\rangle\big)$$

   Podemos unificar os somatórios:

$$\left|\psi_2\right\rangle = \frac{1}{2^{n/2}\sqrt{2}}\Big[\sum_{z=0}^{2^n-1}\big(\left|0\right\rangle - \left|1\right\rangle\big)\left|z\right\rangle\Big]$$

   Trazendo $\sqrt{2}$ para o somatório:

$$\left|\psi_2\right\rangle = \frac{1}{2^{n/2}}\left[\sum_{z=0}^{2^n-1}\Big(\frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}\Big)\left|z\right\rangle\right]$$

(b) $f(x) = 1, \forall x$ em $\{0,1\}^n$

$$|\psi_2\rangle = \frac{1}{2^{n/2}\sqrt{2}}\Big(\sum_{x=0}^{2^n-1}|x\rangle\,|1\rangle - \sum_{y=0}^{2^n-1}|y\rangle\,|\overline{1}\rangle\Big)$$

De modo análogo ao subcaso (a), unifiquemos o somatório e levemos $\sqrt{2}$ para este novo somatório:

$$|\psi_2\rangle = \frac{1}{2^{n/2}}\Big[\sum_{z=0}^{2^n-1}\Big(\frac{|1\rangle - |0\rangle}{\sqrt{2}}\Big)\,|z\rangle\Big]$$

Coloquemos -1 em evidência:

$$|\psi_2\rangle = (-1)\frac{1}{2^{n/2}}\left[\sum_{z=0}^{2^n-1}\Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)\,|z\rangle\right]$$

Observemos agora os resultados dos subcasos (a) e (b). Eles se diferenciam apenas pelo (-1). Vamos então escrever $|\psi_2\rangle$ em função do valor $f(x)$ para encontrarmos uma generalização para as funções constantes.

Temos que:

$$(-1)^0 = 1 \Rightarrow (-1)^{f(z)}, \forall z \in \{0,1\}^n | f(z) = 0$$
$$(-1)^1 = -1 \Rightarrow (-1)^{f(z)}, \forall z \in \{0,1\}^n | f(z) = 1$$

Logo,

$$|\psi_2\rangle = \frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}(-1)^{f(z)}\,|z\rangle\,\Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

2. f é balanceada

$$|\psi_2\rangle = \frac{1}{2^{n/2}\sqrt{2}}\Big(\sum_{x=0}^{2^n-1}|x\rangle\,|f(x)\rangle - \sum_{y=0}^{2^n-1}|y\rangle\,\Big|\overline{f(y)}\Big\rangle\Big)$$

Vamos agora expandir os somatórios:

$$|\psi_2\rangle = \frac{1}{2^{n/2}\sqrt{2}}\Bigg[|0\rangle\,|f(0)\rangle + |1\rangle\,|f(1)\rangle + \ldots + |2^n-1\rangle\,|f(2^n-1)\rangle - \Big(|0\rangle\,\Big|\overline{f(0)}\Big\rangle + + |1\rangle\,\Big|\overline{f(1)}\Big\rangle + \ldots + \\ + |2^n-1\rangle\,\Big|\overline{f(2^n-1)}\Big\rangle\Big)\Bigg]$$

Organizando os elementos acima segundo um somatório em z:

$$|\psi_2\rangle = \frac{1}{2^{n/2}\sqrt{2}}\sum_{z=0}^{2^n-1}|z\rangle\,\Big(|f(z)\rangle - \Big|\overline{f(z)}\Big\rangle\Big)$$

Colocando $\sqrt{2}$ para dentro do somatório:

$$|\psi_2\rangle = \frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}|z\rangle\,\Big(\frac{|f(z)\rangle - \Big|\overline{f(z)}\Big\rangle}{\sqrt{2}}\Big)$$

Numa função balanceada, temos metade dos $f(z) = 0$ e a outra metade $f(z) = 1$. Podemos então escrever:

$$|\psi_2\rangle = \frac{1}{2}\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) + \frac{1}{2}\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}|z\rangle\left(\frac{|1\rangle - |0\rangle}{\sqrt{2}}\right)$$

$$|\psi_2\rangle = \frac{1}{2}\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) + (-1)\frac{1}{2}\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Observe que:

$$f(z) = 0 \Rightarrow \sum_{z=0}^{2^n-1}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$
$$= (-1)^{f(z)}\sum_{z=0}^{2^n-1}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

e que

$$f(z) = 1 \Rightarrow \sum_{z=0}^{2^n-1}(-1)|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$
$$= \sum_{z=0}^{2^n-1}(-1)^{f(z)}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Vamos então reescrever $|\psi_2\rangle$:

$$|\psi_2\rangle = \frac{1}{2}\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}(-1)^{f(z)}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) + \frac{1}{2}\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}(-1)^{f(z)}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Logo,

$$|\psi_2\rangle = \frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}(-1)^{f(z)}|z\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Observe que pudemos generalizar o estado $|\psi_2\rangle$ independentemente de qual é o tipo de função. Isso significa que após a passagem do oráculo ainda não temos informações suficientes para determinar sea função é constante ou balanceada.

Já temos o estado após a aplicação de U_f. Agora devemos seguir o último passo, que é aplicar portas de Hadamard aos n primeiros qubits do estado $|\psi_2\rangle$, resultando $|\psi_3\rangle$:

$$|\psi_3\rangle = (H^{\otimes n} \otimes I)\,|\psi_2\rangle$$

$$= (H^{\otimes n} \otimes I)\left(\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}(-1)^{f(z)}|z\rangle\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$$= H^{\otimes n} \otimes \left(\frac{1}{2^{n/2}}\sum_{z=0}^{2^n-1}(-1)^{f(z)}|z\rangle\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Algumas pequenas explicações devem ser dadas ao passo acima. Primeiro quanto à aplicação de $(H^{\otimes n} \otimes I)$. Isso se deve ao fato de $|\psi_2\rangle$ ter uma dimensão maior do que $H^{\otimes n}$, então utilizamos a Matriz Identidade a fim de fazer o produto tensorial compatível. Ainda quanto a isso, podemos enxergar um produto tensorial da seguinte forma: $(A \otimes B) = A \otimes I + I \otimes B$. Derivando para a situação que temos, vale a seguinte igualdade: $(U^{\otimes n} \otimes I)(|a\rangle\,|b\rangle) = U^{\otimes n}\,|a\rangle \otimes I\,|b\rangle$.

Aplicando $H^{\otimes n}$ e resolvendo o somatório:

$$|\psi_3\rangle = \frac{1}{2^{n/2}}\Big(H^(-1)^{f(0)}\,|0\rangle + \ldots + H^(-1)^{f(2^n-1)}\,|2^n-1\rangle\Big) \otimes \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

Aproximando H ao vetor a que ele se aplica:

$$|\psi_3\rangle = \frac{1}{2^{n/2}}\Big((-1)^{f(0)}H\,|0\rangle + \ldots + (-1)^{f(2^n-1)}H\,|2^n-1\rangle\Big) \otimes \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

Observe que:

$$H^{\otimes n}\,|x_1,\ldots,x_n\rangle = \frac{\sum_{z_1,\ldots,z_n}(-1)^{x_1z_1+\ldots+x_nz_n}\,|z_1,\ldots,z_n\rangle}{\sqrt{2^n}}$$

O resultado acima pode ser escrito de uma forma mais simplificada, que é a generalização da aplicação de portas de Hadamard à um ket qualquer e está denotada abaixo:

$$H^{\otimes n}\,|x\rangle = \frac{\sum_z(-1)^{xz}\,|z\rangle}{\sqrt{2^n}}$$

Esclarecidos os passos matemáticos a serem utilizados, continuemos na resolução do algoritmo:

$$|\psi_3\rangle = \left[\frac{(-1)^{f(0)}}{2^{n/2}}\sum_{z=0}^{2^n-1}(-1)^{0\cdot z}\,|z\rangle + \ldots + \frac{(-1)^{f(2^n-1)}}{2^{n/2}}\sum_{z=0}^{(2^n-1)\cdot z}|z\rangle\right] \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$$= \frac{1}{2^{n/2}}\frac{1}{2^{n/2}}\left[\sum_{z=0}^{2^n-1}(-1)^{f(0)}(-1)^{0\cdot z}\,|z\rangle + \ldots + \sum_{z=0}^{2^n-1}(-1)^{f(2^n-1)}(-1)^{(2^n-1)\cdot z}\,|z\rangle\right] \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$$= \frac{1}{2^n}\left(\sum_{y=0}^{2^n-1}\sum_{x=0}^{2^n-1}(-1)^{x\cdot y+f(y)}\,|x\rangle\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Denominamos amplitude a_0 do componente $|0\rangle$ do estado do 1º registrador a seguinte expressão:

$$a_0 = \sum_{y=0}^{2^n-1}\frac{(-1)^{f(y)}}{2^n}$$

Então analizando os possíveis valores da amplitude nas configurações de funções que podemos ter:

1. Se f é constante
   (a) Se $f_i(x) = 0, \forall x \in \{0,1\}^n$

$$a_0 = \sum_{y=0}^{2^n-1}\frac{(-1)^{f(y)}}{2^n} = \frac{(-1)^0(2^n)}{2^n} = 1$$

   (b) Se $f_i(x) = 1, \forall x \in \{0,1\}^n$

$$a_0 = \sum_{y=0}^{2^n-1}\frac{(-1)^{f(y)}}{2^n} = \frac{(-1)^1(2^n)}{2^n} = -1$$

2. Se f é balanceada

$$a_0 = \sum_{y=0}^{2^n-1} \frac{(-1)^{f(y)}}{2^n}$$

Mas metade dos $f(y) = 0$ e a outra metade $f(y) = 1$

$$a_o = \frac{1}{2}\frac{(-1)^0(2^n)}{2^n} + \frac{1}{2}\frac{(-1)^1(2^n)}{2^n} \Rightarrow$$
$$a_o = \frac{1}{2}\frac{2^n-1}{2^n} - \frac{1}{2}\frac{2^n-1}{2^n} \Rightarrow$$
$$a_0 = 0$$

Uma medição dos n qubits encontrará todos eles no estado $|0\rangle$ com 100% de certeza. Se f for balanceada, uma medição deve produzir um resultado nulo.

