UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIAS DA COMPUTAÇÃO

Gabriel Medeiros Lopes Carneiro

Estudo de Algoritmos Quânticos

Florianópolis, SC
2022

Gabriel Medeiros Lopes Carneiro

Estudo de Algoritmos Quânticos

Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Ciências da Computação

Orientador: Eduardo Inácio Duzzioni

Florianópolis, SC
2022

Resumo

A bolsa de iniciação científica teve como foco de estudo computação quântica. Durante o período, duas linguagens de programação quântica foram estudadas, sendo elas Qiskit e Ket. Além disso, os principais algoritmos quânticos foram vistos, como, por exemplo, o algoritmo de busca de Grover, a estimativa de fase, busca de ordem, entre outros. Com os conhecimentos adquiridos também foi possível participar de um projeto de extensão relacionado a um simulador quântico.

Lista de ilustrações

Lista de Códigos

Sumário

1 Introdução

1.1 Motivação

Alguns problemas não possuem solução clássica em tempo polinomial, como a fatoração. Para vários desses, a computação quântica já se mostrou eficiente, inclusive para a fatoração, algo que pode comprometer a criptografia RSA.

1.2 Justificativas

A computação quântica ainda está crescendo, mas já mostra grande potencial para resolver problemas de otimização, logística, finanças, álgebra linear e vários outros. Grandes empresas já começaram a investir fortemente no setor, algo que faz aumentar a busca por profissionais na área. Então, estudar, entender e se adaptar ao novo modo de computação pode trazer grandes retornos num futuro relativamente próximo.

1.3 Objetivos

1.3.1 Objetivo Geral

Estudo de computação e algoritmos quânticos.

1.3.2 Objetivos Específicos

- Entender a base da computação quântica.
- Conhecer linguagens de programação quântica.
- Estudar e implementar algoritmos quânticos.

2 Computação Quântica

2.1 Breve Histórico

No início do século XX os cientistas enfrentavam um grande problema, que era explicar o comportamento da radiação emitida por um corpo negro. A solução desse problema levou ao surgimento da Mecânica Quântica, que segundo a hipótese de Max Planck "a radiação só pode ser emitida ou absorvida por um corpo negro em quantidades múltiplas inteiras de hf", em que $h \approx 6,62 \cdot 10^{-34} J \cdot s$ é a constante de Planck e f é a frequência de radiação. A quantização da energia e de outras grandezas na escala atômica foi importante para explicar uma série de outros fenômenos, como por exemplo o efeito fotoelétrico e o espectro da radiação emitida por átomos e moléculas. O desenvolvimento da Mecânica Quântica nos permitiu compreender melhor o comportamento da matéria na escala microscópica, nesse caso particular, os materiais semicondutores, que permitiram a criação do transistor. Os transistores substituíram as válvulas usadas nos primeiros computadores digitais a partir de 1955. É importante notar que a lógica usada para realizar as operações computacionais nos nossos notebooks, PCs, tablets e smartphones é a lógica booleana, ou seja, uma lógica clássica que envolve operações como AND, OR e NOT sobre os bits 0 e 1. Devido a sua grande capacidade de cálculo e armazenamento, os computadores são fundamentais para o desenvolvimento de qualquer sociedade moderna. Essa é a chamada primeira revolução quântica.

Um dos grandes impulsionadores da computação quântica foi o físico Richard Feynman, que no início da década de 80 sugeriu o uso de computadores quânticos para simular sistemas quânticos. A percepção de Feynman baseia-se no fato de que o número de configurações possíveis nos sistemas quânticos cresce de maneira exponencial com o número de entes (spins, elétrons, átomos, ...) considerados, tornando-se proibitivo para a memória dos computadores atuais guardar tanta informação mesmo para um número pequeno (< 100) de partículas.

Na década seguinte, os primeiros algoritmos quânticos começaram a surgir, dentre eles, os que mais se destacaram foram o algoritmo de busca de Grover e o de fatoração de Shor. Este último algoritmo foi provavelmente um dos grandes responsáveis pelo desenvolvimento da computação quântica, já que é capaz de encontrar os fatores primos p e q que multiplicados resultavam em um número inteiro $N = p \cdot q$ em uma escala de tempo que cresce polinomialmente com o tamanho do número N. Ou seja, a base do sistema de segurança RSA, amplamente usado para realizar transações bancárias no mundo todo, pode estar comprometida a partir da existência de computadores quânticos de larga escala.

A partir da segunda década do século XXI, não apenas a computação quântica, mas outras áreas como criptografia quântica, sensores quânticos e simulação quântica tem recebido forte atenção não apenas do setor acadêmico, mas também do setor industrial. Dessa forma, tem-se observado o rápido desenvolvimento das chamadas tecnologias quânticas, o que configura a segunda revolução quântica, uma vez que a lógica por trás dos processos é de natureza quântica.

