Universidade Federal Fluminense
Instituto de Computação

Alexandre Santiago de Abreu
Matheus Manzoli Ferreira

# Simulador Numérico para o Algoritmo Quântico de Distinção de Elementos

Niterói
2014

ALEXANDRE SANTIAGO DE ABREU
MATHEUS MANZOLI FERREIRA

# SIMULADOR NUMÉRICO PARA O ALGORITMO QUÂNTICO DE DISTINÇÃO DE ELEMENTOS

Trabalho de Conclusão de Curso Apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. LUIS ANTONIO BRASIL KOWADA, D.Sc.
Coorientador: Prof. FRANKLIN DE LIMA MARQUEZINO, D.Sc.

Niterói
2014

**Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF**

A162 Abreu, Alexandre Santiago de
Simulador numérico para o algoritmo quântico de distinção de elementos / Alexandre Santiago de Abreu, Matheus Manzoli Ferreira. – Niterói, RJ : [s.n.], 2014.
41 f.

Trabalho (Conclusão de Curso) – Departamento de Computação, Universidade Federal Fluminense, 2014.
Orientadores: Luis Antonio Brasil Kowada, Franklin de Lima Marquezino.

1. Algoritmo. 2. Simulação por computador. 3. Computação quântica. I. Ferreira, Matheus Manzoli. II. Título.

CDD 005.136

ALEXANDRE SANTIAGO DE ABREU
MATHEUS MANZOLI FERREIRA

# SIMULADOR NUMÉRICO PARA O ALGORITMO QUÂNTICO DE DISTINÇÃO DE ELEMENTOS

Trabalho de Conclusão de Curso Apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense para obtenção do Grau de Bacharel em Ciência da Computação.

Aprovada em 01 Dezembro de 2014.

BANCA EXAMINADORA

---

Prof. LUIS ANTONIO BRASIL KOWADA, D.Sc. - Orientador
UNIVERSIDADE FEDERAL FLUMINENSE

---

Prof. FRANKLIN DE LIMA MARQUEZINO, D.Sc. - Coorientador
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

---

Prof. ERNESTO FAGUNDES GALVÃO, Ph.D.
UNIVERSIDADE FEDERAL FLUMINENSE

---

Prof. VALENTIN SISKO, D.Sc.
UNIVERSIDADE FEDERAL FLUMINENSE

Niterói
2014

Dedicamos este trabalho às nossas famílias, amigos e professores pelo apoio incondicional que nos deram ao longo de todo este tempo. Para sempre estarão em nossos corações.

# Agradecimentos

Eu, Alexandre Santiago de Abreu, primeiramente agradeço a Deus por toda a perseverança que me foi dada para conseguir trilhar este árduo caminho que é a graduação.

Agradeço à minha família por cada batalha que enfrentaram para que eu pudesse ter uma educação de qualidade, por todo o apoio que me deram e toda a compreensão que tiveram comigo em alguns momentos.

Agradeço aos meus orientadores por me colocarem sempre direcionado para bons caminhos, por me incentivarem a sempre buscar mais conhecimento e por terem sido grandes amigos ao longo desta minha jornada.

Agradeço ao meu colega de projeto por seu companheirismo ao longo de toda a graduação.

Por fim agradeço a todos professores e amigos que ao longo de todos esses anos me ensinaram muitas vezes conhecimentos que vão além daqueles encontrados em literaturas.

Eu, Matheus Manzoli Ferreira, agradeço a Deus, pois Nele pude depositar toda minha fé.

Agradeço aos meus pais por tudo que fizeram por mim ao longo de todos esses anos, sempre batalhando para que eu tivesse a melhor educação e as melhores oportunidades.

Agradeço à minha irmã e aos meus familiares por todo apoio que me deram durante esses anos.

Agradeço ao meu colega de trabalho pela confiança em mim, por ter me convidado para fazer parte deste projeto, e por toda ajuda durante o curso.

Aos meus orientadores por terem me auxiliado da melhor maneira durante a monografia.

Agradeço aos meus amigos e professores pela amizade que tivemos nesses anos.

Agradeço aos professores Ernesto Galvão e Valentin Sisko por terem aceitado fazer parte da banca.

# Resumo

O problema de distinção de elementos consiste em descobrir se em uma lista de entrada todos seus elementos são distintos. Classicamente, para o caso em que existam dois elementos iguais em uma lista de $N$ elementos, são necessários pelo menos $\Omega(N)$ passos para solucionar este problema. Utilizando o Bucket sort este problema é solucionado em $O(N)$ passos, sendo assim, o melhor algoritmo para este caso visto que este resultado já é ótimo. No paradigma quântico, Aaronson mostrou que para solucionar esta questão são necessários ao menos $\Omega(N^{2/3})$ passos, desta forma, o algoritmo de distinção de elementos proposto por Ambainis é ótimo, já que possui complexidade $O(N^{2/3})$. Tal algoritmo faz uso de importantes conceitos e técnicas da computação quântica, tais como caminhada quântica e busca quântica em um grafo de Johnson, de tal modo que faz com que este algoritmo seja muito importante neste paradigma. Por possuir superposições de estados e cálculos não-triviais, além de estados que crescem de maneira exponencial, estudar de maneira analítica este algoritmo se torna impraticável e o uso de ferramentas apropriadas se torna necessário. Devido ao fato de não existir um computador quântico de propósito geral, para se estudar e compreender o comportamento de algoritmos quânticos em determinadas situações, são necessárias ferramentas de simulação. Por executarem em máquinas clássicas, estes simuladores consomem memória e tempo de execução que crescem exponencialmente. Foi desenvolvido um simulador, de propósito educacional, para o algoritmo de Ambainis, pois até onde os autores tem conhecimento, não existe nenhum simulador que tenha o foco sobre este algoritmo tão importante. Este software pode trabalhar em ambientes Windows e Linux, sem grandes diferenças de desempenho. O código fonte é aberto, permitindo que qualquer usuário que tenha conhecimentos sobre programação possa modificá-lo, se tornando facilmente modificável para uso profissional.

**Palavras-chave**: algoritmos quânticos, simulação, caminhadas quânticas, distinção de elementos, computação quântica educacional.

# Abstract

The problem of element distinctness consists in finding out if all elements of a input list are distinct. Classically, for the case where there are two equal elements in a list of $N$ elements, at least $\Omega(N)$ steps are needed to solve this problem. Using the Bucket sort algorithm this problem is solved in $O(N)$ steps, thus being the best algorithm for this case since this result is already optimal. In the quantum paradigm, Aaronson showed that to solve this issue at least $\Omega(N^{2/3})$ steps are needed, thus the algorithm of element distinctness proposed by Ambainis is optimal, as it has complexity $O(N^{2/3})$. This algorithm makes use of important concepts and techniques of quantum computing, such as quantum walks and quantum search in a Johnson graph, so that makes this algorithm very important in this paradigm. As the systems states are in superposition and the number of states grows exponentially, it is impratical to analyze this algorithm analytically. Because there is not a general purpose quantum computer, to study and understand the behavior of quantum algorithm in certain situations, simulation tools are required. Running in classic machines, these simulators consume memory and runtime that grows exponentially. An educational purpose simulator for Ambainis algorithm was developed, until the limits of the authors knowledge, there is no simulator for this important algorithm. This software works on Windows and Linux environments, without major performance differences. The source code is open-source, allowing any user with knowledge of programming to modify it.