Assim, está provado que, em uma única passagem pelo circuito, o algoritmo de Deutsch-Jozsa resolve o problema em questão [dLJ05].

4 Busca de Período

A busca de período, ou de modo equivalente, o problema de Bernstein-Vazirani, é definir o valor de um certo período a, que é uma cadeia de bits desconhecida. De forma mais completa, temos a seguinte descrição:

> Seja $f_a(x) = a \cdot x$ uma função de n bits inteiros em $\{0, 1\}$, onde a é um inteiro de n bits e $a \cdot x$ é o produto interno módulo 2 bit-a-bit: $a \cdot x = a_0x_0 \oplus a_1x_1 \oplus \ldots \oplus a_nx_n$, onde $\oplus$ indica a adição módulo 2.
>
> Existe uma caixa-preta U_a que leva os n-qubits da entrada em um qubit de saída, ou seja, de $|x\rangle_n |y\rangle_1$ em $|x\rangle_n |y \oplus f(x)\rangle_1$. O problema é determinar o valor de a com o mínimo de consultas ao oráculo [Mer07].

De modo mais formal, porém equivalente, temos a seguinte descrição para o problema:

> Suponha um oráculo que computa uma função f definida da seguinte forma:
>
> $$f : \{0, 1\}^n \to \{0, 1\}$$
>
> onde $f_a(x) = (a_1 \cdot x_1) \oplus \ldots \oplus (a_n \cdot x_n)$, com $a, x \in \{0, 1\}^n$, onde a_i e x_i são os i-ésimos qubits de a e de x. Deseja-se saber o valor de a [Mer03].

4.1 Solução Clássica

A solução clássica para o Problema de Bernstein-Vazirani é feita analisando-se o produto de cada a_ix_i. Para entender como ela funciona é interessante ter em mente os valores dispostos na tabela abaixo, que mostra a multiplicação de dois números binários:

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

Observe que se temos $a \cdot 0 = 0$ não é possível conhecer, com absoluta certeza, qual o valor de a, pois os valores 0 e 1 satisfazem a igualdade. Já na multiplicação por 1, temos que $a \cdot 1 = 0 \quad \Rightarrow a = 0$ senão se $a \cdot 1 = 1 \quad \Rightarrow a = 1$. Partindo então da multiplicação de um número qualquer (na base binária) por 1, através do resultado dessa operação conseguiremos saber qual é o valor do número. Um algoritmo clássico para o

problema de Bernstein-Vazirani é dado abaixo:

BERNSTEIN-VAZIRANI(x)
(1) **Para cada** $x_i \in x$ **faça**
(2) $f_i(x_i)$ = *consultaOraculo*(x_i)
(3) **Se** $f_i(x_i) = 0$ **então**
(4) $a_i = 0$
(5) **Senão** $a_i = 1$
(6) *retorna*(a)

É interessante notar que o processo acima se aplica a um único bit por vez, ou seja, caso o número a ser descoberto possua n bits, teremos que repetir o processo n vezes, uma para cada bit.

Como o processo tem que ser repetido n vezes, onde n é o tamanho de bits do número que se deseja conhecer, podemos concluir que o custo desta tarefa é $O(n)$.

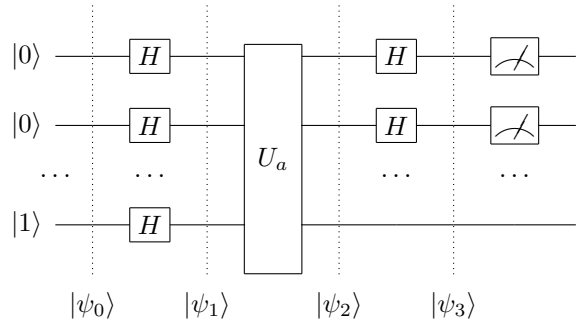
Uma implementação da solução do Problema de Bernstein-Vazirani utilizando a linguagem de programação Java está disponível em *http://dsc.ufcg.edu.br/ ˜elloa*.

4.2 Algoritmo de Bernstein-Vazirani

A demonstração a seguir segue o raciocínio sugerido por [Pre06], [PZ+04] e [BV93].

4.2.1 Circuito para o Algoritmo de Bernstein-Vazirani

Vamos entender o algoritmo a partir do circuito abaixo:



Circuito Resolvedor do algoritmo de Bernstein-Vazirani

Vamos interceptar o circuito em 4 pontos e analisar detalhadamente como se dá a resolução.

O estado inicial, $|\psi_0\rangle$, é apenas a entrada de $|0\rangle$ n vezes e do $|1\rangle$. Podemos representar da seguinte forma:

$$|\psi_0\rangle = |0\rangle^n \otimes |1\rangle$$

O passo seguinte visa criar superposições, para tanto, aplica portas de Hadamard às entradas. Esta aplicação visa deixar todos os vetores de entrada com norma unitária e ortogonais entre si.

Vamos agora analisar como se caracteriza o $|\psi_1\rangle$. Lembremos que $H|1\rangle = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$ e que $H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$. Vale salientar que para o resultado de $H^{\otimes n}|0\rangle^{\otimes n}$ usamos a notação decimal. Com a utilização dessas informações, podemos concluir que:

$$|\psi_1\rangle = (H^{\otimes n} \otimes H) \otimes |\psi_0\rangle \Rightarrow$$
$$|\psi_1\rangle = (H^{\otimes n} \otimes H) \otimes |0\rangle^n \otimes |1\rangle \Rightarrow$$
$$|\psi_1\rangle = \left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \Rightarrow$$

$$|\psi_1\rangle = \frac{1}{2^{\frac{n}{2}}\sqrt{2}}\left(\sum_{x=0}^{2^n-1} |x\rangle\,|0\rangle - |x\rangle\,|1\rangle\right)$$

O próximo passo é a passagem pelo Oráculo U_a. O efeito desse oráculo é calcular a função $f(x) = a \cdot x$, onde x é o parâmetro de entrada e $a \cdot x = \sum_{i=0}^{n} a_i \cdot x_i$. O oráculo U_a tem a seguinte atuação:

$U_a\,|x\rangle\,|y\rangle \rightarrow |x\rangle\,|y \oplus f(x)\rangle$ onde $f(x) = a \cdot x$

Sabendo que U_a é linear, vamos aplicar a $|\psi_1\rangle$ resultando $|\psi_2\rangle$.

$$|\psi_2\rangle = U_a\,|\psi_1\rangle \Rightarrow$$

$$|\psi_2\rangle = U_a\left[\frac{1}{2^{\frac{n}{2}}\sqrt{2}}\sum_{x=0}^{2^n-1}\Big(\,|x\rangle\,|0\rangle - |x\rangle\,|1\rangle\,\Big)\right] \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}\sqrt{2}}\sum_{x=0}^{2^n-1}\Big[U_a\Big(\,|x\rangle\,|0\rangle\,\Big) - U_a\Big(\,|x\rangle\,|1\rangle\,\Big)\Big] \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\Big[\sum_{x=0}^{2^n-1}\Big(\,|x\rangle\,|0 \oplus f(x)\rangle - |x\rangle\,|1 \oplus f(x)\rangle\,\Big)\Big] \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\Big[\sum_{x=0}^{2^n-1}\Big(\,|x\rangle\,|0 \oplus ax\rangle - |x\rangle\,|1 \oplus ax\rangle\,\Big)\Big]$$

Lembremos agora a atuação do operador $\oplus$ (resto da divisão por 2 da soma de dois números):

$$\left.\begin{array}{l} 0 \oplus 0 = 0 \\ 1 \oplus 0 = 1 \end{array}\right\}\ 0 \oplus ax = ax$$

$$\left.\begin{array}{l} 0 \oplus 1 = 1 \\ 1 \oplus 1 = 0 \end{array}\right\}\ 1 \oplus ax = \overline{ax}$$

Portanto,

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\Big[\sum_{x=0}^{2^n-1}\Big(\,|x\rangle\,|ax\rangle - |x\rangle\,|\overline{ax}\rangle\,\Big)\Big]$$

Como f é uma função cuja imagem é a base binária, podemos ter $f(x) = a \cdot x = 0$ ou $f(x) = a \cdot x = 1$.