2.2 Desenvolvimento Atual

A Computação Quântica ainda está em fase de amadurecimento, mas já mostra o seu grande potencial para resolver problemas práticos, além do algoritmo de fatoração de Shor e de busca de Grover, tais como problemas de otimização, machine learning, logística, química quântica, finanças, álgebra linear, entre outros. Nos últimos cinco anos, grandes empresas como Google, IBM, Amazon e Microsoft intensificaram ainda mais os seus investimentos no setor, além de vários governos de diversos países da América do Norte, Europa, Oceania e Ásia. Um dos grande temores em relação ao computadores quânticos está relacionado à segurança da informação, que afeta não apenas as transações bancárias, mas toda e qualquer transmissão de informação sigilosa, incluindo a militar, naturalmente. Tentativas de barrar possíveis ataques por computadores quânticos incluem a criptografia pós-quântica, que apesar do nome, se baseia em métodos criptográficos clássicos.

Para quem estiver interessado em aprender computação quântica, vale lembrar que algumas empresas disponibilizam computadores quânticos reais, simuladores, kits e linguagens para desenvolvimento de algoritmos ao público geral. Por exemplo, é possível acessar gratuitamente o kit de desenvolvimento quântico criado pela IBM (Qiskit) e rodar algoritmos em alguns dos seus computadores quânticos com poucos qubits.

2.3 O que é um qubit?

O qubit (**qu**antum + **bit**) é um bit quântico. O bit clássico sempre está em um dos possíveis estados 0 ou 1, já um qubit pode estar em ambas configurações simultaneamente. Chamamos esse fenômeno de superposição. Para representar um qubit utilizamos a notação Dirac ou "braket":

$$|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix} = a\,|0\rangle + b\,|1\rangle$$

em que a e b são amplitudes de probabilidade (números complexos), de modo que

$|a|^2$ representa a probabilidade de após uma medida encontrar o sistema no estado $|0\rangle$

$|b|^2$ representa a probabilidade de após uma medida encontrar o sistema no estado $|1\rangle$

Como a probabilidade total deve somar 100%, temos que a condição de normalização para o estado $|\psi\rangle$ é $|a|^2 + |b|^2 = 1$.

2.3.1 Esfera de Bloch

Os estados de um qubit podem ser representados por meio de pontos em uma superfície esférica de raio unitário, utilizando o sistema de coordenadas esféricas. Para isso, é preciso parametrizar o estado do qubit $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$ da seguinte forma

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \text{ tal que } \theta \in [0,\pi], \phi \in [0,2\pi)$$

Agora, utilizando θ e ϕ no sistemas de coordenadas esféricas, tem-se a Esfera de Bloch. Todos os estados acessíveis a um qubit podem ser representados utilizando-se Figura 1.

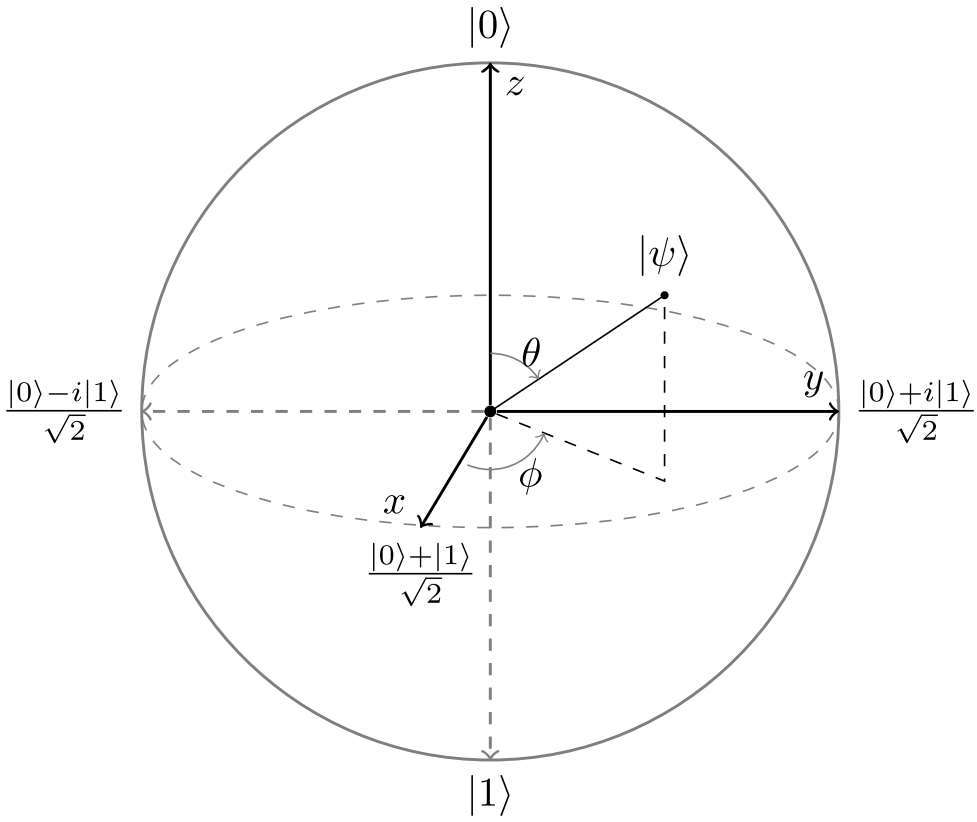

Figura 1 – Representação de um qubit na Esfera de Bloch.