**Keywords**: quantum algorithms, simulations, quantum walks, element distinctness, quantum computing education.

# Sumário

# Lista de Tabelas e Figuras

# Lista de Siglas e Abreviaturas

| | |
|---|---|
| $\langle . \vert$ | Representação de um vetor dual em notação de Dirac, também conhecido como Bra. |
| $\langle . \vert \vert . \rangle$ | Representação de um produto interno em notação de Dirac, também podendo ser escrito como $\langle . \vert . \rangle$. |
| $\vert . \rangle$ | Representação de um vetor em notação de Dirac, também conhecido como Ket. |
| $\vert . \rangle \langle . \vert$ | Representação de um produto externo em notação de Dirac. |
| $\vert . \rangle \otimes \vert . \rangle$ | Representação de um produto tensorial em notação de Dirac, que também pode ser abreviado como $\vert .. \rangle$, $\vert ., . \rangle$ ou ainda $\vert . \rangle \vert . \rangle$, desde que esteja claro pelo contexto. |
| $\mathcal{H}$ | Espaço de Hilbert. |
| $\otimes$ | Operador de produto tensorial. |
| [X] | Notação para indicar que se trata de um conjunto do tipo $[X] = \{1, 2, 3, ..., X\}$. |

# 1 Introdução

A construção de um computador quântico de propósito geral é um grande desafio nos dias de hoje, visto que a tecnologia atual não possibilita o isolamento e controle individual de um grande número de partículas quânticas. Não existe um consenso entre os especialistas da área sobre qual será a tecnologia mais adequada para que se possa controlar uma grande quantidade de *qubits*[1] por tempo necessário para que sejam executadas computações úteis. Apesar disso, o paradigma quântico possui algoritmos e conceitos de estudo interessantes que, se compreendidos, podem ser úteis para que sejam desenvolvidos algoritmos quânticos que superam seus correspondentes clássicos. Simulações de algoritmos quânticos em máquinas clássicas têm sido objeto de estudo de diversos autores [5, 7, 9, 14, 20, 21] como ferramenta para a compreensão dos importantes conceitos da computação quântica, visando dois objetivos principais: (i) prover ferramentas educacionais para aprender como os algoritmos quânticos se comportariam em diferentes situações, e (ii) estudar determinados aspectos dos algoritmos onde a obtenção de resultados analíticos é bastante difícil.

Um dos grandes problemas na simulação de algoritmos quânticos em computadores clássicos é que, de forma geral, tais simulações possuem complexidade exponencial, tanto em tempo quanto em memória. Apesar disso, muitas ferramentas de simulação têm sido desenvolvidas tendo como foco diferentes formalismos da computação quântica, tais como: circuitos quânticos [5, 9], formalismo quântico estabilizador [6], caminhadas quânticas em grafos [14], simulação de Hamilton [8], entre outros. A utilização dessas ferramentas permite compreender o poder e as limitações da computação quântica, além de serem muito úteis no desenvolvimento de circuitos e algoritmos quânticos.

O algoritmo de distinção de elementos, proposto por Ambainis [4], faz uso de importantes conceitos e técnicas da computação quântica, tais como caminhadas quânticas e busca quântica em grafos. A execução deste algoritmo envolve vários cálculos e superposições de estados que não são triviais, tornando a abordagem analítica para o estudo da evolução destes estados inviável até mesmo para listas de entrada não muito grandes. Além do mais, o algoritmo de Ambainis baseia-se em caminhadas quânticas sobre um tipo de grafo particular, conhecido como grafo de Johnson, tornando inadequada a simulação deste algoritmo em programas que simulam caminhadas quânticas, como o QWalk [13, 14], por exemplo.

Neste trabalho foi desenvolvido um simulador numérico para o algoritmo de distinção de elementos. Até onde os autores têm conhecimento, não há outro simulador para

[1] O termo qubit, ou q-bit, vem do inglês quantum bit, ou seja, bit quântico

o algoritmo de Ambainis disponível na literatura. A proposta principal do software é servir como um instrumento educacional, simulando este importante algoritmo e mostrando a evolução do mesmo ao longo de cada passo. O código fonte do simulador encontra-se gratuitamente disponível sob a licença GNU GPL, e pode facilmente ser estendido para outras propostas, tais como simulações envolvendo decoerência ou operações com ruído.

A estrutura deste trabalho é como segue. No Capítulo 2 são apresentados os conceitos básicos que fundamentam a computação quântica, tais como conceitos de Mecânica Quântica, caminhadas quânticas e a notação utilizada. No Capítulo 3, cada passo do algoritmo é detalhadamente apresentado. No Capítulo 4 são mostrados aspectos técnicos do desenvolvimento e utilização do simulador, explicando como ele pode ser utilizado no estudo de cada passo do algoritmo de distinção de elementos. Também são apresentados, na Seção 4.2, os resultados numéricos obtidos a partir do uso do simulador, mostrando como o consumo de memória e tempo crescem a medida que a lista de entrada aumenta. Finalmente, no Capítulo 5, é apresentada a conclusão final bem como projetos futuros.

# 2 Noções Preliminares

Para a compreensão do algoritmo de Distinção de Elementos é necessário conhecer um pouco de álgebra linear, que faz parte dos fundamentos da Mecânica Quântica e o conceito de caminhada quântica.

A álgebra linear define os espaços e operações vetoriais, que são utilizados pela computação quântica. Em geral, tais operações ocorrem entre operadores unitários através de um produto entre vetores conhecido como produto tensorial, que será definido mais adiante.

Junto com a álgebra linear, os quatro postulados da Mecânica Quântica definem as leis físicas sobre as quais a computação quântica está baseada, desta forma, o não cumprimento de algum destes postulados descaracteriza o ambiente quântico. Além destes conceitos básicos para a computação quântica, o algoritmo de distinção de elementos faz uso de conceitos importantes como caminhadas quânticas e busca quântica em grafos.

Neste capítulo são apresentados os conceitos básicos do paradigma quântico, de forma breve e direta, bem como a notação utilizada, conhecida como a notação de Dirac.

## 2.1 A notação utilizada e o produto tensorial

A notação vetorial que é utilizada para descrever os estados quânticos na Mecânica Quântica, bem como para denotar vetores, é conhecida como Notação de Dirac. Nesta notação um vetor $\psi$ é representado por $|\psi\rangle$, também chamado de Ket. A representação de seu vetor dual $\psi^\dagger$ é dada por $|\psi\rangle^\dagger = \langle\psi|$, também chamado de Bra. Dessa forma, um vetor $\psi = \{v_1, ..., v_n\}$ e seu vetor dual tem as seguintes representações, respectivamente:

$$|\psi\rangle = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \text{ e } \langle\psi| = [{v_1}^*, \ldots, {v_n}^*]. \tag{2.1}$$

Quaisquer operações lineares podem ser escritos utilizando esta notação. Desse modo o produto interno, também conhecido como produto escalar, entre dois vetores é escrito como

$$\langle\phi|\psi\rangle, \tag{2.2}$$

enquanto o produto externo, ou produto vetorial, entre os mesmos dois vetores é escrito como

$$|\phi\rangle\langle\psi|. \tag{2.3}$$

Além do mais esta notação permite a simplificação da escrita de um produto tensorial entre dois ou mais vetores, como é visto mais adiante.

O produto tensorial, também conhecido na literatura como produto de Kronecker, entre dois vetores consiste na multiplicação de cada componente do primeiro vetor com cada um dos componentes do outro vetor. Este produto é uma forma de combinar dois espaços vetoriais distintos, bem como vetores e operadores de diferentes dimensões. Desta forma, sejam os espaços vetoriais $\mathcal{H}_1$ e $\mathcal{H}_2$ de dimensões $n$ e $p$, respectivamente. O produto tensorial entre estes dois espaços vetoriais é representado por $\mathcal{H}_1 \otimes \mathcal{H}_2$, gerando assim um novo espaço vetorial de dimensão $np$. Analogamente, o produto tensorial entre dois vetores

$$|V\rangle = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \text{ e } |W\rangle = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix},$$

onde $|V\rangle$ possui comprimento $n$ e $|W\rangle$ possui comprimento $p$, será um novo vetor denotado por $|V\rangle \otimes |W\rangle$ de comprimento $np$ que será escrito como

$$|V\rangle \otimes |W\rangle = \begin{bmatrix} v_1 w_1 \\ \vdots \\ v_1 w_p \\ \vdots \\ \vdots \\ v_n w_1 \\ \vdots \\ v_n w_p \end{bmatrix}.$$

O produto de Kronecker não é comutativo, e este possui três axiomas:

1. Para qualquer $c \in \mathbb{C}$, $|\psi_1\rangle \in \mathcal{H}_1$ e $|\psi_2\rangle \in \mathcal{H}_2$,

$$c(|\psi_1\rangle \otimes |\psi_2\rangle) = c|\psi_1\rangle \otimes |\psi_2\rangle = |\psi_1\rangle \otimes c|\psi_2\rangle.$$

2. Para qualquer $|\psi_1\rangle, |\theta_1\rangle \in \mathcal{H}_1$ e $|\psi_2\rangle \in \mathcal{H}_2$

$$(|\psi_1\rangle + |\theta_1\rangle) \otimes |\psi_2\rangle = (|\psi_1\rangle \otimes |\psi_2\rangle) + (|\theta_1\rangle \otimes |\psi_2\rangle).$$

3. Para qualquer $|\psi_1\rangle \in \mathcal{H}_1$ e $|\psi_2\rangle, |\theta_2\rangle \in \mathcal{H}_2$

$$|\psi_1\rangle \otimes (|\theta_2\rangle + |\psi_2\rangle) = (|\psi_1\rangle \otimes |\theta_2\rangle) + (|\psi_1\rangle \otimes |\psi_2\rangle).$$

Sejam agora $A$ e $B$ dois operadores lineares em $\mathcal{H}_1$ e $\mathcal{H}_2$, respectivamente. Logo, $A \otimes B$ é um operador linear no espaço $\mathcal{H}_1 \otimes \mathcal{H}_2$, definido por

$$(A \otimes B)(|\psi_1\rangle \otimes |\psi_2\rangle) \equiv A|\psi_1\rangle \otimes B|\psi_2\rangle.$$

O produto tensorial pode ser escrito omitindo-se o operador $\otimes$, de forma que $|\psi\rangle \otimes |\theta\rangle$ é frequentemente escrito como $|\psi\rangle|\theta\rangle$, ou também como $|\psi\theta\rangle$.

## 2.2 Mecânica Quântica

Os quatro postulados da Mecânica Quântica descrevem quais características uma máquina quântica deve possuir, definindo assim o espaço de estados, a evolução temporal do estado, a composição entre estados, e por fim a medição. Todos os quatro postulados devem ser respeitados de forma a manter as características quânticas.

O **primeiro postulado** descreve o espaço de estados do sistema. Todo sistema físico isolado tem associado a ele um espaço vetorial complexo, que é o espaço de estados do sistema. No paradigma quântico, o estado mais trivial é o próprio qubit, que é um vetor normal no espaço de Hilbert bidimensional. Este vetor pode ser escrito na base computacional $\{|0\rangle, |1\rangle\}$ de maneira genérica como sendo

$$\alpha|0\rangle + \beta|1\rangle,$$

onde $|0\rangle$ e $|1\rangle$ são os estados que representam os valores 0 e 1 respectivamente, e $\alpha, \beta \in \mathbb{C}$ representam as amplitudes de $|0\rangle$ e $|1\rangle$ respectivamente. Para que este estado seja normal, o produto interno de um vetor com ele mesmo deve ser igual a 1, dessa forma $|\alpha|^2+|\beta|^2 = 1$.

A evolução do estado em relação ao tempo é descrita no **segundo postulado**. Todo sistema quântico evolui de maneira unitária, ou seja, a evolução do estado do sistema tem associada a ela um operador unitário (operador cujo seu adjunto é igual ao seu conjugado, de modo que $UU^\dagger = U^\dagger U = I$). Devido a este fato, esta evolução é determinística e reversível, *i.e.*, a partir do estado final resultante é possível recuperar o estado inicial. Assim, seja $U$ um operador unitário qualquer e $|\psi_0\rangle$ o estado inicial do sistema. Ao aplicar o operador $U$ em $|\psi_0\rangle$ é obtido um novo estado $|\psi_1\rangle$, tal que

$$|\psi_1\rangle = U|\psi_0\rangle$$

e pelo fato de ser unitário tem-se

$$|\psi_0\rangle = U^\dagger|\psi_1\rangle.$$

Algoritmos quânticos são desenvolvidos utilizando prescrições de operadores uni-

tários aplicados a uma condição inicial. Os exemplos de operadores unitários mais famosos presentes na literatura são as matrizes de Pauli, e o operador de Hadamard, representado por

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

cuja ação é muito importante, pois possibilita que seja feita uma superposição de todos os estados possíveis de uma dada entrada.

No **terceiro postulado**, a composição de estados é descrita através do produto tensorial entre estes estados. Quando há um estado superposto que não é possível decompor através de um produto tensorial é dito que tal estado está emaranhado. Estar emaranhado significa que dois ou mais qubits não são independentes. Este é, por exemplo, o caso do estado composto

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}},$$

cuja fatoração em dois estados independentes não é possível. No entanto, o estado

$$|\psi\rangle = \frac{|00\rangle + |10\rangle - |01\rangle - |11\rangle}{2},$$

também representa um estado composto, porém este estado é formado pelo produto tensorial entre os estados

$$|\psi_1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

e

$$|\psi_2\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

O **quarto postulado** descreve o procedimento para se acessar uma informação de um ou mais qubits, a medição, que deve ser feita sobre uma base, que neste trabalho será sempre considerada como sendo a base computacional. Ao se realizar a medição, o estado irá deixar de estar isolado, desrespeitando as condições do primeiro postulado. Dessa forma, tal estado irá colapsar para algum outro estado, que será o estado obtido após a medição, resultando na perda de todas as demais informações. Em outras palavras, para um estado qualquer

$$|\psi\rangle = \sum_i \alpha_i |\phi_i\rangle, \tag{2.4}$$

ao ser realizada a medição, o estado colapsará para o estado $|\phi_i\rangle$, com probabilidade $|\alpha_i|^2$.

Supondo o estado

$$|\psi\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}},$$

que é gerado a partir da aplicação do operador de Hadamard no estado $|0\rangle$, tem-se que

tanto a amplitude de $|0\rangle$, $\alpha$, quanto a amplitude de $|1\rangle$, $\beta$, valem $1/\sqrt{2}$. Dessa forma a probabilidade de que ao se medir este estado, o sistema colapse ou para o estado $|0\rangle$ ou para o estado $|1\rangle$ é $|\alpha|^2$ e $|\beta|^2$, respectivamente, o que resulta em ambos os casos em uma probabilidade de 50%.

## 2.3 Caminhadas quânticas

Nesta seção são vistos os conceitos de caminhadas quânticas, que são utilizados pelo algoritmo de Distinção de Elementos. Na computação clássica existe o conceito de *random walk*, ou caminhada aleatória, que ocorre sobre um grafo de maneira não determinística onde cada passo da caminhada é definido aleatoriamente por uma moeda. Este conceito é utilizado na modelagem de diversos problemas importantes computacionalmente como, por exemplo, o problema de conectividade em grafos. Esta caminhada pode ser modelada em tempo contínuo ou em tempo discreto, sendo esta última modelagem utilizada nas seções subsequentes.

No paradigma quântico existe o conceito análogo conhecido como *quantum walk*, ou caminhada quântica, e assim como sua correspondente clássica, define o movimento de uma partícula em um grafo, onde a partícula, neste caso, é descrita por um vetor normalizado no espaço de Hilbert e seu movimento é dado por um operador unitário, mantendo assim as características quânticas.

Em um grafo pode haver ligações entre vértices que o caminhante não poderá percorrer, o que é conhecido com ligações interrompidas. Este tipo de modelagem é importante, porém está fora do escopo deste trabalho. Para mais informações sobre este conceito, recomenda-se ao leitor a referência [13].

Inicialmente é apresentado o conceito de caminhada quântica em malha unidimensional infinita, pois este é o modelo mais simples e é suficiente para a compreensão do funcionamento da caminhada quântica. Em seguida, é apresentado o conceito de caminhada quântica de Szegedy, que é uma caminhada descrita por operadores de reflexão em um grafo bipartido [18].