Retomando a multiplicação na base binária apresentada anteriormente, temos que essa operação pode ser vista como a operação lógica 'AND' da álgebra booleana. Abaixo a tabela verdade para a multiplicação:

$$\begin{array}{l} 0 \cdot 0 = 0 \equiv 0\ AND\ 0 \\ 0 \cdot 1 = 0 \equiv 0\ AND\ 1 \\ 1 \cdot 0 = 0 \equiv 1\ AND\ 0 \\ 1 \cdot 1 = 1 \equiv 1\ AND\ 1 \end{array}$$

Assim, para o $|\psi_2\rangle$, temos as seguintes possibilidades:

1. Se $f(x) = 0$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}|x\rangle \otimes \Big(\frac{|ax\rangle - |\overline{ax}\rangle}{\sqrt{2}}\Big) \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}|x\rangle \otimes \Big(\frac{|0\rangle - |\overline{0}\rangle}{\sqrt{2}}\Big) \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}|x\rangle \otimes \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

2. Se $f(x) = 1$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \Big(\frac{|ax\rangle - |\overline{ax}\rangle}{\sqrt{2}}\Big) \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \Big(\frac{|1\rangle - |\overline{1}\rangle}{\sqrt{2}}\Big) \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \Big(\frac{|1\rangle - |0\rangle}{\sqrt{2}}\Big) \Rightarrow$$

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} (-1)\,|x\rangle \otimes \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

Observe que os casos 1 e 2 se diferenciam apenas pelo (-1). Se levarmos em consideração o valor de $f(x)$ e utilizarmos esse valor associado ao (-1) a fim de generalizarmos o estado $|\psi_2\rangle$, temos:

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}\,|x\rangle \otimes \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

Voltando à definição de f, podemos ver que $f(x) = a \cdot x$ o que significa que podemos substituir o valor de $f(x)$ por $a \cdot x$ na generalização de $|\psi_2\rangle$. Então, como caso geral desse estado, temos:

$$|\psi_2\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} (-1)^{a\cdot x}\,|x\rangle \otimes \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

O passo seguinte no algoritmo é aplicar aos n primeiros componentes do estado $|\psi_2\rangle$ portas de Hadamard, resultando em $|\psi_3\rangle$. Vale lembrar que $(U^{\otimes n} \otimes I)(|a\rangle^{\otimes n}\,|b\rangle) = U^{\otimes n}\,|a\rangle \otimes I\,|b\rangle$, onde I é a matriz identidade.

$$|\psi_3\rangle = (H^{\otimes n} \otimes I)\,|\psi_2\rangle \Rightarrow$$

$$|\psi_3\rangle = H^{\otimes n}\Big(\frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} (1)^{a\cdot x}\,|x\rangle\Big) \otimes I\Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big) \Rightarrow$$

$$|\psi_3\rangle = \frac{1}{2^{\frac{n}{2}}} H^{\otimes n}\Big(\sum_{x=0}^{2^n-1} (1)^{a\cdot x}\,|x\rangle\Big) \otimes \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big)$$

A generalização da aplicação de n portas de Hadamard a um ket qualquer é:

$$H^{\otimes n}\,|z\rangle = \sum_{w=0}^{2^n-1} (-1)^{z\cdot w}\,|w\rangle$$

onde $z \cdot w = \sum_{i=0}^{2^n-1} z_i w_i$
Voltando ao $|\psi_3\rangle$ e fazendo uso da aplicação de portas de Hadamard à um ket qualquer:

$$|\psi_3\rangle = \frac{1}{2^{\frac{n}{2}}} \left[\frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{a\cdot x}(-1)^{x\cdot y}\,|y\rangle\right] \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \Rightarrow$$

$$|\psi_3\rangle = \frac{1}{2^n} \left[\sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{x(a\oplus y)}\,|y\rangle\right] \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

Mas,

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{x(a\oplus y)} = \delta_{a,y}$$

onde δ é o delta de Kronecker, que assume valor 1 caso os dois parâmetros sejam iguais e 0 caso contrário. É interessante observar que o valor de y é conhecido e que o delta de Kronecker terá valor 1 quando $a = y$,

logo, saberemos o valor de a, pois o estado resultante será da forma $|a\rangle$. O circuito será executado uma única vez e serão medidos os registradores dos n-qubits, encontrando os n-bits da string a, com probabilidade de 100% de acerto [Pre06]. Assim, podemos visualizar que o algoritmo resolve o problema a que se propõe visto que é possível conhecer o valor de a.

Como só é feita uma única passagem pelo circuito, o tempo decorrido para a solução é o da passagem pelo oráculo, portanto, podemos observar que o custo desta operação é $O(1)$, que é menor que o custo da solução clássica equivalente.

Existe uma versão recursiva do algoritmo de Bernstein-Vazirani, tolerante a pequenas falhas, que mostra que o problema proposto pode ser resolvido de forma ainda mais eficiente [Bac06]. Também já foram realizadas implementações práticas diversas desse algoritmo: [BLC$^+$03a], [BLC$^+$03b] e [DYW$^+$01].

5 Problema de Simon

O problema de Simon possui a seguinte descrição:

> Suponha que existe um oráculo que computa a função:
>
> $$f : \{0,1\}^n \to \{0,1\}^n$$
>
> Sabe-se que: $f(x) = f(y) \Leftrightarrow y = x \oplus s$ onde s é um número composto de n bits e que denominamos "*período*" [Pre06].
>
> Como s é um número binário composto de n bits, s é um valor entre 2^n valores possíveis.
>
> O problema de Simon pode ser classificado como um problema de decisão, e o objetivo é descrever quando existe ou não um período (ou seja, quando $s = 0$ e quando $s \neq 0$) [EW04].

Em termos da existência ou não deste período e dada a condição de igualdade entre $f(x)$ e $f(y)$, duas configurações são possíveis para a função f:

1. f é uma função 1-pra-1 (ou injetiva), quando $f(x) = f(y)$ se e somente se $x = y$.
2. f é uma função 2-pra-1, quando $f(x) = f(y)$ se e somente se $x = y \oplus s$ ou $x = y$.

5.1 Solução Clássica

A solução clássica para o problema de Simon é feita através de muitas consultas ao oráculo na tentativa de achar s com uma grande probabilidade de acerto.

Esse grande número de consultas não é à toa. Isso ocorre porque precisamos encontrar um par de números x e y tais que $x \oplus y = s$. Para entender melhor o porquê desse par de números, basta acompanhar abaixo:

$$\begin{aligned} x \oplus y = s \Rightarrow \\ x \oplus x \oplus s = s \Rightarrow \\ s = s \end{aligned}$$

Pois vimos anteriormente que $y = x \oplus s$ e que se temos um número quaisquer z, teremos que $z \oplus z = 0$.

Porém, como saber quais são os valores de x e y desejados? Através dos sucessivos testes, como foi dito anteriormente. Isso faz com que a resolução clássica seja bastante custosa e que não haja uma possibilidade totalmente segura de acerto.

Para entender melhor como funciona, vamos considerar o caso de um período s escolhido em $\{0,1\}^n \to \{0^n\}$. Agora consideremos um algoritmo probabilístico que já consultou a função k vezes, ou seja, $x_1, \ldots, x_k$. Agora queremos saber o quanto de informações é possível obter a respeito de s a partir dos pares $(x_i, f(x_i))$.

Temos duas situações:

- Existe um par x_i e x_j (onde $1 \leq i$ e $j \leq k$), de modo que $f(x_i) = f(x_j)$. Nesse caso, já temos informações suficientes do algoritmo para determinar $s : s = x_i \oplus x_j$.

- Suponha que o par dito anteriormente não exista. Então todos os $f(x_i)$ são distintos e o a não é nenhum dos $\begin{pmatrix} k \\ 2 \end{pmatrix}$ valores de $x_i \oplus x_j$.

  A probabilidade de sucesso para uma próxima consulta é de:

$$\frac{k}{2^n - 1 - \begin{pmatrix} k \\ 2 \end{pmatrix}}$$

  pois existem pelo menos $2^n - 1 - \begin{pmatrix} k \\ 2 \end{pmatrix}$ possíveis valores de escolha bem sucedidos para a $(k+1)$-ésima consulta. E $f(x_{k+1})$ deve se igual a um dos $f(x_i), i \in [1,k]$. Somando $k \in \{1, \ldots, m\}$ temos:

$$\sum_{k=1}^{m} \frac{k}{2^n - 1 - \begin{pmatrix} k \\ 2 \end{pmatrix}} \leq \sum_{k=1}^{m} \frac{k}{2^n - k^2} \leq \frac{m^2}{2^n - m^2}$$

  Visto a necessidade de uma probabilidade constante, devemos escolher $m = \Omega(2^{\frac{n}{2}})$, ou seja, um algoritmo clássico deve fazer um número de consultas exponencial em relação à entrada a fim de determinar com probabilidade maior que a de uma constante o valor de s [Vaz07].

Lembremos ainda que o problema de Simon não é encontrar exatamente o valor de s e sim classificar a função em 2-para-1 ou 1-para-1(injetiva). Assim, depois de $2^{\frac{n}{4}}$ consultas clássicas ao oráculo, a probabilidade de classificarmos corretamente a função f é:

$$P(sucesso) < \frac{1}{2} + \frac{1}{2^{\frac{n}{2}}}$$

Observe que a probabilidade de sucesso tende a $\frac{1}{2}$ quando n é grande, isso significa que, nestas condições, a classificação dada por um algoritmo clássico é tão segura quanto "chutar" uma das duas opções possíveis [Pre06].

Em termos de custo assintótico, Mosca afirma que:

> Qualquer algoritmo clássico que resolva este problema com probabilidade superior a $\frac{2}{3}$ pra qualquer f, custará (em termos de consultas ao oráculo), no mínimo $\Omega(2^{\frac{n}{3}})$ [Mos07].

5.2 Algoritmo de Simon

O conteúdo a seguir é uma compilação de [IB05] e [Mos07].

Considere uma função $f : \{0,1\}^n \to X$, pra algum conjunto finito X, onde nós temos a promessa que existe uma string "*oculta*" $s = s_1 s_2 \ldots s_n$ de modo que $f(x) = f(y)$ se e somente se $x = y$ ou $x = y \oplus s$. Vamos considerar o domínio $\{0,1\}^n$ de f como o espaço vetorial $\mathbb{Z}_2^n$ que leva em $\mathbb{Z}_2$. Por conveniência, vamos assumir que $X \subseteq \{0,1\}^n$.