2.3.2 Representação de 2 ou mais qubits

Existem diversas formas de se representar um sistema de 2 qubits, seguem algumas equivalências:

$$|\psi_0\rangle \otimes |\psi_1\rangle = |\psi_0\rangle\,|\psi_1\rangle = |\psi_0\psi_1\rangle$$

em que $\otimes$ é produto tensorial de ψ_0 com ψ_1. Seja

$$|\psi_0\rangle \otimes |\psi_1\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{bmatrix}$$

De forma análoga, é possível representar sistemas de n qubits como

$$|\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle = |\psi_0\rangle\,|\psi_1\rangle \ldots |\psi_n\rangle = |\psi_0\psi_1 \ldots \psi_n\rangle$$

Como será mostrado na seção 2.7, a superposição de estados desse tipo pode levar ao emaranhamento.

2.4 Etapas de um Algoritmo Quântico

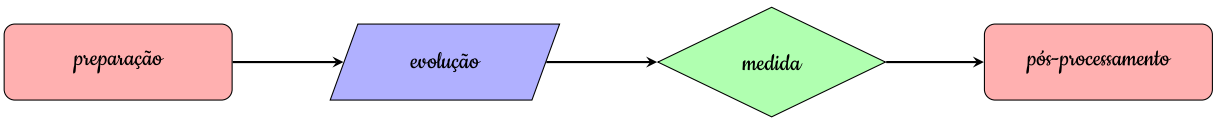

Figura 2 – Etapas básicas de um algoritmo quântico

De forma geral, é possível separar um algoritmo quântico em quatro etapas, como mostra a Figura 2.

1. **Preparação**: aqui cada qubit é inicializado em algum estado, geralmente em $|0\rangle$.
2. **Evolução**: nessa parte o algoritmo é de fato aplicado, através das portas lógicas quânticas.
3. **Medida**: após a aplicação das portas, é necessário medir os qubits, para se ter o resultado do circuito.
4. **Pós-processamento**: finalmente, nessa etapa o resultado obtido deve ser interpretado de acordo com o contexto.

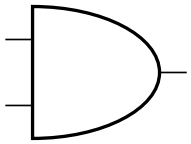

Figura 3 – Representação da porta AND.

2.5 Comparação com Computação Clássica

2.5.1 Entradas e Saídas

- **Clássica**: portas podem ter diferentes números de bits entrando e saindo.

Exemplo
A porta AND possui dois ou mais bits de entrada e apenas um de saída.

- **Quântica**: portas possuem mesmo número de qubits na entrada e na saída.

2.5.2 Reversibilidade

- **Clássica**: a maioria das portas clássicas não são reversíveis, isto é, dado uma saída não conseguimos identificar quais foram as entradas.

Exemplo
Na porta OR de dois bits podemos obter 1 como saída em três casos.

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Sabendo que a saída foi 1 não é possível identificar qual/quais bits eram 1.

- **Quântica**: seus circuitos são reversíveis, isso ocorre, pois, seus operadores são unitários.

Observação
Embora a evolução temporal seja reversível durante o processamento da informação no circuito quântico, a medição dos qubits é um processo irreversível.

2.6 Portas Lógicas Quânticas

As portas lógicas quânticas são operações unitárias que ao atuar em um estado inicial levam para outro estado final, ou seja, funcionam como rotações na esfera de Bloch. A seguir, alguns exemplos de portas lógicas quânticas que atuam sobre um qubit.

2.6.1 Porta X

Essa porta é o equivalente a porta NOT da computação clássica.
Matriz

$$X = \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Comportamento

$$\begin{aligned} X|0\rangle &= |1\rangle \\ X|1\rangle &= |0\rangle \end{aligned}$$

Símbolo

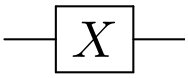

Figura 4 – Representação da porta X.

ou ainda

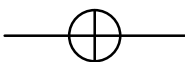

Figura 5 – Outra representação da porta X.

2.6.2 Porta Y

Matriz

$$Y = \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Comportamento

$$\begin{aligned} Y|0\rangle &= i|1\rangle \\ Y|1\rangle &= -i|0\rangle \end{aligned}$$

Símbolo

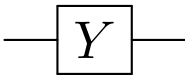

Figura 6 – Representação da porta Y.

2.6.3 Porta Z

A porta Z introduz uma fase relativa de π entre os estados da base computacional.
Matriz

$$Z = \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Comportamento

$$\begin{aligned} Z\left|0\right\rangle &= \left|0\right\rangle \\ Z\left|1\right\rangle &= -\left|1\right\rangle \end{aligned}$$

Símbolo

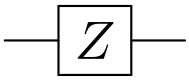

Figura 7 – Representação da porta Z.