### 2.3.1 Caminhadas em malhas unidimensionais

Este modelo de caminhada está definido sobre uma reta, *i.e.*, uma malha unidimensional. Seja $\mathcal{H}_1$ o espaço de Hilbert que é gerado por todas as possíveis posições da partícula nesta malha. Suponha que este estado tenha dimensões infinitas, dessa forma, define-se a base canônica para o subespaço-posição como sendo $\mathcal{B}_p = \{|x\rangle : x \in \mathbb{Z}\}$. Para que este modelo possa ser definido em tempo discreto é necessário que se adicione mais um grau de liberdade, chamado de estado da moeda ou quirilidade, e que possui evolução determinística e unitária, devido ao segundo postulado da Mecânica Quântica, visto na

Seção 2.2.

Denote $\mathcal{H}_2$ como sendo o espaço gerado por todos os possíveis estados da moeda, que determinam qual o sentido do movimento do caminhante sobre a malha. Com isso a base canônica para o espaço da moeda é dado por $\mathcal{B}_c = \{|j\rangle : j \in \{0, 1\}\}$. Desta forma define-se o espaço de Hilbert como sendo $\mathcal{H}_2 \otimes \mathcal{H}_1$.

O estado genérico do caminhante num instante de tempo $t$ é definido como

$$|\psi(t)\rangle = \sum_{j=0}^{1} \sum_{x=-\infty}^{\infty} \psi_{j,x}(t)|j\rangle|x\rangle,$$

onde $\psi_{j,x} \in \mathbb{C}$ e $\sum_j \sum_x |\psi_{j,x}|^2 = 1$.

O primeiro passo do caminhante ocorre devido à uma operação unitária que se assemelha com o "lançamento da moeda"da caminhada aleatória, no entanto, vale a pena ressaltar que no caso da caminha quântica este "lançamento de moeda"ocorre de forma determinística, logo, é reversível, diferentemente do operador moeda da caminhada clássica. O operador moeda é dado por

$$\mathcal{M} = \sum_{j,k=0}^{1} \mathcal{M}_{j,k}|j\rangle\langle k|,$$

porém este operador pode ser definido de maneira livre desde que se mantenha unitário. No caso da caminhada na reta em geral é utilizado o operador de Hadamard, visto na Seção 2.2.

Em seguida, o resultado do passo do caminhante é condicionado ao resultado da moeda, que pode ser descrito, neste caso, como o operador unitário que define o operador de deslocamento

$$\mathcal{D} = \sum_{j=0}^{1} \sum_{x=-\infty}^{\infty} |j\rangle\langle j| \otimes |x + (-1)^j\rangle\langle x|.$$

Logo, o operador que define a evolução unitária para um passo de caminhada é equivalente a

$$\mathcal{U} = \mathcal{D} \circ (\mathcal{M} \otimes \mathcal{I}_p),$$

onde $\mathcal{I}_p$ é o operador identidade no subespaço-posição.

A probabilidade de que seja encontrado o caminhante na posição $x$ fazendo-se a medição é

$$\mathcal{P}(x, t) = \sum_{j=0}^{1} |\psi_{j,x}(t)|^2.$$

### 2.3.2 Caminhada quântica de Szegedy

Inspirado no algoritmo de Ambainis [4], este modelo de caminhada quântica é descrito por operadores de reflexão em um grafo bipartido, onde operador de reflexão é todo operador de $\mathbb{R}^2$ ou $\mathbb{R}^3$ que transforma cada vetor, em suas respectivas dimensões, em seu simétrico relativamente a uma reta ou um plano que contenha a origem. Denote $S$ e $T$ partições de conjuntos de vértices em um grafo bipartido onde $v_s$ e $v_t$ são vértices genéricos destes respectivos conjuntos. $P$ e $Q$ são matrizes estocásticas que descrevem as probabilidades de movimentos de $S$ para $T$ e de $T$ para $S$, respectivamente, com suas componentes

$$\sum_{v_t \in T} p_{st} = 1, \forall v_s \in S$$

e

$$\sum_{v_s \in S} q_{ts} = 1, \forall v_t \in T.$$

Ao grafo bipartido é associado um espaço de Hilbert $\mathcal{H}^{n_S n_T}$, em que $n_S = |S|$ e $n_T = |T|$, possuindo a base computacional $b = \{|v_s, v_t\rangle : v_s \in S, v_t \in T\}$. Definem-se os operadores $A : \mathcal{H}^{n_S} \to \mathcal{H}^{n_S n_T}$ como

$$A = \sum_{v_s \in S} |\phi_s\rangle\langle v_s|,$$

onde

$$|\phi_S\rangle = |v_s\rangle \otimes \left( \sum_{v_t \in T} \sqrt{p_{st}} |v_t\rangle \right)$$

e $B : \mathcal{H}^{n_T} \to \mathcal{H}^{n_S n_T}$ como

$$B = \sum_{v_t \in T} |\psi_t\rangle\langle v_t|,$$

onde

$$|\psi_T\rangle = \left( \sum_{v_s \in S} \sqrt{q_{ts}} |v_s\rangle \right) \otimes |v_t\rangle.$$

A interpretação que se tem é que $|\phi_s\rangle$ representa a superposição de todas as arestas que são originados em $v_s$. O mesmo vale para o caso de $|\psi_t\rangle$.

Pode-se então definir os operadores de reflexão $R_A$ e $R_B$, onde $R_A$ reflete um vetor genérico de $\mathcal{H}^{n_S n_T}$ em torno do subespaço $\mathcal{H}_A$, que é gerado pelos vetores $|\phi_s\rangle$. O mesmo vale para $R_B$. Estes operadores são definidos como

$$R_A = 2 \sum_{v_s \in S} |\phi_s\rangle\langle\phi_s| - I_{n_S n_T}$$

e

$$R_B = 2\sum_{v_t \in T} |\psi_t\rangle\langle\psi_t| - I_{n_S n_T}$$

e juntos definem o operador de reflexão do sistema, que é dado por

$$U_{P,Q} := R_B R_A.$$

Tomando como exemplo o grafo da Figura 2.1, tem-se as matrizes de

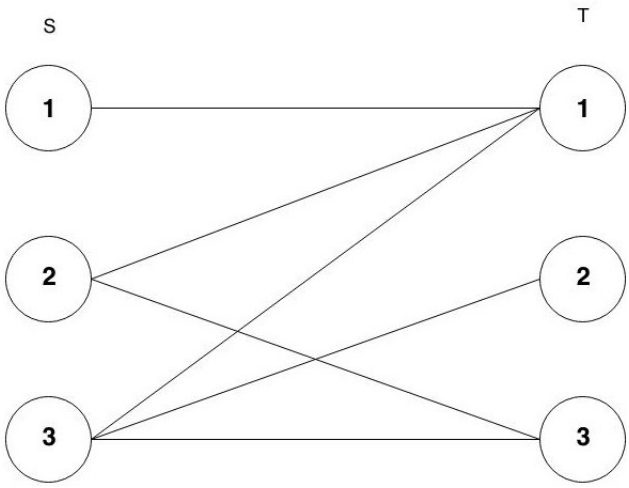


Figura 2.1: Grafo bipartido onde $S = \{1, 2, 3\}$ e $T = \{1, 2, 3\}$.

probabilidade

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$

e

$$Q = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1 \\ 0 & 1/2 & 1/2 \end{pmatrix}.$$

No caso apresentado, tem-se que a base computacional no espaço de Hilbert é dada por

$$\{|1,1\rangle, |1,2\rangle, |1,3\rangle, |2,1\rangle, |2,2\rangle, |2,3\rangle, |3,1\rangle, |3,2\rangle, |3,3\rangle\}.$$

Como $|\phi_s\rangle$ é a superposição de todas arestas que se originam a partir de $v_s$, tem-se a descrição dos estados como

$$|\phi_1\rangle = |1,1\rangle,$$

$$|\phi_2\rangle = \sqrt{\frac{1}{2}}|2,1\rangle + \sqrt{\frac{1}{2}}|2,3\rangle,$$

$$|\phi_3\rangle = \sqrt{\frac{1}{3}}|3,1\rangle + \sqrt{\frac{1}{3}}|3,2\rangle + \sqrt{\frac{1}{3}}|3,3\rangle.$$

O mesmo vale para o caso de $|\psi_T\rangle$, que é a superposição dos estados

$$|\psi_1\rangle = \sqrt{\frac{1}{3}}|1,1\rangle + \sqrt{\frac{1}{3}}|2,1\rangle + \sqrt{\frac{1}{3}}|3,1\rangle,$$

$$|\psi_2\rangle = |3,2\rangle,$$

$$|\psi_3\rangle = \sqrt{\frac{1}{2}}|2,3\rangle + \sqrt{\frac{1}{2}}|3,3\rangle.$$

A partir daí já se torna possível obter as matrizes $A$ e $B$, e por fim, o operador de evolução $U_{P,Q}$.