Antes de começar a descrever o algoritmo de Simon, vamos fazer algumas considerações sobre a aplicação da porta de Hadamard à n-qubits. Já sabemos que ela é da forma:

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} (-1)^{x \cdot z} |z\rangle$$

O que acontece se aplicarmos $H^{\otimes n}$ à superposição de dois estados base $|0\rangle + |s\rangle$?

$$\begin{aligned} H^{\otimes n}\Big(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|s\rangle\Big) &== \frac{1}{\sqrt{2^{n+1}}} \sum_{z=0}^{2^n-1} |z\rangle + \sum_{z=0}^{2^n-1} (-1)^{s \cdot z} |z\rangle \\ &= \frac{1}{\sqrt{2^{n+1}}} [-1 + (-1)^{s \cdot z}] |z\rangle \end{aligned}$$

Note que se $s \cdot z = 1$ nós temos que $[-1 + (-1)^{s \cdot z}] = 0$ e o estado $|z\rangle$ é perdido na superposição, caso contrário, $|z\rangle$ continua com a amplitude $\frac{1}{\sqrt{2^{n-1}}}$. Vamos definir $s^{\perp} = \{z \in \{0,1\}^n | s \cdot z = 0\}$. Note que $s^{\perp}$ é subespaço vetorial de $\mathbb{Z}_2^n$ e é ortogonal ao subespaço $S = \{0, s\}$, também chamado de complemento ortogonal de S e denotado $S^{\perp}$. Isso implica que $dim(S) + dim(S^{\perp}) = dim(\mathbb{Z}_2^n) = n$ e $s^{\perp}$ tem dimensão $n - 1$.

$$H^{\otimes n}\Big(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|s\rangle\Big) = \frac{1}{\sqrt{2^{n-1}}} \sum_{z \in \{s\}^{\perp}} |z\rangle .$$

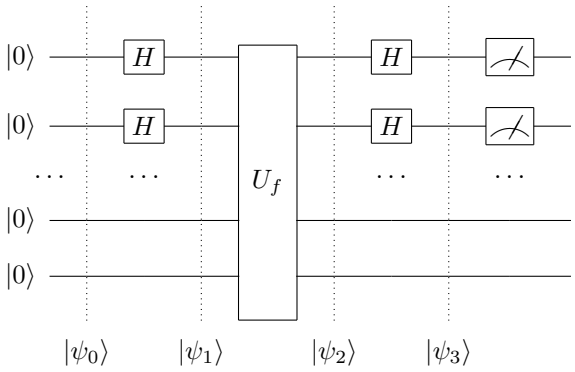
Vimos então como a porta de Hadamard mapeia $\frac{1}{\sqrt{2^n}}|x\rangle + \frac{1}{\sqrt{2^n}}|x \oplus s\rangle$ em uma superposição uniforme de estados $z \in s^{\perp}$. Esse raciocínio é o ingrediente para analisar o algoritmo de resolução do problema de Simon.

Assumimos que existe um oráculo reversível U_f que implementa f da seguinte forma:

$$U_f : |x\rangle |b\rangle \rightarrow |x\rangle |b \oplus f(x)\rangle$$

5.2.1 Circuito para o algoritmo de Simon

O circuito proposto para resolução é mostrado abaixo:



Circuito Resolvedor do algoritmo de Simon

5.2.2 Algoritmo para o problema de Simon

1. Inicialize um contador $i = 1$
2. Prepare a entrada $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0\rangle$
3. Aplique U_f para produzir o estado

   $$\sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$$

4. (opcional) Meça o segundo registrador
5. Aplique $H^{\otimes n}$ ao primeiro registrador
6. Meça o primeiro registrador e salve o valor w_i
7. Se a dimensão do intervalo de $\{w_i\}$ é igual a $n - 1$:
   (a) Vá ao passo 8
   (b) Caso contrário, incremente i e volte ao passo 2
8. Resolva a equação linear $Ws^T = 0^T$ e então s é a única solução não-trivial
9. Retorne s

Vamos entender melhor cada passo do algoritmo. Observe que o passo 2 é equivalente ao $|\psi_1\rangle$ do circuito, onde temos a aplicação da porta de Hadamard à entrada. Escrevendo em detalhes:

$$\begin{aligned} |\psi_1\rangle &= H^{\otimes n}|\psi_0\rangle \Rightarrow \\ |\psi_1\rangle &= (H^{\otimes n} \otimes I^{\otimes n})(|0\rangle^{\otimes n} \otimes |0\rangle^{\otimes n}) \Rightarrow \\ |\psi_1\rangle &= \Big(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle\Big) \otimes |0\rangle \Rightarrow \\ |\psi_1\rangle &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle\,|0\rangle \end{aligned}$$

Note que $\{0,1\}^n$ pode ser particionado em 2^{n-1} pares de strings da forma $\{x, x\oplus s\}$. Seja I um subconjunto de $\{0,1\}^n$ consistindo de um representante de cada um desses pares. Note então que o passo 3 (equivalente ao $|\psi_2\rangle$ do circuito) pode ser re-escrito como:

$$\frac{1}{\sqrt{2^{n-1}}} \sum_{x\in I} \frac{1}{\sqrt{2}}(|x\rangle + |x\oplus s\rangle)\,|f(x)\rangle$$

Depois de medirmos o segundo registrador no passo 4 para obter o valor de $f(x)$, o primeiro registrador será mantido em superposição $\frac{1}{\sqrt{2}}(|x\rangle + |x\oplus s\rangle)$.

Após a aplicação de Hadamard, no passo 5, o primeiro registrador será um equilíbrio de superposições de elementos de $s^\perp$. Assim, os valores w_i medidos no passo 6 serão elementos do $s^\perp$ selecionados uniformemente de forma aleatória. Isso significa que o passo 7 a dimensão do intervalo de $\{w_i\}$ é igual a $n-1$, ou seja, $intervalo\{w_i\} = s^\perp$.

Segue-se que 0 e s são as únicas soluções para a equação linear no passo 8, e que s pode ser encontrada através de eliminação Gaussiana módulo 2 em tempo polinomial em n. Para clarear mais a compreensão do custo do algoritmo, basta observar o vínculo da quantidade de vezes que o algoritmo ao contador i que, no máximo, será de tamanho n, o que indica que o algoritmo será executado n vezes, o que conduz ao tempo polinomial $O(n)$.

6 Transformada de Fourier Quântica

Fourier, físico e matemático francês, é respeitado por suas descobertas matemáticas que tiveram grande utilização prática. Ele estava interessado no estudo da propagação do calor, impulsionado especialmente pela emergente revolução industrial [Her89], e publicou, em 1807, um estudo sobre as senóides para representar distribuições de temperatura. O artigo continha a controversa afirmação de que qualquer sinal periódico e contínuo poderia ser representado pela soma de de ondas senoidais [Smi99], a chamada Transformada de Fourier.

A curva senoidal é uma espécie de arquétipo ondulatório; seu perfil curvilíneo é o que a maioria das pessoas têm em mente quando visualizam uma onda. As cordas em vibração e as pequenas ondulações em um lago tomam a cada momento a forma de onda senoidais [Her89]. Uma pergunta interessante que pode surgir é: Por que utilizar senóides ao invés de, por exemplo, ondas quadradas ou triangulares? Apesar de ser possível, o objetivo da decomposição é lidar com algo com algo mais simples do que o sinal original. Além disso, existem relações práticas com as ondas que tornam inviáveis essas outras possibilidades [Smi99].

O termo geral "*Transformada de Fourier*" pode ser subdividido em quatro categorias, baseadas nos quatro tipos básicos de sinais que são encontrados:

1. Aperiódico-contínuo: Esses sinais se extendem para o infinito tanto pelo lado negativo quanto positivio, em particular, sem repetição de algum padrão.

2. Periódico-contínuo: Sinais que se repetem segundo um padrão, desde $-\infty$ até $+\infty$. Essa versão da transformada de Fourier chama-se Séries de Fourier

3. Aperiódico-discreto: Esses sinais são definidos somente em alguns pontos e não se repetem segundo algum padrão. A esse tipo de transformada denominamos Transformada Discreta de Fourier no tempo.

4. Periódico-discreto: Sã os sinais discretos que se repetem de acordo com um período. Essa classe da transformada de fourier é chamada de Transformada Discreta de Fourier [Smi99].

A transforma de Fourier é utilizada em várias áreas do conhecimento, desde a própria matemática até o processamento digital de imagens. O uso mais comum das transformações de Fourier é no processamento de sinais. Um sinal é dado no domínio do tempo: como uma função de mapeamento de tempo para amplitude. A análise de Fourier nos permite expressar o sinal como uma soma ponderada de senóides de freqüências variadas deslocadas em fase. Os pesos e as fases associados com as freqüências caracterizam o sinal no domínio de freqüência [CLRS02].

Um caso particular é análise de ondas sonoras produzidas por diversos instrumentos musicais, em termos do alfabeto ondulatório senoidal de Fourier. Ainda quando reproduz a mesma nota, cada instrumento emite o seu "*timbre*" próprio - uma diferença que se reflete em sua receita de vibração. Cada instrumento deixa no ar a sua "*impressão digital de Fourier*" [Her89].

6.1 Transformada Discreta de Fourier

Computadores digitais só podem lidar com informações que são discretas e finitas em tamanho. Ou seja, em se tratando deste domínio, utiliza-se a Transformada Discreta de Fourier, também chamada DFT [Smi99].

6.2 Definição

Seja x um vetor $[x_0, x_1, \ldots, x_{n-1}]^T$, onde $x_i \in \mathbb{C}$ e seja y a Transformada Discreta de x, denotada por $y = DFT\{x\}$, onde os coeficientes de Fourier de y são denotados por:

$$y_k \triangleq \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \cdot e^{j \cdot \frac{2\pi}{N} \cdot i \cdot k}$$

Lembrando que $\frac{2\pi}{N}$ é normalmente conhecido por ω_0.

A Transformada Discreta de Fourier cria uma conexão entre dois vetores [IB05].

A DFT é usada em várias aplicações da computação, por exemplo: análise espectral, compressão de dados, equações diferenciais parciais, multiplicação polinomial, multiplicação de inteiros grandes, dentre outros.

6.3 Transformada de Fourier Quântica

Como vimos no postulados da Mecânica Quântica, sistemas físicos fechados também são representados por vetores complexos chamados superposições e que são representados pela notação bra-ket [dL07].