2.6.4 Porta Hadamard

Essa porta gera uma superposição dos estados da base computacional.
Matriz

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

Comportamento

$$\begin{aligned} H\left|0\right\rangle &= \frac{1}{\sqrt{2}}\left(\left|0\right\rangle + \left|1\right\rangle\right) &= \left|+\right\rangle \\ H\left|1\right\rangle &= \frac{1}{\sqrt{2}}\left(\left|0\right\rangle - \left|1\right\rangle\right) &= \left|-\right\rangle \end{aligned}$$

Símbolo

Figura 8 – Representação da porta H.

2.6.5 Portas Controladas

Para se fazer computação quântica universal, ou seja, realizar todas as transformações unitárias desejadas entre os qubits de entrada e saída em um algoritmo, é necessário realizar operações que façam dois ou mais qubits interagirem entre si. Tais portas podem envolver um qubit de controle e o outro como alvo, sendo possível generalizá-la para múltiplos qubits de controle e de alvo. Segue o exemplo para porta controlada X, ou CNOT, com um controle e um alvo.

Matriz

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Comportamento

$$\begin{aligned} \text{CNOT}\,|00\rangle &= |00\rangle \\ \text{CNOT}\,|01\rangle &= |01\rangle \\ \text{CNOT}\,|10\rangle &= |11\rangle \\ \text{CNOT}\,|11\rangle &= |10\rangle \end{aligned}$$

Símbolo

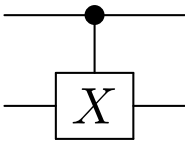

Figura 9 – Representação da porta X-controlada.

ou ainda

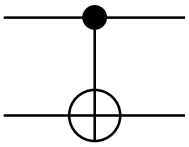

Figura 10 – Outra representação da porta X-controlada.

2.7 Emaranhamento

Estados emaranhados são aqueles que não podem ser escritos como produto tensorial de estados de 1 qubit, ou seja, não é possível separá-los. Os mais conhecidos são os estados de Bell, os quais envolvem apenas 2 qubits, sendo dados por:

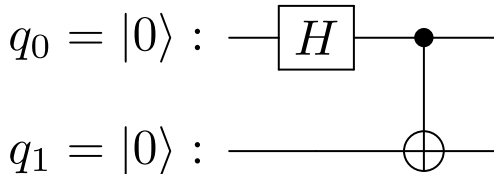

Figura 11 – Circuito para criar um estado de Bell.

$$
\begin{aligned}
|\beta_{00}\rangle &= |\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
|\beta_{01}\rangle &= |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
|\beta_{10}\rangle &= |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\
|\beta_{11}\rangle &= |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)
\end{aligned}
$$

Os estados emaranhados são apontados como sendo os responsáveis por fazer não apenas a computação quântica mais veloz do que a computação clássica, mas também permitem aumentar a precisão de medidas de observáveis físicos e realizar comunicação de forma segura.

2.7.1 Criando um Estado de Bell

Com o conceito de emaranhamento explicado, resta saber como criá-lo. Como exemplo, o estado $|\beta_{00}\rangle$ será criado, na Figura 11 temos o circuito para isso e em Código 1 o código para o mesmo usando Ket.

```
q0, q1 = quant(2)    # cria dois qubits
H(q0)                # aplica a porta de Hadamard no qubit 0
ctrl(q0, X, q1)      # aplica a porta X no qubit 1, com o qubit 0 como controle
```

Código 1: Criando um estado de Bell em Ket.

Seja $|\psi\rangle = q_0 \otimes q_1$. Após a aplicação da porta de Hadamard, teremos $q_0 = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, conforme visto anteriormente. Logo,

$$
\begin{aligned}
|\psi\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \\
&= \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)
\end{aligned}
$$

Na sequência, temos uma porta CNOT, com o qubit 0 como controle e o qubit 1 como alvo. Gerando a seguinte situação

$$
\begin{aligned}
|\psi\rangle &= \mathrm{CNOT}\left[\frac{1}{\sqrt{2}}\left(|00\rangle + |10\rangle\right)\right] \\
&= \frac{1}{\sqrt{2}}\left(\mathrm{CNOT}\,|00\rangle + \mathrm{CNOT}\,|10\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\
&= |\beta_{00}\rangle
\end{aligned}
$$

Portanto, com apenas duas portas é possível gerar uma situação de emaranhamento.

3 Desenvolvimento

3.1 Qiskit

Inicialmente, foi planejado utilizar o material didático fornecido pelo Qiskit para o aprendizado de conceitos básicos. Qiskit é uma ferramenta de desenvolvimento quântico criado pela IBM que permite o uso de simuladores e computadores quânticos no nível de pulsos, circuitos e outros tipos de aplicações. Atualmente, é a ferramenta mais popular.