A diferença entre a caminhada de Szegedy [19] e a caminhada quântica, introduzida por Aharonov et al. [3], está na estrutura de como a caminhada ocorre. Enquanto a segunda utiliza uma moeda, como visto na seção anterior, a primeira é vista como uma caminhada nas arestas da malha. Desta forma, define-se um estado $|a\rangle|b\rangle$ como sendo uma aresta $(a, b)$, que representa que o caminhante saiu de $b$ e chegou em $a$.

Todo grafo pode ser convertido para um grafo bipartido por meio de um simples processo de duplicação, como mostrado na Figura 2.2, de modo que cada aresta $(a_i, a_j)$ do grafo original é convertida em duas arestas $(a_i, b_j)$ e $(b_i, a_j)$ no grafo bipartido. Assim, pode-se utilizar o passeio quântico de Szegedy para problemas que envolvam cadeia de Markov.

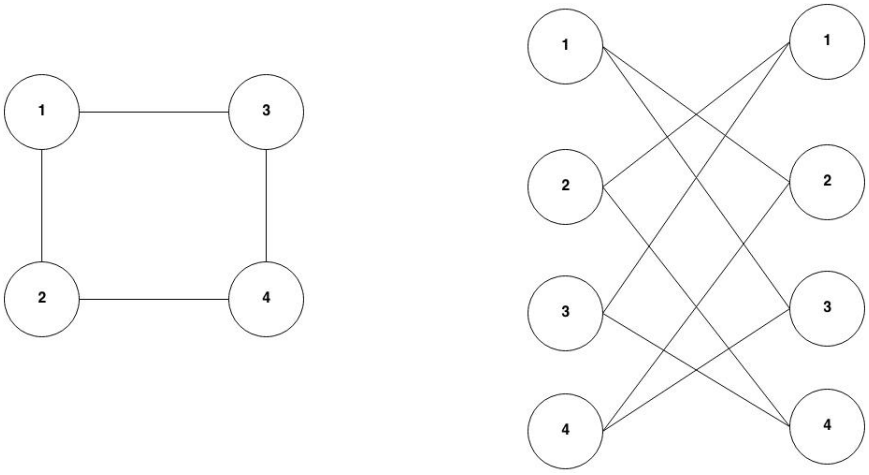


Figura 2.2: Exemplo de um grafo de 4 vértices e seu grafo bipartido gerado a partir do processo de duplicação.

# 3 Algoritmo Quântico de Distinção de Elementos

O algoritmo de distinção de elementos, proposto por Ambainis [4], soluciona o problema de descobrir se todos os elementos de uma lista são distintos em $O(N^{k/k+1})$ passos, onde $N$ é o número de elementos na lista e $k$ representa o número de elementos não-distintos, *i.e.*, representa a quantidade de elementos na lista que possuem os mesmos valores. O caso abordado pelo simulador apresentado neste trabalho é conhecido como 2-colisão, *i.e.*, $k = 2$, indicando que dois elementos da lista de entrada são iguais. Este algoritmo ainda pode ser estendido para o caso de múltiplas $k$-colisões, neste caso, o algoritmo consegue identificar a existência de $k$ valores onde cada um destes valores possui repetição na lista. Por estar fora do escopo deste trabaho, para um aprofundamento maior sobre este outro caso, além do caso onde $k$ possui valores arbitrários, recomenda-se a leitura da referência [4].

Neste capítulo define-se o problema solucionado pelo algoritmo de Ambainis. Também apresentam-se as técnicas clássicas, e posteriormente, as técnicas quânticas para solucionar este problema. Finalmente apresenta-se o algoritmo de distinção de elementos, explicando-se cada passo do algoritmo de maneira sucinta.

## 3.1 O problema abordado e as ideias principais

Suponha uma lista de $N$ elementos $x_1, ..., x_N \in [M]$, onde $M$ representa o valor máximo que um elemento pode ter na lista. Deseja-se descobrir se nesta lista todos os elementos são distintos. No paradigma clássico pode-se simplesmente comparar cada número com os demais, o que gera um procedimento que requer $O(N^2)$ passos. Uma outra abordagem mais interessante e que requer $O(N \log N)$ passos é primeiramente ordenar a lista, que pode ser feito em $O(N \log N)$ passos utilizando o algoritmo mais eficiente conhecido, e posteriormente comparar cada número com o próximo vizinho, o que requer $O(N)$ comparações. No entanto, o limite inferior para a este problema, no modelo clássico, é $\Omega(N)$ já que todo algoritmo necessita ler cada um dos $N$ elementos da lista pelo menos uma vez. De fato, assumindo que todo elemento $x$ na lista está contido em $x_{MIN} \leq x \leq x_{MAX}$, pode-se buscar algum elemento repetido em tempo linear, por exemplo, utilizando o Bucket sort, a fim de resolver o problema de distinção de elementos em $O(N)$ passos.

No modelo quântico, este problema pode ser solucionado de forma mais eficiente que no modelo clássico. A metodologia adotada é a criação de um grafo bipartido, co-

nhecido como grafo de Johnson. Neste tipo de grafo tem-se duas partições, $S$ e $T$, em que cada vértice destas partições corresponde a um subconjunto de $[N]$. Vértices $v_S$ da partição $S$ representam conjuntos $S \subseteq [N]$ de tamanho $r$, onde $r = \lfloor N^{2/3} \rfloor$, enquanto vertices $v_T$ da partição $T$ representam conjuntos $T \subseteq [N]$ de tamanho $r+1$. Por ser bipartido, apenas ocorrerão conexões entre vértices $v_S$ e $v_T$, se e somente se $T = S \cup \{i\}$ para algum $i \in [N]$. Um vértice $v_S$ é dito marcado se ele possuir $i, j$ onde $i \neq j$ e $x_i = x_j$.

Uma abordagem que pode ser feita é utilizar o algoritmo quântico de busca de Grover [10]. Seja $\varepsilon$ uma fração de vértices marcados então a busca de Grover encontrará algum vértice marcado depois de $O(1/\sqrt{\varepsilon})$ vértices. Nessa abordagem a probabilidade de que haja um vértice marcado para um $S$ qualquer é

$$Pr[i \in S; j \in S] = Pr[i \in S] Pr[j \in S | i \in S] = \frac{N^{2/3}}{N} \frac{N^{2/3} - 1}{N - 1} = (1 - o(1)) \frac{1}{N^{2/3}}. \quad (3.1)$$

Dessa forma, este algoritmo irá encontrar o vértice marcado em $O(N^{1/3})$ vértices. No entanto, ao encontrar um vértice marcado, o algoritmo verifica se há dois elementos iguais em $O(N^{2/3})$, o que resulta em uma complexidade final de $O(N)$, representando nenhum ganho em relação ao método clássico descrito anteriormente.

O algoritmo de Distinção de Elementos faz uso da busca em grafos junto aos conceitos de caminhadas quânticas. Cada partição do grafo encontra-se em um espaço vetorial de Hilbert diferente. Dessa forma a partição $S$ encontra-se no espaço $\mathcal{H}$, cuja dimensão é $\binom{N}{r} M^r (N - r)$ e as bases de seus estados são $|S, x, y\rangle$, onde $y \in [N] \backslash S$. A partição $T$ encontra-se no espaço de Hilbert $\mathcal{H}'$, cuja dimensão é $\binom{N}{r+1} M^{r+1}(r+1)$ e também possui bases $|S, x, y\rangle$, no entanto, neste caso, $y \in S$.