Vamos agora definir a Transformada Quântica de Fourier (QFT), denotada por QFT ou por apenas F. Esta transformada é a mesma transformação proposta pela DFT mas com a notação convencional modificada. Atua em uma base ortornomal $|0\rangle, \ldots, |N-1\rangle$, ou de modo equivalente, $|i\rangle, i = 1, 2, \ldots N-1$ e é definida como um operador linear com a seguinte ação sobre as bases:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{j \cdot \frac{2\pi}{N} \cdot i \cdot k} |k\rangle$$

De modo equivalente, a ação em um estado arbitrário é descrita como:

$$\sum_{j=0}^{N-1} x_j |j\rangle \rightarrow \sum_{k=0}^{N-1} y_k |k\rangle$$

A QFT é uma transformação unitária, como vamos verficar a seguir.

Seja $N = 2^n$, onde n é um inteiro e a base $|0\rangle + \ldots + |2^n - 1\rangle$ é a base computacional para um computador quântico de n qubits.

O estado $|j\rangle$ pode ser representado através da representação binária $j = j_1 j_2 \ldots j_n$, ou ainda, $j = j_n 2^{n-1} + j_2 2^{n-2} + \ldots + j_n 2^0$.

Com a utilização de um pouco de álgebra, a QFT pode ser denotada como o seguinte produto:

$$|j_1, \ldots, j_n\rangle \rightarrow \frac{(|0\rangle + e^{2\cdot\pi\cdot i0\cdot j_n}|1\rangle)\ldots(|0\rangle + e^{2\cdot\pi\cdot i0\cdot j_1 j_2 \ldots j_n}|1\rangle)}{2^{\frac{n}{2}}}$$

Através da equivalência da representação acima e da definição da QFT, podemos ter a seguinte equivalência [NL05]:

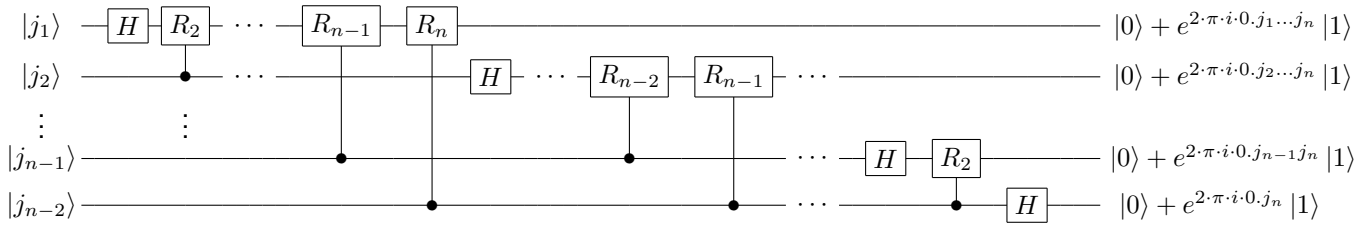
$$\begin{aligned}
|j\rangle &\rightarrow \frac{1}{2^{\frac{n}{2}}}\sum_{k=0}^{2^n-1} e^{2\cdot\frac{\pi}{2^n}\cdot i\cdot j\cdot k}|k\rangle \\
&= \frac{1}{2^{\frac{n}{2}}}\sum_{k_1=0}^{1}1\ldots\sum_{k_n=0}^{1} e^{2\cdot\pi\cdot i\cdot j(\sum_{l=1}^{n}k_l 2^{-l})}|k_1\ldots k_n\rangle \\
&= \frac{1}{2^{\frac{n}{2}}}\sum_{k_1=0}^{1}\cdots\sum_{k_n=0}^{1}\bigotimes_{l=1}^{n} e^{2\cdot\pi\cdot i\cdot j\cdot k_l\cdot 2^{-l}}|k_l\rangle \\
&= \frac{1}{2^{\frac{n}{2}}}\bigotimes_{l=1}^{n}\left[\sum_{k_l=0}^{1} e^{2\cdot\pi\cdot i\cdot j\cdot k_l\cdot 2^{-l}}|k_l\rangle\right] \\
&= \frac{1}{2^{\frac{n}{2}}}\bigotimes_{l=1}^{n}\left[|0\rangle + 2^{2\cdot\pi\cdot i\cdot j\cdot 2^{-l}}|1\rangle\right] \\
&= \frac{(|0\rangle + e^{2\cdot\pi\cdot i0\cdot j_n}|1\rangle)\ldots(|0\rangle + e^{2\cdot\pi\cdot i0\cdot j_1 j_2\ldots j_n}|1\rangle)}{2^{\frac{n}{2}}}
\end{aligned}$$

6.4 Circuito para a Transformada de Fourier Quântica

Seja a porta R_k denotada pela transformação abaixo:

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\cdot\pi\cdot i}{2^k}} \end{bmatrix}$$

Um circuito que resolve a QFT é dado a seguir:



Circuito Resolvedor da Transformada de Fourier Quântica

Para verificar que o circuito de fato computa a QFT, vamos analisar a sua ação sobre um estado de entrada $|j_1\ldots j_n\rangle$. A aplicação da porta de Hadamard ao primeiro qubit resulta em:

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

Lembrando que $e^{2\cdot\pi\cdot 0.j_1} = -1$ para $j_1 = 1$, e $+1$ para outro valor. Aplicando agora a porta R_2 controlada, temos o estado:

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

A aplicação sucessiva de R_3-controlada até R_n adiciona, a cada vez, um bit extra na fase do coeficiente do primeiro estado $|1\rangle$. Após esse processo, temos:

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

De modo análogo para o segundo q-bit. A porta de Hadamard cria o seguinte estado:

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

E as portas controladas de R_2 até R_{n-1} produzem:

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

De modo análogo, para cada um dos q-bits, chega-se ao estado final:

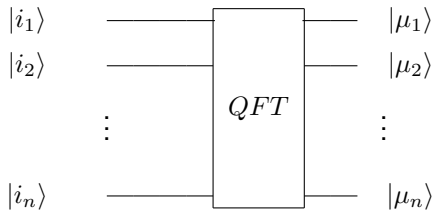
$$\frac{1}{2^{\frac{n}{2}}}(|0\rangle + e^{2\cdot\pi\cdot i\cdot 0.j_1 j_2\ldots j_n}\,|1\rangle)(|0\rangle + e^{2\cdot\pi\cdot i\cdot 0.j_2\ldots j_n}\,|1\rangle)\ldots(|0\rangle + e^{2\cdot\pi\cdot i\cdot 0.j_n}\,|1\rangle)$$

Revertendo a ordem dos qubits, tem-se:

$$\frac{1}{2^{\frac{n}{2}}}(|0\rangle + e^{2\cdot\pi\cdot i\cdot 0.j_n}\,|1\rangle)(|0\rangle + e^{2\cdot\pi\cdot i\cdot 0.j_{n-1} j_n}\,|1\rangle)\ldots(|0\rangle + e^{2\cdot\pi\cdot i\cdot 0.j_1 j_2\ldots j_n}\,|1\rangle)$$

Olhando para o estado acima, podemos concluir que trata-se da Transformada de Fourier Quântica para a entrada dada, logo, o circuito satisfaz seu propósito [NL05].

A troca na ordem dos qubits está abstraída do circuito anterior, a fim de facilitar a compreensão da equivalência do primeiro com a QFT. Esta pode ser feita com uma porta do tipo "*swap*". De modo geral, abstraímos a complexidade do circuito da QFT apenas denotando-o como abaixo [IB05]:



Simplificação do circuito da Transformada de Fourier Quântica

6.5 Complexidade

O melhor algoritmo clássico para computar a transformada de Fourier de n elementos é o algoritmo da Transformada de Fourier Rápida (FFT),que utiliza $\Theta(n \cdot 2^n)$ portas [NL05].

A FFT reduziu o custo da DFT, possibilitando que uma ampla classe de problemas pudessem ser atacados por computadores. O sucesso da FFT é tamanho que muitos outros problemas puderam ser reduzidos à Transformada de Fourier, desde a multiplicação de números e polinômios até o processamento digital de imagens [Lom04].

A QFT utiliza portas de Hadamard e $n-1$ rotações condicionais no primeiro q-bit - um total de n portas. Em seguida, existe uma operação de Hadamard e $n-2$ rotações sobre o segundo q-bit, em um total de $n+(n-1)$ operações. Além disso, temos a operação "*swap*" onde, no máximo, $\frac{n}{2}$ portas são usadas (cada porta de troca é constituída de 3 portas *cnot*). Assim, temos:

$$\begin{aligned} C_{usto} &= \sum_{i=0}^{n-1}(n-i) + \tfrac{n}{2} \\ &= \tfrac{n\cdot(n-1)}{2} + \tfrac{n}{2} \\ &= \Theta(n^2) \end{aligned}$$

Portanto, o circuito fornece um algoritmo $\Theta(n^2)$ para a QFT [NL05]. O custo de consultas que esse algoritmo faz é exponencialmente menor em relação ao equivalente clássico mais eficiente.

A maioria dos algoritmos quânticos são exponencialmente mais rápidos que seus equivalentes clássicos e uma boa explicação para isso se deve à QFT [Lom04].

6.6 Estimativa de Fase

Esta seção é baseada majoritariamente em [Mos07] e [NL05].

Para compreender o conceito de estimativa de fase, vamos relembrar o que já foi visto em algoritmos anteriores. Em particular, no algoritmo de Deutsch-Jozsa, a aplicação das portas de Hadamard após a saída do oráculo visava recuperar informação que estava contida na fase relativa ao estado.