Com o decorrer dos estudos, alguns problemas foram identificados. O primeiro foi referente aos cursos introdutórios, os quais não apresentavam aprofundamento necessário para se compreender os assuntos, em geral, faltavam mais explicações sobre conceitos físicos e matemáticos por trás. Esse problema começou a ser parcialmente resolvido com a atualização dos cursos no final de 2021, mas as mudanças chegavam a passos lentos, muitas vezes as alterações feitas já estavam defasadas em relação ao desenvolvimento atual da ferramenta. Isso se deve ao fato da ferramenta ser de código aberto e possuir uma comunidade muito ativa, recebendo várias alterações e melhorias com frequência. Além disso, por se tratar de uma tecnologia relativamente nova e devido a essas frequentes atualizações, a ferramenta não possui um longo período de estabilidade entre versões, algo que torna códigos obsoletos rapidamente.

Outro problema encontrado foi com o acesso aos computadores quânticos da IBM. Mesmo com a licença de estudante, os computadores disponíveis são poucos e possuem poucos qubits, tornando inviável usá-los para experimentos que exijam mais qubits. Recentemente, a IBM retirou o computador que possuía mais qubits da licença de estudantes, dificultando ainda mais os estudos. A alternativa, portanto, foi usar os simuladores quânticos disponíveis, mas logo se tornou enviável devido às constantes atualizações.

3.2 Material Básico

Após os problemas enfrentados com o material do Qiskit, passou-se a usar[POLLACHINI, 2018] como fonte de estudos. Esse material foi criado por POLLACHINI durante seu TCC em Engenharia Eletrônica e facilitou, e muito, o aprendizado. Ao ler o material passa-se por uma revisão de algebra linear, com foco em computação quântica, para então se ter uma introdução à mecânica quântica e computação quântica, algo que os tutoriais do Qiskit não fornecem.

3.3 Tópicos Avançados

Após consolidar uma base suficientemente forte com a leitura de[POLLACHINI, 2018], os livros[NIELSEN; CHUANG, 2010; WONG, 2022] foram usados para aprofundar o conhecimento em tópicos mais avançados. O primeiro é um clássico no assunto, mostrando conceitos fundamentais, funcionamento de computadores quânticos, principais algoritmos quânticos e mais. Já o segundo é mais recente, trazendo explicação menos densa do conteúdo ele passa por conceitos fundamentais da computação clássica e trazendo alguns comparativos entre as diferentes formas de computação.

3.4 Ket

Ket é uma linguagem de programação embarcada em Python, projetada para facilitar o desenvolvimento de aplicações híbridas clássica-quântica. Ket é um projeto de código aberto derivado do trabalho de mestrado de ROSA no Programa de Pós-Graduação em Ciência da Computação (PPGCC) da UFSC.

Essa linguagem fornece uma grande facilidade na implementação de portas lógicas quânticas, principalmente nas controladas. Fazendo uma comparação com o Qiskit, ela é mais estável, porém não possui tantos recursos, visto que apenas o criador desenvolve a linguagem.

3.5 QuBOX

Com o conteúdo aprendido, foi possível participar do projeto de extensão "Simulador quântico portátil para a linguagem de programação Ket com fins educacionais e de pesquisa", registrado no SIGPEX sob o número 202123813 e feito em uma parceria entre a startup Quantuloop e o Grupo de Computação Quântica — UFSC. O projeto tem o objetivo de promover a computação quântica e auxiliar no desenvolvimento de novos algoritmos, métodos e aplicações quânticas. O público alvo são alunos e professores da educação básica ao ensino superior interessados em pesquisar, aprender e ensinar computação quântica. Além de todos os interessados em computação quântica.

Através desse projeto é possível programar em um simulador quântico de até 30 qubits de maneira gratuita, através da linguagem Ket e do simulador QuBOX. Para o projeto, foi feito um resumo sobre Computação Quântica e informações e tutoriais sobre algoritmos quânticos.

4 Algoritmos Quânticos

Alguns dos principais algoritmos quânticos foram estudados, esses algoritmos serviram de base para vários outros, por isso foram escolhidos.

4.1 Algoritmo de Busca de Grover

Aqui será abordado como implementar o algoritmo quântico de Grover para resolver o problema da busca em base de dados desordenada, o qual é mais eficiente que as soluções clássicas.

4.1.1 Problema

O problema pode ser resumido em encontrar um determinado elemento dado uma lista desordenada de tamanho 2^n. Para isso, deve ser criada uma função booleana $f : \{0,1\}^n \to \{0,1\}$, a qual só sinalize '1' para o elemento desejado.

$$f(x) = \begin{cases} 0, x \neq x_0 \\ 1, x = x_0 \end{cases}$$

Observação

Cada elemento da lista será codificado em um estado da base computacional $|i\rangle, i = \{0, \ldots, 2^n - 1\}$.

4.1.2 Oráculo

Um oráculo faz o mesmo que a função booleana descrita anteriormente. Existem dois tipos de oráculos, XOR e os de fase, nesse problema será utilizado o de fase.

4.1.2.1 Oráculo de Fase

O oráculo de fase introduzirá uma fase de π, ou seja, multiplicará por -1.