É assumido que já foi verificado se um vértice é marcado ou não apenas consultando todos os $x_i$, onde $i \in [S]$. Como o próximo vértice $v_T$ é tal que existe apenas um elemento de índice $i$, onde $i \notin S$, é preciso apenas que seja verificado esse único elemento $x_i$, $i \in T$, reduzindo assim o espaço de busca e diminuindo a complexidade do procedimento para $O(N^{2/3})$. Todo este procedimento utiliza uma quantidade de qubits de memória, em um computador quântico, da ordem de

$$O\left( \binom{N}{r} M^r (N - r) + \binom{N}{r+1} M^{r+1}(r+1) \right) = O(r(\log N + \log M)).$$

## 3.2 Descrição do algoritmo

Nesta seção é descrito o algoritmo e cada um de seus passos até a medição para a obtenção da resposta final. Basicamente, o algoritmo verifica se o vértice $v_s$, no qual o caminhante se encontra, está marcado e posteriormente o move para o vértice adjacente $v_t$. De maneira informal, pode-se assumir que existe um algoritmo $A$ que encontra tal

vértice marcado em $M$ movimentos através dos vértices, desse modo o algoritmo soluciona o problema em $M + r$ passos, já que cada vértice possui tamanho $r$. Tem-se então a seguinte ideia:

1. Use $r$ consultas para verificar todos os $x_i, i \in S$ para o vértice inicial $v_s$.
2. Repita as próximas operações $M$ vezes:
   (i) Verifique se o vértice corrente está marcado. (Isto é feito sem nenhuma consulta, pois isto já foi previamente verificado).
   (ii) Simule o algoritmo $A$ até o próximo movimento, encontre o vértice $v_t$ para qual o caminhante se move a partir de $v_s$. Então o caminhante é movido para $v_t$ consultando $x_i, i \in T \backslash S$. Após isso, todos os $x_i, i \in T$ já serão conhecidos, e então torne $S = T$.

Inicialmente todos os estados se tornam um estado superposto de tal modo que possibilite o paralelismo quântico. O primeiro passo do algoritmo consiste em criar justamente esta superposição uniforme, gerando o estado

$$\frac{1}{\sqrt{\binom{N}{r}(N-r)}} \sum_{|S|=r, y \notin S} |S\rangle|y\rangle. \tag{3.2}$$

Após a geração desta superposição, é feita a consulta de todos os $x_i$ para todo $i \in S$, desta forma o registrador $|x\rangle$ passa a fazer parte da superposição, resultando no estado

$$\frac{1}{\sqrt{\binom{N}{r}(N-r)}} \sum_{|S|=r, y \notin S} |S\rangle|y\rangle \bigotimes_{i \in S} |x_i\rangle. \tag{3.3}$$

Estes dois passos iniciais de certa forma preparam o estado inicial para o algoritmo propriamente dito, sendo no passo seguinte a ocorrência de fato da ideia explicitada anteriormente. Neste passo existem dois laços onde ocorrerá a caminhada quântica que fará com que a amplitude do vértice marcado cresça em detrimento das demais amplitudes de vértices não marcados. De forma genérica define-se o laço externo $t_1 = O((\frac{N}{r})^{k/2})$, onde $k = 2$ no caso abordado. Sendo assim, dentro deste laço é aplicado um operador condicional que inverterá a fase da amplitude do estado que possuir algum $x_i = x_j$, onde $i \neq j$ e $i, j \in S$. Matematicamente o que ocorre é a transformação $|S\rangle|y\rangle|x\rangle \rightarrow -|S\rangle|y\rangle|x\rangle$. Em seguida tem-se o segundo laço que, como demonstrado por Ambainis, irá executar $t_2 = \lceil \frac{\pi}{3\sqrt{k}} \sqrt{r} \rceil$, onde de fato ocorre a caminhada quântica que moverá o caminhante através do grafo, *i.e.*, entre os espaços de $\mathcal{H}$ e $\mathcal{H}'$. A cada repetição deste laço, o caminhante irá se mover do espaço $\mathcal{H}$ para o espaço $\mathcal{H}'$, retornando posteriormente para o espaço de origem.

Pode-se definir um passo desta caminhada em seis partes, onde as três primeiras descrevem a passagem do caminhante do espaço $\mathcal{H}$ para o espaço $\mathcal{H}'$, enquanto as três últimas descrevem o retorno do mesmo para o espaço inicial. A primeira parte consiste

em aplicar a transformação que mapeia $|S\rangle|y\rangle$ para

$$|S\rangle\left(\left(-1+\frac{2}{N-r}\right)|y\rangle+\frac{2}{N-r}\sum_{y'\notin S,y'\neq y}|y'\rangle\right). \tag{3.4}$$

Omite-se o $|x\rangle$ neste caso apenas para facilitar a leitura, já que este não sofrerá transformação neste momento. Note que esta ação faz com que o caminhante comece a se deslocar para todos os possíveis vértices adjacentes ao vértice de partida.

Em seguida inicia-se o mapeamento do estado de $\mathcal{H}$ para $\mathcal{H}'$, *i.e.*, ocorre a expansão do estado inicial para que este passe a comportar mais um elemento em seus subconjuntos, representados pelos respectivos vértices, pois como já descrito, o espaço $\mathcal{H}'$ possui vértices que representam subconjuntos de tamanho $r+1$, enquanto no espaço inicial, o tamanho dos subconjuntos é $r$. Esta expansão é feita adicionando $y$ ao registrador $|S\rangle$ e aumentando a capacidade do registrador $|x\rangle$ de $k$ para $k+1$ introduzindo 0 no local que corresponde a $y$.

Agora, procura-se por $x_y$, onde o índice $y$ é o valor do $y$ que foi adicionado anteriormente ao registrador $|S\rangle$. Ao obter o valor de $x_y$, adiciona-se tal valor em $x$, no local correspondente ao $y$, *i.e.*, na posição de $x$ que foi acrescentado o valor 0.

Novamente aplica-se uma transformação em $|S\rangle|y\rangle$ transformando-o no estado

$$|S\rangle\left(\left(-1+\frac{2}{r+1}\right)|y\rangle+\frac{2}{r+1}\sum_{y'\in S,y'\neq y}|y'\rangle\right), \tag{3.5}$$

que faz com que o caminhante comece a retornar para o espaço de Hilbert inicial.

A próxima parte ajusta as dimensões do estado para as dimensões do espaço $\mathcal{H}$. Isto é feito apagando o elemento do registrador $|x\rangle$ que corresponde ao novo $y$ usando-o como entrada para procurar por $x_y$. Para finalizar o passo da caminhada, remove-se o componente 0 que corresponde a $y$ no registrador $|x\rangle$, e remove-se $y$ do registrador $|S\rangle$. Desse modo, o caminhante retornou para o espaço de Hilbert inicial e as amplitudes dos vértices foram modificadas, de forma que agora a amplitude do vértice marcado começa a ficar maior que as demais amplitudes, destacando o vértice marcado entre os demais.

Ao final dos laços, ocorre a medição. Esta ação, como visto na seção 2.2, fará com que o estado superposto colapse para um único estado, que será a resposta final do algoritmo, onde a probabilidade de colapsar para cada estado é definida por suas respectivas amplitudes. Para consultar o algoritmo originalmente escrito por Ambainis [4], consulte o apêndice A.

# 4 Simulador e Resultados Numéricos das Simulações

Neste capítulo é apresentado um simulador numérico desenvolvido para abordar o algoritmo de distinção de elementos [2], provendo uma forma fácil de analisar cada passo deste algoritmo. São descritos detalhes técnicos além da descrição da implementação deste simulador, sua instalação e utilização. Também são apresentados os resultados numéricos obtidos a partir das simulações, analisando-se o crescimento exponencial de tempo de execução e memória consumida. Finalmente, é apresentado um breve exemplo de execução para uma lista com quatro elementos de entrada.