A porta de Hadamard é auto-inversível, ou seja, $H^2 = \mathbb{I}$, podendo ser usada para codificar informações dentro das fases. Para exemplificar, seja $|x\rangle$, onde $x \in \{0,1\}$. É fácil verificar que:

$$\begin{aligned} H|x\rangle &= \left(\frac{1}{\sqrt{(2)}}|0\rangle + \frac{(-1)^x}{\sqrt{2}}|1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{x\cdot y}|y\rangle \end{aligned}$$

Como dito, por ser auto-inversível basta que apliquemos uma porta de Hadamard ao resultado acima, para obtermos o valor de x. Isso significa que estamos extraindo, ou melhor, decodificando o valor de x de uma fase. Abaixo, mostramos esse processo para o caso anterior:

$$H\left(\frac{1}{\sqrt{(2)}}|0\rangle + \frac{(-1)^x}{\sqrt{2}}|1\rangle\right) = |x\rangle$$

Generalizando, podemos ver a porta de Hadamard aplicada a n-qubits como se esta codificasse informações referentes ao $|x\rangle$ nas fases $(-1)^{x\cdot y}$ do estado base y:

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}}(-1)^{x\cdot y}|y\rangle$$

Se aplicarmos Hadamard novamente aos n-qubits acima, teremos a decodificação do valor de x:

$$\begin{aligned} H^{\otimes n}\left[\frac{1}{\sqrt{2^n}}\sum_{y\in\{0,1\}}(-1)^{x\cdot y}|y\rangle\right] &= H^{\otimes n}(H^{\otimes n}|x\rangle) \\ &= \mathbb{I}|x\rangle \\ &= |x\rangle \end{aligned}$$

No caso acima, $(-1)^{x\cdot y}$ é a fase de uma forma particular. No geral, uma fase é um número complexo da forma $e^{2\Pi}$ para qualquer número real $\varphi \in (0,1)$. A fase -1 corresponde a $\omega = \frac{1}{2}$.

A transformação realizada pela Porta de Hadamard é um caso especial da Transformada de Fourier [CEMM96].

6.7 O problema da Estimativa de Fase

O problema da estimativa de fase pode ser descrito como se segue:

Entrada: (1) Uma caixa-preta que realiza a operação U^j-controlada, para inteiros j; (2) um auto-estado $ketu$ de U com autovalor $e^{2\cdot i\cdot \pi\cdot \varphi_u}$; e (3) $t = n + \lceil log(2 + \frac{1}{2\cdot\epsilon})\rceil$ q-bits inicializados no estado $ket0$.

Saída: Um bom estimador para o parâmetro de fase φ, ou seja, uma aproximação de n bits $\widetilde{\varphi_u}$ para φ_u.

Tempo de Processamento: $O(t^2)$ operações com uma chamada da "caixa-preta" U^j-controlada. Funciona com probabilidade de pelo menos $1 - \epsilon$.

Em si, a estimativa de fase resolve um problema não trivial e muito interessante do ponto de vista físico: o de como conhecer o autovalor associado a um dado autovetor no espaço unitário. Seu verdadeiro uso, no entanto, aparece do fato de que outros problemas interessantes podem ser reduzidos ao problema da estimativa de fase [NL05].

Suponha que um operador unitário U tenha autovetor $|u\rangle$ com autovalor $e^{2\cdot i\cdot pi\cdot\varphi}$, no qual o valor de φ é desconhecido. O objetivo é estimar φ.

O algoritmo que resolve o problema da estimativa de fase quântica é descrito abaixo:

1. $|0\rangle\,|u\rangle$ (estado inicial)
2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle\,|u\rangle$ (cria superposição)
3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle\, U^j\,|u\rangle$ (aplica a caixa-preta)
4. $= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\cdot\pi\cdot i\cdot j\cdot\varphi_u}\,|j\rangle\,|u\rangle$ (resultado da caixa-preta)
5. $\rightarrow |\widetilde{\varphi_u}\rangle\,|u\rangle$ (aplica a transformada de Fourier inversa)
6. $\rightarrow \widetilde{\varphi_u}$ (mede o primeiro registro)

A estimativa de fase é necessária em vários algoritmos quânticos, em particular, algoritmos de contagem, fatorização de números grandes, etc., e é através desse problema que motivamos o estudo da Transformada Quântica de Fourier [IB05].

7 Busca de Ordem

Um problema comum é encontrar o período r de uma função $\mathbb{N} \rightarrow \mathbb{N}$ que é periódica segundo a adição. Essa função f satisfaz $f(x) = f(y)$ para valores distintos de x e y se, e somente se, x e y diferem por um inteiro múltiplo de r [Mer06]. A esse cálculo denominamos cálculo da ordem.

O problema de Shor visa encontrar o cálculo da ordem, que é uma redução do problema da fatoração. Vamos descrever mais detalhadamente como este problema consiste.

A redução da fatoração de N ao problema de achar a ordem de um inteiro positivo x menor que N pode ser descrita da seguinte forma:

Se x e N possuem fatores comuns, então *gcd*(x,N) (*gcd* é uma função que calcula o máximo divisor comum) resulta em um fator de N. Conseqüentemente, é suficiente investigar o caso que x é co-primo[1] de N. A ordem de x módulo N é definida como o menor inteiro positivo r tal que:

$$x^r \equiv 1 \ \ mod \ \ N$$

[1] Ser co-primo significa que o maior divisor comum de x e N é 1.

Se r é par, definimos da seguinte forma:

$$x^{\frac{r}{2}} \equiv 1 \ \ mod \ \ N$$

Se N ainda possui fatores, eles podem ser calculados através de chamadas recursivas.

A notação acima significa que se y é o resto de $x^{\frac{r}{2}}$ dividido por N e, por definição, $0 \leq y < N$, note que y satisfaz $y^2 \equiv 1 \ \ mod \ \ N$, o que significa que N divide $(y-1)(y+1)$.

Se $1 < y < N-1$, os fatores $y-1$ e $y+1$ satisfazem $0 < y-1 < y+1 < N$, e consequentemente N não pode dividir $y-1$ e $y+1$ separadamente. A única alternativa é que tanto $y-1$ quanto $y+1$ sejam fatores de N (resultam em N quando multiplicados). Então, o *gcd*(y-1, N) e *gcd*(y+1,N) conduzem a fatores não triviais de N [PLM07].

O ponto mais importante do problema de Shor é a necessidade de computar o período r da função f. Este problema é significante pois o resultado da teoria dos números nos diz que ao conhecermos r nós poderemos obter os fatores de um número n. Calcular o maior divisor comum é simples e rápido, num procedimento conhecido desde a época de Euclides [WC98].

7.1 Solução Clássica

Suponha que temos um inteiro n que desejamos fatorar, ou seja, decompor este inteiro em um produto de números primos.

Para números pequenos, utiliza-se a heurística Rho de Pollard porém, por ser uma heurística, nem seu tempo de execução nem seu sucesso são garantidos. Abaixo o pseudo-código para a heurística, onde a entrada n é o número que desejamos fatorar.

7.1.1 Pseudo-código para a Heurística Rho de Pollard

POLLARD-RHO(n)
(1) $i \leftarrow 1$
(2) $x_1 \leftarrow RANDOM(0, n-1)$
(3) $y \leftarrow x_1$
(4) $k = 2$
(5) **while** (*TRUE*)
(6) do $i \leftarrow i + 1$
(7) $x_i \leftarrow (x_{i-1}^2 - 1 \ \ mod \ \ n)$
(8) $d \leftarrow mdc(y - x_i, n)$
(9) **if** $d \neq 1$ **and** $d \neq n$
(10) **then** *imprimir*(d)
(11) **if** $i = k$
(12) **then** $y \leftarrow x_i$
(13) $k \leftarrow 2 \cdot k$

A heurística Rho de Pollard diz que encontramos pelo menos um fator de n quando fatorarmos até $\Theta(\sqrt{n})$ [CLRS02].

Porém, ela não é de todo eficiente, visto que pode resultar somente em parte dos fatores de um número. O melhor algoritmo clássico para este problema possui custo $O(e^{c \cdot [(logn)^{\frac{1}{3}}] \cdot [(loglogn)^{\frac{2}{3}}]})$, ou seja, realiza a fatoração em tempo exponencial [Hay05].

Até o presente momento não é conhecido um algoritmo clássico eficiente para fatoração ou cálculo de ordem [Por07].

7.2 Algoritmo de Shor

O algoritmo de Shor para fatoração se sustenta sobre um resultado da Teoria dos Números clássica que relaciona o período de uma função periódica em particular aos fatores de um número inteiro.

7.3 Como encontrar os fatores de um número

Dada um inteiro n, o número a ser fatorado, e uma função $f_n(a) = x^a \ \ mod \ \ n$, onde x é um inteiro escolhido randomicamente e co-primo de n. Encontrar $f_n(a)$ é uma tarefa de custo exponencial.

É interessante entender a relação entre o que foi dito e a fatoração de números. Esta relação se torna clara quando a função que acabamos de construir $f_n(a)$ se torna periódica. Isso significa que, à medida que se aumenta o argumento da função, tomando $a = 0, 1, 2, k$, os valores da função $f_n(0), f_n(1), f_n(2), k$ expressam um padrão de repetição. Diferentes valores de x levam à diferentes padrões.

O número de valores a cada repetição do padrão é denominado "o período de x módulo n", denotado por r. A cada r-ésimo elemento, a seqüência $f_n(0), f_n(1), f_n(2), k$ é a mesma. Em particular, isso implica que:

$$x^r \equiv 1 \ \ mod \ \ n$$

Se o período r é par, então podemos escrever da seguinte forma:

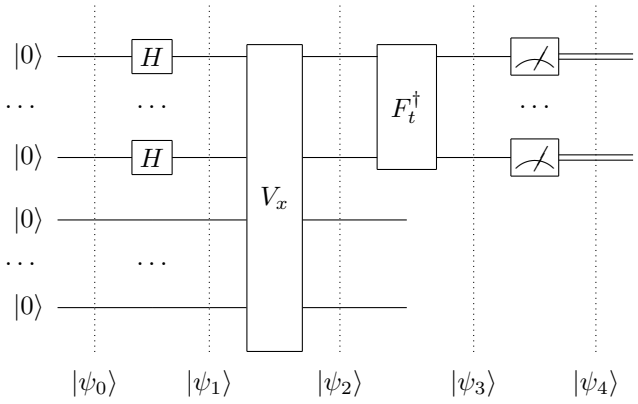
$$\begin{aligned}(x^{\frac{r}{2}})^2 &\equiv 1 \ \ mod \ \ n \\ (x^{\frac{r}{2}})^2 - 1 &\equiv 0 \ \ mod \ \ n \\ (x^{\frac{r}{2}})^2 - 1^2 &\equiv 0 \ \ mod \ \ n \\ (x^{\frac{r}{2}} - 1) \cdot (x^{\frac{r}{2}} + 1) &\equiv 0 \ \ mod \ \ n\end{aligned}$$

E a forma final nos revela que o produto $(x^{\frac{r}{2}} - 1) \cdot (x^{\frac{r}{2}} + 1)$ é algum inteiro múltiplo de n. Ou seja, dividindo n por $(x^{\frac{r}{2}} - 1) \cdot (x^{\frac{r}{2}} + 1)$ o resto é zero.