- Exemplo: dado $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Se o oráculo for usado para marcar o estado $|1\rangle$, o resultado será $O(|\psi\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

4.1.3 Algoritmo

O primeiro passo consiste em aplicar a porta de Hadamard em todos os qubits para se conseguir uma superposição com pesos iguais.

$$H^{\otimes n} |0\rangle^{\otimes n}.$$

Após isso é preciso aplicar G (operador de Grover) k vezes, em que G representa a seguinte sequência de passos:

1. Aplicar o oráculo de fase para o estado desejado;
2. Aplicar a porta de Hadamard em todos os qubits;
3. Aplicar o operador $2 |0\rangle \langle 0| - I$; Na notação utilizada aqui $|0\rangle$ representa $|0\rangle^{\otimes n}$.
4. Aplicar a porta de Hadamard em todos os qubits.

4.1.3.1 Notação Auxiliar

$$\mathbb{B}_n : \text{conjunto de todas as palavras de n bits.}$$
$$\mathbb{M} : \text{conjunto de todos itens desejados.}$$
$$N = 2^n \begin{cases} n : \text{número de qubits.} \\ N : \text{número de itens.} \end{cases}$$
$$M : \text{número de itens desejados (usaremos } M = 1).$$
$$|\alpha\rangle \coloneqq \sum_{\substack{x \in \mathbb{B}_n \\ f(x)=0}} \frac{|x\rangle}{\sqrt{N-M}}$$
$$|\beta\rangle \coloneqq \sum_{\substack{x \in \mathbb{B}_n \\ f(x)=1}} \frac{|x\rangle}{\sqrt{M}} : \text{itens desejados.}$$
$$S \coloneqq \text{span}_{\mathbb{R}}\{|\alpha\rangle, |\beta\rangle\} : \text{espaço vetorial gerado por } |\alpha\rangle \text{ e } |\beta\rangle.$$

4.1.3.2 Difusor

$$
\begin{aligned}
2\left|0\right\rangle\left\langle 0\right| - I &= 2 \cdot \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1} \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}_{1\times n} - \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{n\times n} \\
&= \begin{bmatrix} 2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{n\times n} - \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{n\times n} \\
&= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & -1 \end{bmatrix}_{n\times n} \\
&= \left|0\ldots0\right\rangle\left\langle 0\ldots0\right| - \left|0\ldots1\right\rangle\left\langle 0\ldots1\right| - \cdots - \left|1\ldots1\right\rangle\left\langle 1\ldots1\right|
\end{aligned}
$$

Usando o conceito de fase global, é possível escrever o resultado de outra forma, sendo ela:

$$-\left|0\ldots0\right\rangle\left\langle 0\ldots0\right| + \left|0\ldots1\right\rangle\left\langle 0\ldots1\right| + \cdots + \left|1\ldots1\right\rangle\left\langle 1\ldots1\right|$$

Para obter esse resultado, basta usar o oráculo de fase visto anteriormente e usá-lo para marcar o estado $\left|0\right\rangle$.

4.1.3.3 Primeira Aplicação de G

Antes de fazer as aplicações, temos:

$$
\begin{aligned}
\left|\psi_0\right\rangle &= \left|+\right\rangle^{\otimes n} \\
&= \textstyle\sum_{x\in\mathbb{B}_n} \dfrac{\left|x\right\rangle}{\sqrt{N}} \\
&= \textstyle\sum_{\substack{x\in\mathbb{B}_n \\ x\neq x_0}} \dfrac{\left|x\right\rangle}{\sqrt{N}} + \dfrac{\left|x_0\right\rangle}{\sqrt{N}} \\
&= \dfrac{\sqrt{N-1}}{\sqrt{N}} \textstyle\sum_{\substack{x\in\mathbb{B}_n \\ x\neq x_0}} \dfrac{\left|x\right\rangle}{\sqrt{N-1}} + \dfrac{\left|x_0\right\rangle}{\sqrt{N}} \\
&= \dfrac{\sqrt{N-1}}{\sqrt{N}}\left|\alpha\right\rangle + \dfrac{1}{\sqrt{N}}\left|\beta\right\rangle
\end{aligned}
$$

Após a aplicação do oráculo (Figura 12):

$$
\begin{aligned}
\left|\psi_1\right\rangle &= O_F\left|\psi_0\right\rangle \\
&= \dfrac{\sqrt{N-1}}{\sqrt{N}}\left|\alpha\right\rangle - \dfrac{1}{\sqrt{N}}\left|\beta\right\rangle
\end{aligned}
$$

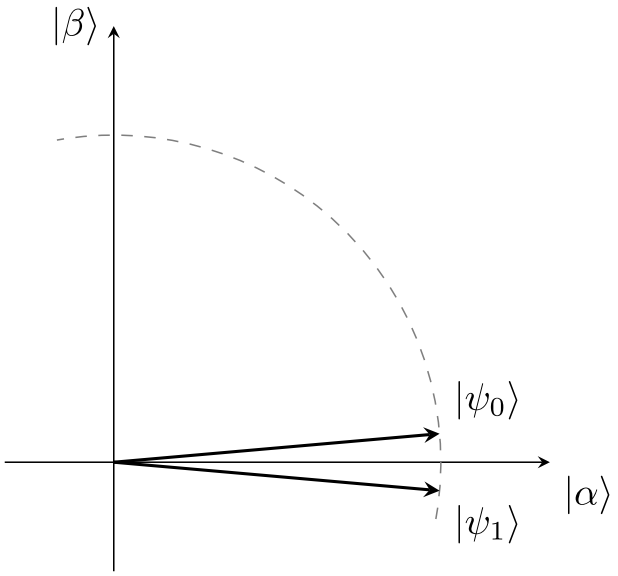

Figura 12 – Aplicação do oráculo de fase equivale a uma reflexão em relação ao eixo $|\alpha\rangle$[POLLACHINI, 2018].