## 4.1 Simulador do algoritmo quântico de distinção de elementos

Abordagens analíticas para o estudo do comportamento de computadores e algoritmos quânticos de maneira geral se tornam impraticáveis devido a existência de cálculos e superposição de estados que não são triviais, e estes crescem de forma exponencial. Ainda que sejam utilizadas ferramentas de simulação em máquinas clássicas, encontra-se problema semelhante visto que tanto o consumo de memória quanto o tempo necessário para uma simulação possuem também crescimento exponencial. Apesar disso, diversos simuladores foram desenvolvidos com a finalidade de auxiliar nos estudos a respeito da computação quântica, como circuitos quânticos [5, 20, 21] e caminhadas quânticas em grafos [7, 13, 14].

O software foi implementado utilizando a linguagem $C++$ visando obter melhor eficiência e menor consumo de memória. O código fonte está disponível sob a licença GNU GPL, desta forma, qualquer usuário que possua conhecimentos acerca da linguagem de programação utilizada poderá modificar o simulador para alguma aplicação específica. Qualquer computador que possuir um compilador de $C++$ poderá executar o software sem problemas. Foram realizados experimentos em sistemas Linux e Windows obtendo resultados satisfatórios em ambos os casos sem que houvesse grande diferença de desempenho. Mais detalhes sobre o desempenho obtido pode ser visto na Seção 4.2.

Antes do início da implementação propriamente dita, foi feita uma abordagem analítica para o caso onde na lista de entrada possuía apenas quatro elementos. Este tamanho de lista é o menor caso não trivial possível de ser escolhido. Por simplicidade,

para reduzir a quantidade de cálculos executados, bem como o estado corrente, foram escolhidos poucos vértices do grafo gerado. Dessa forma, o processo de execução do algoritmo se deu de maneira mais rápida sem prejuízo na obtenção de conhecimento.

Após a compreensão do funcionamento do algoritmo, cada passo foi implementado e os estados foram implementados de forma a ocuparem o menor espaço de memória possível. Isto foi feito utilizando uma estrutura que armazenava para um único registrador $|S\rangle$ todos os valores possíveis para o registrador $|y\rangle$, mapeados para as suas respectivas amplitudes, evitando assim repetições de valores de $|S\rangle$ desnecessariamente. Pelo fato do registrador $|S\rangle$ armazenar os índices que representam os elementos na lista de entrada, o registrador $|x\rangle$ que contém os elementos não foi armazenado visto que para obter os elementos correntes de $|x\rangle$ basta utilizar os índices já armazenados no registrador $|S\rangle$.

A medida que os passos do algoritmo vão sendo executados as amplitudes dos estados vão sendo alteradas. Ao final do processo cada estado da superposição possuirá armazenado em sua estrutura de dados o valor de sua amplitude, com esse valor é calculada a probabilidade de cada estado ser medido. Tal medição é feita gerando-se um número aleatório entre zero e cem. Após isso, a probabilidade de cada estado é somada até que seja obtido um valor igual ou superior ao valor gerado aleatoriamente, desta forma o estado que fornecer a última probabilidade somada será o estado medido. Desse modo, o estado que representar o vértice marcado possuirá uma probabilidade muito maior do que os demais estados, logo terá uma chance maior de que um número gerado aleatoriamente esteja contido dentro de seu intervalo.

Como será visto mais adiante na Seção 4.2, estima-se que menos de 10 MB de memória seja necessário para a execução do simulador para o caso de uma lista pequena, com até 13 elementos, no entanto, simulações mais interessantes, *i.e.*, cuja lista de entrada possui uma quantidade de elementos maiores, serão limitadas pela quantidade de memória RAM disponível, devido ao crescimento exponencial do consumo da mesma. A princípio o software não utiliza nenhum subprograma externo, trabalhando de forma independente.

O simulador pode trabalhar tanto em ambientes Linux, quanto em ambientes Windows, todavia sua instalação nestes ambientes se dá de maneira distinta. Em um sistema Linux, deve-se primeiramente efetuar o download do código fonte[1] e descompactá-lo. Posteriormente é necessário compilar o código fonte utilizando um compilador de $C++$. Isto pode ser feito utilizando o comando *make* ou utilizando alguma *IDE*, obtendo ao final do processo o programa executável, que pode ser executado através do Linux Shell.

O processo de instalação em um sistema Windows também é bastante simples, basta fazer o download do instalador e seguir os passos indicados até que seja concluída a instalação. Para executar o programa basta que seja dado dois cliques com o mouse sobre o ícone do simulador.

Em ambos os casos ao executar o simulador, serão fornecidas duas opções de

[1] https://github.com/matheusmanzoli/QuantumDistinctness

preenchimento da lista de entrada: (i) gerar uma lista aleatoriamente, e (ii) definir cada elemento manualmente. Independentemente da escolha, o passo seguinte é informar o tamanho da lista de entrada. Após isso, caso tenha sido escolhida a opção (i), o simulador irá gerar aleatoriamente o valor de cada elemento e em seguida a simulação começará. Caso a opção (ii) tenha sido escolhida, o usuário deverá informar o valor de cada elemento. Após inserir o valor do último elemento, a simulação será iniciada. Ao final da simulação o simulador irá responder a pergunta *"Existe uma 2-colisão?"*, assim como o algoritmo.

Durante a execução um relatório é escrito descrevendo o estado superposto de cada passo do algoritmo, onde são mostradas as amplitudes de cada estado que compõe a superposição de forma que fique o crescimento da amplitude de um vértice marcado e o decaimento dos demais vértices.

## 4.2 Resultados numéricos

Nesta seção são apresentados os resultados obtidos nas simulações feitas usando o simulador para o algoritmo quântico de distinção de elementos, tais como o crescimento de memória consumida e o tempo necessário para a conclusão das simulações a medida que a lista de entrada cresce. Também é apresentado um exemplo para o caso de uma lista que possua quatro elementos.

### 4.2.1 Resultados numéricos obtidos

A máquina na qual foram executadas as simulações possui a seguinte configuração: 2.84 GB de memória RAM disponível, processador Intel Core i5 CPU 650, 3.20 GHz x 4 e 320 GB de memória secundária. Como pode-se perceber, trata-se de uma máquina comum, sendo assim, qualquer usuário poderá executar o simulador para o algoritmo de Ambainis sem que seja necessário um computador muito superior.

Como já era esperado, o tempo gasto para as simulações e a memória consumida nas mesmas, possuem crescimento exponencial. Isto ocorre devido ao fato de simulações de algoritmos quânticos em computadores clássicos, de forma geral, possuírem complexidade exponencial, tanto em tempo quanto em memória. Além disso, os espaços vetoriais utilizados por esses algoritmos também possuem crescimento exponencial, explicando o crescimento da memória consumida.

A complexidade assintótica deste software é $O(2^N)$ devido ao fato de que o algoritmo usado para gerar os subconjuntos que representam os vértices do grafo ser exponencial. Como pode ser visto na Figura 4.1, para listas de entrada que possuem poucos elementos, a simulação é finalizada em pouco tempo, no entanto, ao se acrescentar alguns poucos elementos à lista, o tempo necessário para a simulação cresce de forma exponencial. O gráfico da Figura 4.2, que representa o consumo de memória a medida que a lista

cresce, possui aparência bastante semelhante ao gráfico da Figura 4.1.

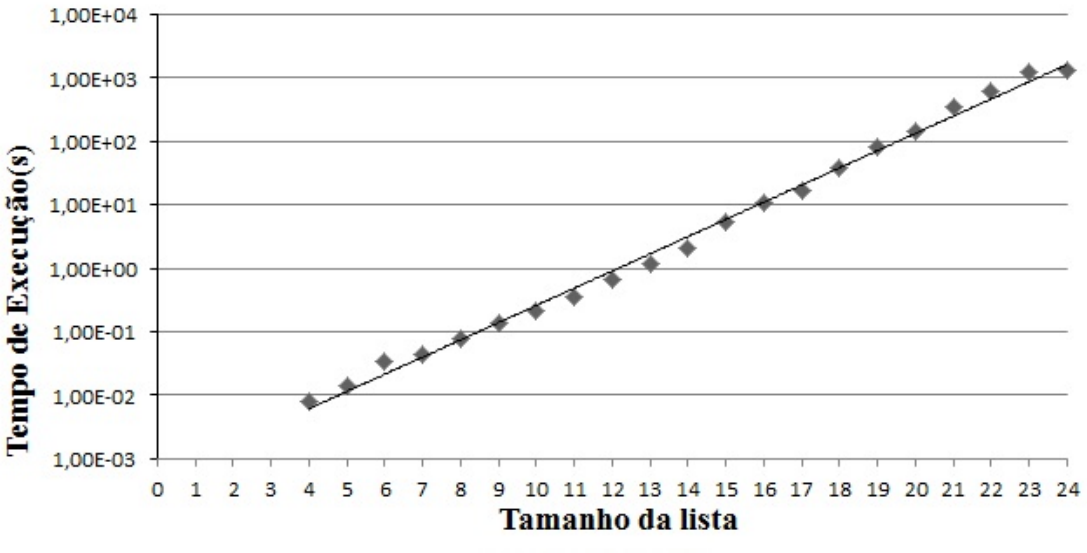


Figura 4.1: Gráfico (escala logarítmica) de tendência do tempo de simulação, em segundos, em função da quantidade de elementos na lista de entrada.