Assim, a menos que $x^{\frac{r}{2}} \equiv \pm 1 \ \ mod \ \ n$, pelo menos um dos termos do lado esquerdo da equação é um fator não-trivial de n. Conseqüentemente, temos uma boa chance de encontrar um fator de n através da computação de *gcd*$(x^{\frac{r}{2}} - 1, n)$ e *gcd*$(x^{\frac{r}{2}} + 1, n)$ [WC98].

7.4 Circuito Resolvedor do Algoritmo de Shor

Vamos entender melhor o algoritmo de Shor através da observação dos seus passos no circuito que o resolve, ilustrado abaixo:



Circuito Resolvedor do algoritmo de Shor

Primeiro fazemos a inicialização da entrada, que consiste apenas na entrada de kets zero:

$$|\psi_0\rangle = |0\rangle^{\otimes t} \otimes |0\rangle^{\otimes n}$$

As portas de Hadamard aplicadas ao primeiro registrador criam superposições. A aplicação de z portas de Hadamard à z kets zero é da forma:

$$H^{\otimes z} |0\rangle^{\otimes z} = \frac{1}{\sqrt{2^z}} \sum_{i=0}^{2^x-1} |i\rangle$$

Assim, para o estado $|\psi_1\rangle$, temos:

$$\begin{aligned} |\psi_1\rangle &= H^{\otimes t} |\psi_0\rangle \Rightarrow \\ |\psi_1\rangle &= H^{\otimes t}(|0\rangle^{\otimes t} \otimes |0\rangle^{\otimes n}) \Rightarrow \\ |\psi_1\rangle &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |0\rangle \end{aligned}$$

O passo seguinte é aplicar o oráculo V_x ao estado $|\psi_1\rangle$. O oráculo é linear e atua da seguinte forma:

$$V_x(|j\rangle |k\rangle) = |j\rangle \left|k + x^j\right\rangle$$

Logo, sabendo que $0 + a = a$, temos:

$$\begin{aligned} |\psi_2\rangle &= V_x |\psi_1\rangle \Rightarrow \\ |\psi_2\rangle &= V_x\Big(\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle |0\rangle \Big) \Rightarrow \\ |\psi_2\rangle &= \frac{1}{\sqrt{2^t}} V_x\Big(\sum_{j=0}^{2^t-1} |j\rangle |0\rangle \Big) \Rightarrow \\ |\psi_2\rangle &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle \left|0 + x^j\right\rangle \Rightarrow \\ |\psi_2\rangle &= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle \left|x^j\right\rangle \end{aligned}$$

O passo seguinte é a aplicação da Transformada de Fourier Inversa aos t primeiros qubits. A inversa da QFT é da forma:

$$F^\dagger\Big(\sum_{j=0}^{2^t-1} |j\rangle\Big) = |\psi_j\rangle$$

onde,

$$|\psi_j\rangle = \frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} \omega^{j \cdot k} |k\rangle$$

valendo lembrar que $\omega = e^{\frac{2 \cdot \pi \cdot i}{2^t}}$.
Assim, temos:

$$|\psi_3\rangle = F_t^\dagger |\psi_2\rangle \Rightarrow$$
$$|\psi_3\rangle = F_t^\dagger \Big(\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle \left|x^j\right\rangle \Big) \Rightarrow$$
$$|\psi_3\rangle = \frac{1}{\sqrt{2^t}} F_t^\dagger \Big(\sum_{j=0}^{2^t-1} |j\rangle \left|x^j\right\rangle \Big) \Rightarrow$$
$$|\psi_3\rangle = \frac{1}{\sqrt{2^t}} \left(\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} \sum_{j=0}^{2^t-1} \omega^{j\cdot k} |k\rangle \left|x^j\right\rangle \right)$$

O passo final do algoritmo de Shor utiliza frações contínuas. É um artifício puramente clássico. A expansão de uma fração contínua é utilizada para identificar qual múltiplo inverso do período r é obtido.

Uma fração contínua é uma expressão da forma:

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{k + \frac{1}{a_N}}}}$$

a qual é usualmente abreviada como $x = [a_0, a_1, k, a_N]$. Uma fração contínua truncada é dita "*convergente*". O n-ésimo convergente é escrito $[a_0, a_1, k, a_n]$. Existe um teorema sobre frações contínuas que enuncia que: "Se $\frac{p}{q}$ é um número racional satisfazendo $\left|\frac{p}{q} - x\right| < \frac{1}{2 \cdot q^2}$", então $\frac{p}{q}$ é um convergente da fração contínua. Mais além, o convergente é tal que *GCD*$(p, q) = 1$". Nós utilizaremos este teorema a fim de encontrar os racionais mais próximos aos termos $\frac{\lambda}{r}$ e conseqüentemente encontrar o período r.

Uma vez que r é conhecido, podemos obter os fatores da entrada a partir de *GCD*$(x^{\frac{r}{2}} - 1, n)$ e *GCD*$(x^{\frac{r}{2}} + 11, n)$ [WC98].

O custo do algoritmo em questão é dado pela seguinte soma: o custo da aplicação das portas de Hadamard e do oráculo V_x é $O(1)$ e a QFT possui custo $O(n^2)$, assim, temos:

$$\begin{aligned} C_{usto} &= O(1) + O(1) + O(n^2) \\ &= O(n^2) \end{aligned}$$

É fácil ver que o custo associado é menor que o custo da solução clássica mais equivalente para o mesmo problema.

8 Problema do Logaritmo Discreto

Seja p um número e seja $\mathbb{Z}_p^*$ o grupo[2] de inteiro $\{1, 2, \ldots, p-1\}$ fechado segundo a multiplicação módulo p.

Um número g em $\mathbb{Z}_p^*$ é denominado "*gerador*" (ou raiz primitiva módulo p) se as potências de g geram todos os $\mathbb{Z}_p^*$. Assim, cada elemento x de $\mathbb{Z}_p^*$ pode ser escrito unicamente como $x = g^y$ para algum $y \in \mathbb{Z}_{p-1}$. y é chamado o logaritmo discreto de x (em relação à g) e escrevemos $y = log_g \ \ x$. Note que a multiplicação de $x's \mod p$ corresponde à adição dos $y's \mod (p-1)$ então um gerador provê um modo de identificar $\mathbb{Z}_p^*$ como $\mathbb{Z}_{p-1}$.

O problema do logaritmo discreto é enunciado como se segue:

> Nós temos p e um gerador g de $\mathbb{Z}_p^*$. Para qualquer $x \in \mathbb{Z}_p^*$ desejamos computar o seu logaritmo discreto $y = log_g \ \ x$ [Joz00].

[2]Um grupo é um conjunto não-vazio dotado de um operação $\cdot$, e possui as seguintes propriedades: (1) $\cdot$ é associativa; (2) Existe um elemento neutro; e (3) Todo elemento deste conjunto possui um inverso neste mesmo conjunto em relação à $\cdot$ [Ger95].

8.1 Solução Clássica

Uma solução clássica para o problema do logaritmo discreto é o algoritmo "*baby-step giant-step*". O raciocínio de resolução do problema em questão parte do seguinte conhecimento:

> Dado um grupo cíclico G de ordem n, um gerador α do grupo e um elemento do grupo β. O problema é encontrar um inteiro x tal que:
>
> $$\alpha^x = \beta$$
>
> onde α, β e x são dados. O algoritmo baseia-se na re-escrita de x como $x = i \cdot m + j$ com m constante e $0 \leq i, j < m$. Assim, temos:
>
> $$\beta(\alpha^{-m})^i = \alpha^j$$

O algoritmo pré-computa α^j para vários valores de j. Então, fixa m e tenta valores de i que satisfaçam a igualdade para algum dos α^j pré-calculados.

8.1.1 Algoritmo "*baby-step giant-step*"

Entrada: Um grupo cíclico G de ordem n, um gerador α deste grupo e um elemento β. Saída: Um valor de x que satisfaça $\alpha^x = \beta$.

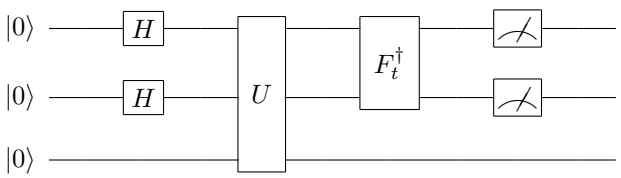
1. $m \leftarrow \lceil \sqrt{(n)} \rceil$
2. **Para todo** j, $0 \leq j < m$:
   (a) **Compute** α^j **e guarde o par** (j, α^j) **em uma tabela**
3. **Compute** α^{-m}
4. $\gamma \leftarrow \beta$
5. **Para todo** $i = 0$ **até** $(m - 1)$
   (a) **Verifique se** γ **é o segundo componente** (α^j) **de algum par armazenado na tabela**
   (b) **Caso seja, retorne** $i \cdot m + j$
   (c) **Senão** $\gamma \leftarrow \gamma \cdot \alpha^{-m}$

A solução acima tem custo $O(\sqrt{(n)})$ em relação à entrada [Sha71].