Após a aplicação de $2\left|\psi_0\right\rangle\left\langle\psi_0\right| - I$, Figura 13:

$$\begin{aligned} \left|\psi_2\right\rangle &= \left(2\left|\psi_0\right\rangle\left\langle\psi_0\right| - I\right)\left|\psi_1\right\rangle \\ &= 2\left\langle\psi_0 \mid \psi_1\right\rangle\left|\psi_0\right\rangle - \left|\psi_1\right\rangle \end{aligned}$$

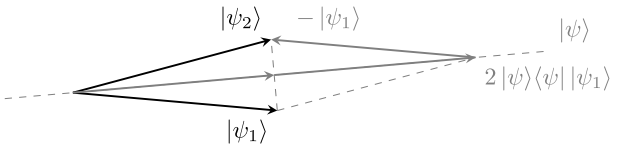

Figura 13 – Aplicação do operador $2\left|\psi_0\right\rangle\left\langle\psi_0\right| - I$ equivale a uma reflexão em relação à reta determinada pelo vetor $|\psi_0\rangle$[POLLACHINI, 2018].

Com isso, após uma reflexão sobre o estado $|\psi_0\rangle$, tem-se o resultado da Figura 14.

4.1.4 Aplicações Sucessivas de G

A cada repetição tem-se uma rotação no sentido anti-horário, logo, o vetor estará se afastando de $|\alpha\rangle$ e se aproximando de $|\beta\rangle$ (item desejado), conforme a Figura 15.

4.1.4.1 Número de Aplicações

O número de aplicações necessárias é dado por:

$$k = \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}}$$

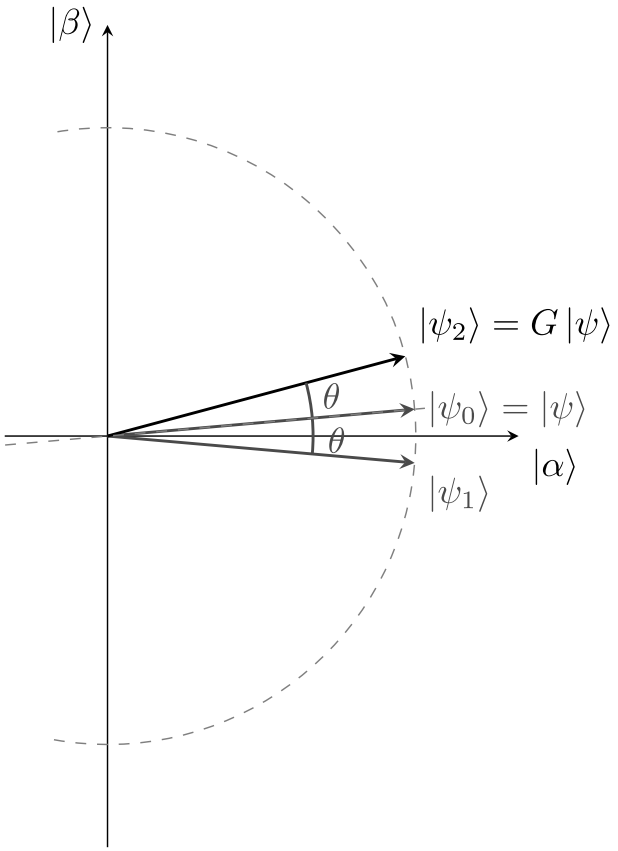

Figura 14 – Aplicação do operador G. O efeito corresponde à rotação do vetor por um ângulo θ no sentido anti-horário[POLLACHINI, 2018].

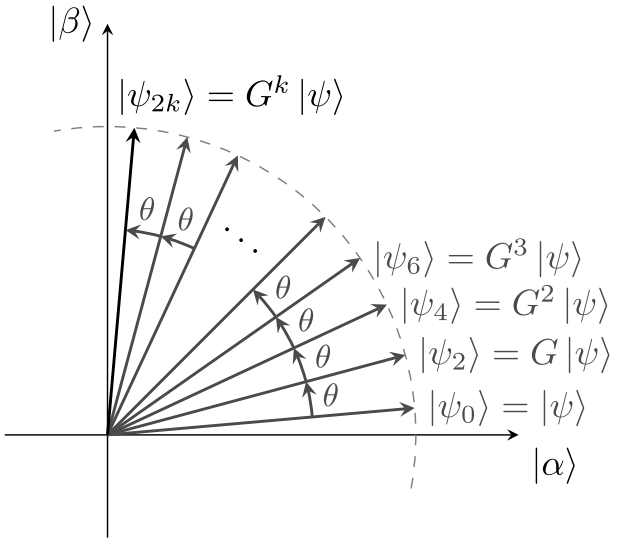

Figura 15 – Aplicações sucessivas do operador G.