8.2 Solução Quântica

A solução quântica para o problema do logaritmo discreto possui como entradas (1) uma caixa-preta que realiza a operação $U|x_1\rangle|x_2\rangle|y\rangle \rightarrow |x_1\rangle|x_2\rangle|y \oplus f(x_1, x_2)\rangle$, para $f(x_1, x_2) = b^{x_1}a^{x_2}$; (2) um estado inicializado em $|0\rangle$ para armazenar a função avaliada e (3) dois registros de $t = O(\lceil logr \rceil + log(\frac{1}{\epsilon}))$ qubits inicializados em $|0\rangle$. A saída é o menor inteiro positivo s, tal que $a^s = b$.

8.2.1 Circuito Resolvedor do Problema do Logaritmo Discreto



O circuito acima resolve o problema do logaritmo discreto em passos que podem ser sintetizados abaixo:

1. $|0\rangle\,|0\rangle\,|0\rangle$
2. $\rightarrow \frac{1}{2^t}\sum_{x_1=0}^{2^n-1}\sum_{x_2=0}^{2^n-1}|x_1\rangle\,|x_2\rangle\,|0\rangle$
3. $\rightarrow \frac{1}{2^t}\sum_{x_1=0}^{2^n-1}\sum_{x_2=0}^{2^n-1}|x_1\rangle\,|x_2\rangle\,|f(x_1,x_2)\rangle$
   $= \frac{1}{2^t\sqrt{r}}\sum_{l_2=0}^{r-1}\left[\sum_{x_1=0}^{2^t-1}e^{\frac{2\pi i(sl_2x_1)}{r}}|x_1\rangle\right]\cdot$
   $\cdot\left[\sum_{x_2=0}^{2^t-1}e^{\frac{2\pi i(sl_2x_2)}{r}}|x_2\rangle\right]|\overline{f}(sl_2,l_2)\rangle$
4. $\rightarrow \sum_{l_2=0}^{r-1}\left|s\widetilde{\frac{l_2}{r}}\right\rangle\left|\widetilde{\frac{l_2}{r}}\right\rangle|\overline{f}(sl_2,l_2)\rangle$
5. $\rightarrow (s\widetilde{\frac{l_2}{r}},\widetilde{\frac{l_2}{r}})$
6. $\rightarrow s$

O passo 1 representa o estado inicial, onde apenas existem $|0\rangle^{\otimes 3}$ como entrada. Em seguida, utilizando portas de Hadamard, criam-se superposições para os dois primeiros registros. O terceiro passo representa a aplicação do oráculo U à entrada. A seguir, aplica-se a QFT Inversa aos dois primeiros registros. No passo 5 fazemos a medição e o passo 6 é obtido através do algoritmo generalizado de frações contínuas [NL05].

O algoritmo acima faz apenas um uso do oráculo U e mais $O(\lceil log \quad r\rceil^2)$ operações, funcionando com probabilidade de 100% de acerto. [NL05].

9 Considerações Finais

Vimos que o problema do subgrupo oculto pode ser resolvido por algoritmos quânticos de forma mais econômica que as melhores soluções clássicas existentes atualmente. Nos problemas de Deutsch-Jozsa e Bernstein-Vazirani o decréscimo do custo[3] foi linear e que nos demais algoritmos este decréscimo é exponencial.

A utilização da linguagem de circuitos, utilizada neste trabalho, favorece a compreensão de cada passo do algoritmo quântico relacionado, facilitando também a escrita da expressão resultante em cada um desses passos. Em virtude disso, a linguagem de circuitos é um bom recurso para expressão de algoritmos quânticos e que também favorece o aprendizado destes por iniciantes.

Vale citar que este trabalho compila conhecimento de várias fontes, a fim de organizar o conteúdo de forma simplificada e objetiva. Dessa forma, este trabalho pode ser utilizado como material introdutório ao estudo de algoritmos quânticos, pois aborda a partir da solução clássica, já compreendida pelo leitor, para então discorrer sobre a versão quântica e as vantagens da mesma.

O estudo de algoritmos quânticos abre as portas não só para um novo paradigma, capaz de computar certos problemas de modo mais econômico, como também possibilita uma aproximação maior da computação com o mundo físico no qual ela se insere, permitindo tirar proveito das características deste contexto e assim re-inventar a própria computação.

10 Trabalhos Futuros

Em trabalhos futuros pretende-se pesquisar sobre a teoria e os problemas em aberto da Complexidade Computacional, também com abordagem clássica e quântica.

[3] Este decréscimo de custo refere-se ao número de consultas ao oráculo em termos da entrada

References

[Bac06] D. Bacon, *Quantum computing the recursive and nonrecursive bernstein-vazirani algorithm*, Tech. report, Department of Computer Science e Engineering, University of Washington, 2006.

[BLC+03a] E. Brainis, L.-P. Lamoureux, N. J. Cerf, Ph. Emplit, M. Haelterman, and S. Massar, *Fiber-optics implementation of the deutsch-jozsa and bernstein-vazirani quantum algorithms with three qubits*, Physical Review Letters **Vol 90 - Number 15** (2003), 157902–1–157902–4.

[BLC+03b] _____, *Optical implementation of deutsch-jozsa and bernstein-vazirani quantum algorithms in eight dimensions*, Tech. report, Universit´e Libre de Bruxelles, 2003.

[BV93] Ethan Bernstein and Umesh Vazirani, *Quantum complexity theory*, Proc. 25th Annual ACM Symposium on the Theory of Computing **ACM Press, New York** (1993), 11–20.

[CEMM96] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, *Quantum algorithms revisited*, Phil. Trans. R. Soc. Lond. **A** (1996).

[CLRS02] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Algoritmos - teoria e prática*, Elsevier, 2002.

[Deu85] David Deutsch, *Quantum theory, the church-turing principle and the universal quantum comouter*, Proc. R. Soc. Lond. **A400** (1985), 97–117.

[dL07] Aercio F. de Lima, *Mecânica quântica: uma introdução para computação quântica e informação quântica*, 2o. Workshop-Escola de Computação e Informação Quânticas, 2007.

[dLJ05] Aercio F. de Lima and Bernardo Lula Jr, *Mini-cursos em matemática aplicada e computacional*, Uma Introdução à Computação Quântica, Sociedade Brasileira de Matemática Aplicada e Computacional, 2005.

[DYW+01] Jiangfeng Du, BangJiao Ye, Jihui Wu, Kelin Gao, Mingjun Shi, and Rongdian Han, *Implementation of a quantum algorithm to solve the bernstein-vazirani parity problem without entanglement on an ensemble quantum computer*, Physical Review A **Volume 64** (2001), 042306–1–042306–5.

[EW04] J. Eisert and M. M. Wolf, *Quantum computing*, http://arXiv.org/quant-ph/0401019 v2 **1** (2004), no. 2.

[Ger95] Judith L. Gersting, *Fundamentos matemáticos para a ciência da computação*, 3 ed., W . H. Freeman and Company, 1995.

[Hay05] Matthew Hayward, *Quantum computing and shors algorithm*, Lecture Notes from University of Illinois, February 2005.

[Her89] Nick Herbert, *A realidade quântica - nos confins da nova física*, Francisco Alves Editora S.A., 1989.

[IB05] Sandor Imre and Ferenc Balazs, *Quantum computing and communications - an engineering approach*, John Wiley & Sons, Ltd, 2005.

[Joz96] Richard Jozsa, *Quantum algorithms and the fourier transform*, Proc. Roy. Soc. Lond **A for the Proceedings of the Santa Barbara Conference on Quantum Coherence and Decoherence** (1996).

[Joz00] _____, *Quantum factoring, discrete logarithms and the hidden subgroup problem*, http://www.arXiv.org/quant-ph/0012084 **1** (2000).

[Lom04] Chris Lomont, *The hidden subgroup problem - review and open problems*, http://arXiv.org/quant-ph/0411037, November 2004.

[Mer03] N. David Mermin, *From cbits to qbits teaching computer scientists quantum mechanics*, American Journal of Physics **71** (2003), 23–30.

[Mer06] _____, *Lecture notes on quantum computation*, Cornell University, Physics 481-681, CS 483; Spring, 2006, 2006.

[Mer07] _____, *Inside the black box: Revisiting the bernstein-vazirani problem*, Tech. report, Cornell University, 2007.

[Mos07] Raymond Mosca, Michele; Kaye; Phillp; Laflamme, *An introduction to quantum computing*, Oxford University, 2007.

[NL05] Michael A. Nielsen and Isaac L.Chuang, *Computação quântica e informação quântica*, Bookman, 2005.

[PCG06] Renato Portugal, C. Cosme, and D. N. Gonçalves, *Algoritmos quânticos*, 1° WECIQ - Workshop Escola de Computação e Informação Quânticas, 2006.

[PLM07] Renato Portugal, Carlille Lavor, and L.R.U. Manssur, *Shors algorithm for factoring large integers*, http://arXiv.org/quant-ph/0303175v1 **1** (2007).

[Por07] Renato Portugal, *Computação quântica avançada*, 2o. Workshop-Escola de Computação e Informação Quânticas, 2007.

[Pre06] J. Preskill, *Quantum computation caltech notes*, Tech. report, www.theory.caltech.edu/people/preskill/ph229/, 2006.

[PZ+04] Xinhua Peng, Xiwen Zhu, , Ximing Fang, Mang Feng, Maili Liu, and Kelin Gao, *Spectral implementation for creating a labeled pseudopure state and the bernstein vazirani algorithm in a four-qubit nuclear magnetic resonance quantum processor*, Journal of Chemical Physics **Vol. 120. Number 8** (2004), 3579–3585.

[Sha71] D. Shanks, *Class number, a theory of factorization and general*, Proc. Symp. Pure Math **20** (1971), 415–440.

[Smi99] Steven W. Smith, *The scientist and engineer's guide to digital signal processing*, California Technical Publishing, 1999.

[Vaz07] Umesh Vazirani, *Simon's algorithm*, CS 294-2, Fall 2004, Lecture 7, September 2007.

[WC98] Colin P. Willians and Scott H. Clearwater, *Explorations in quantum computing*, The Eletronic Library of Science, 1998.