4.1.5 Implementação

Com a teoria explicada, é possível implementar o algoritmo de Grover usando Ket. Aqui, cada estado da base computacional será representado na base decimal, $|i\rangle, i = 0, \ldots, 2^n - 1$. Logo, é necessário passar um inteiro maior ou igual a 0 que indicará qual

desses estados será marcado, também será preciso especificar quantos bits serão simulados.

Primeiro é preciso importar o necessário (Código 2).

```
from math import sqrt, pi
from ket import *
```

Código 2: Importando bibliotecas.

Para facilitar, o algoritmo vai ser dividido em quatro partes:

1. Oráculo de fase;
2. Difusor;
3. Operador de Grover;
4. Função principal `grover()`, a qual será chamada e realizará todas as operações necessárias.

4.1.5.1 Oráculo de fase

```
def phase_oracle(qubits: quant, state: int):
    with around(I if state&1 else X, qubits[-1]):
        ctrl(qubits[:-1], Z, qubits[-1], on_state=state>>1)
```

Código 3: Oráculo de fase.

O que está sendo feito no Código 3 é receber os qubits e o estado desejado como argumentos e aplicando a porta Z controlada. Se os qubits estiverem no estado `state` a porta `Z` é aplicada no último qubit (`qubit[-1]`). Como essa operação controlada aplica a fase -1 apenas quando os qubits estão no estado $|1\rangle$, a porta `flipc` é usada para permutar o estado desejado (`state`) para o estado de controle ($|1\cdots1\rangle$). Com o `with around` é garantido que a permutação será desfeita após a aplicação da porta controlada.

4.1.5.2 Difusor

O difusor (Código 4) é muito semelhante ao código do oráculo, porém nele a porta de hadamard é aplicada antes e depois da porta controlada, e o estado a ser marcado é $|0\rangle$.

```
def grover_diffuser(qubits: quant):
    with around(H, qubits):
        phase_oracle(qubits, 0)
```

Código 4: Difusor

4.1.5.3 Operador de Grover

Agora partindo para o operador de Grover, tem-se o Código 5, consistindo apenas da chamada do oráculo e depois do difusor.

```
def grover_operator(qubits: quant, state: int):
    phase_oracle(qubits, state)
    grover_diffuser(qubits)
```

Código 5: Operador de Grover.

4.1.5.4 Função Principal

Para finalizar, a função principal, a qual irá criar os qubits e definir quantas vezes será preciso aplicar o operador de Grover.

```
def grover(state: int, num_qubits: int) -> int:
    qubits = quant(num_qubits)
    entries = 2**num_qubits
    steps = int((pi / 4) * sqrt(entries))

    H(qubits)

    for _ in range(steps):
        grover_operator(qubits, state)

    return measure(qubits).value
```

Código 6: Função principal do algoritmo.

Após todas as operações, é feito a medida e retorna-se o valor obtido. Como se trata de um algoritmo probabilístico, nem sempre o estado desejado será o resultado, porém é esperado que tenha alta taxa de sucesso. Com tudo pronto pode-se chamar a função `grover()`.

4.1.5.5 Saída

O Código 7 foi criado para testar o funcionamento do algoritmo e se obteve a saída presente no Código 8.

```
from random import randrange
num_qubits = 10
for state in [randrange(2**num_qubits) for _ in range(10)]:
    print('Procurando estado', state, '...', end=' ')
    print('Estado medido:', grover(state, num_qubits))
```

Código 7: Função principal do algoritmo.

```
Procurando estado 882 ... Estado medido: 882
Procurando estado 1013 ... Estado medido: 1013
Procurando estado 557 ... Estado medido: 557
Procurando estado 490 ... Estado medido: 490
Procurando estado 163 ... Estado medido: 163
Procurando estado 429 ... Estado medido: 429
Procurando estado 300 ... Estado medido: 300
Procurando estado 184 ... Estado medido: 184
Procurando estado 394 ... Estado medido: 394
Procurando estado 763 ... Estado medido: 763
```

Código 8: Saída dos testes.

Referências

NIELSEN, M. A.; CHUANG, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. [S.l.]: Cambridge University Press, 2010.
Citado na seção 3.3.

POLLACHINI, G. G. *Computação Quântica: Uma abordagem para estudantes de graduação em Ciências Exatas*. [S.l.: s.n.], 2018.
Citado nas seções (document), 3.2, 3.3, 12, 13 e 14.

ROSA, E. C. R. da. *Ket Quantum Programming*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2021. Disponível em: <https://repositorio.ufsc.br/handle/123456789/229874>.
Citado na seção 3.4.

WONG, T. G. *Introduction to Classical and Quantum Computing*. [S.l.]: Rooted Grove, 2022.
Citado na seção 3.3.