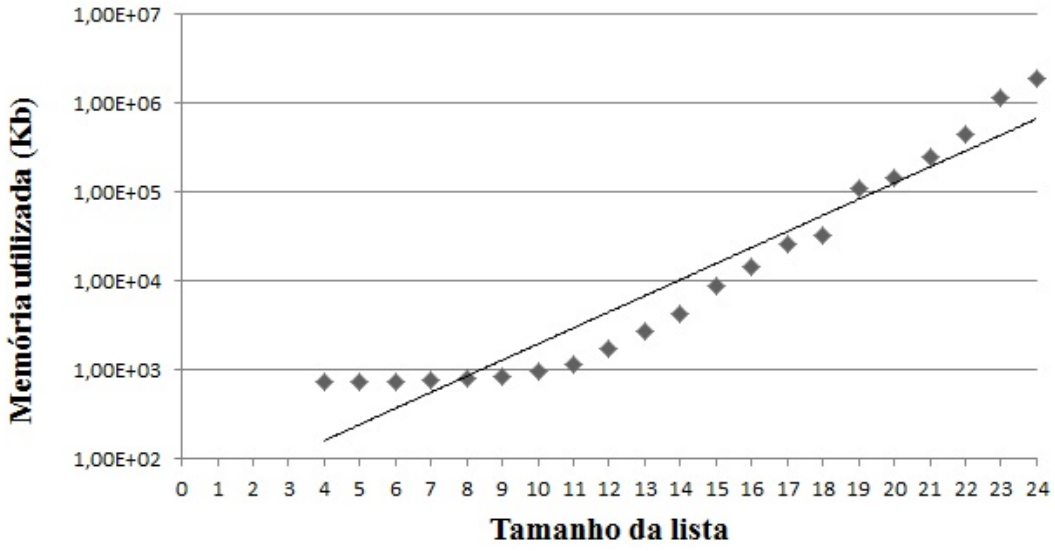


Figura 4.2: Gráfico (escala logarítmica) de tendência do consumo de memória, em kilobytes, em função da quantidade de elementos na lista de entrada.

Os testes foram iniciados com o menor valor não trivial para o tamanho da lista de entrada, neste caso, foram adicionados à lista quatro elementos. A cada novo teste variou-se o posicionamento dos elementos repetidos e, como já era esperado, tal alteração não influenciou nos resultados de desempenho obtidos. O tamanho da lista foi aumentado até vinte e quatro elementos. Para um valor maior do que este a memória da máquina de teste se tornou insuficiente, não possibilitando a execução da simulação.

Pode-se estimar a quantidade de tempo necessário, em segundos, e a memória consumida, em kilobytes, em uma simulação a partir da análise dos gráficos da Figura 4.1, que representa a tendência do tempo necessário para uma simulação (em escala logarítmica) em função da quantidade de elementos na lista de entrada para a execução de uma simulação, e da Figura 4.2, que representa o consumo de memória (em escala logarítmica) em função da quantidade de elementos na lista de entrada para a execução de uma simulação. Isto é feito através do cálculo do coeficiente angular da reta que representa a tendência de crescimento. Analisando primeiro o gráfico da Figura 4.1, tem-se o coeficiente angular $m_1 \approx 0,25$. Fazendo a mesma análise para o gráfico da Figura 4.2, estima-se o coeficiente angular $m_2 \approx 0,20$. De posse destes coeficientes angulares e dos gráficos, pode-se obter as equações reduzidas das retas

$$y \approx 10^{0,25x-3} \tag{4.1}$$

para estimar o tempo de simulação, em segundos, e

$$y \approx 10^{0,20x+1,4} \tag{4.2}$$

para estimar o consumo de memória, em kilobytes, na simulação, onde $x$ representa o tamanho da lista de entrada, enquanto $y$ representa o tempo necessário para ser executada a simulação, no caso da Equação 4.1, e o consumo de memória necessário para a simulação, no caso da Equação 4.2.

Com isso, utilizando estas equações, pode-se observar que para simulações bem pequenas cuja lista de entrada possua 13 elementos, são necessários aproximadamente 10 MB de memória e menos do que 2 segundos para que seja concluída a simulação. Se o tamanho da lista dobrar, estima-se que a simulação necessitará de cerca de 4 GB de memória RAM disponível e 1 hora de execução, aproximadamente.

### 4.2.2 Resultado numérico para simulação com lista de quatro elementos

Nesta seção é apresentado o resultado de uma simulação para uma lista com quatro elementos. Tal lista é representada por $L = \{1, 2, 1, 3\}$. O resultado do algoritmo de distinção de elementos gerou o grafo representado pela Figura 4.3, onde cada vértice é um subconjunto das posições.

O vértice mais escuro representa o vértice marcado, que terá maior probabilidade de ser medido ao final da simulação. Desse modo, a medida que a simulação ocorre a amplitude deste vértice irá aumentar, enquanto as demais amplitudes irão diminuir. A Tabela 4.1 demonstra esta variação de amplitude a medida que a simulação ocorre. Cada coluna da tabela representa um passo do algoritmo, *i.e.*, um passo da simulação,

enquanto cada linha representa um estado formado por $|S\rangle|y\rangle$. Para simplificar a tabela, são exibidos apenas os passos do algoritmo principal (algoritmo 1) a partir do Passo 2. A probabilidade de cada vértice é dada pela soma das probabilidades dos estados $|S_1\rangle|y_1\rangle$ e $|S_1\rangle|y_2\rangle$, onde $S_1$ é um subconjunto de índices de $N$ e $y_1, y_2 \notin S_1$.

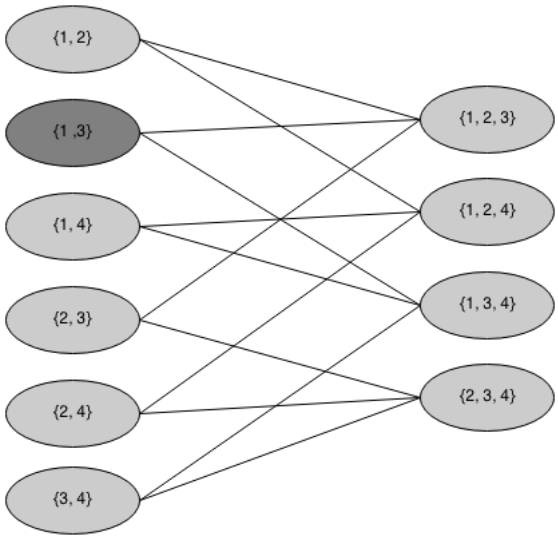


Figura 4.3: Grafo de Johnson no qual ocorre a execução do algoritmo de distinção de elementos. Este grafo é gerado a partir da lista $L = \{1, 2, 1, 3\}$.

| $\|S\rangle\|y\rangle$ | Passo 2 | Passo 3a ($1^o$ passagem) | Passo 3a ($2^o$ passagem) | Passo 4 | Pr(.) |
|---|---|---|---|---|---|
| $\|1,2\rangle\|3\rangle$ | 0,288 | 0,288 | -0.095 | -0.225 | 5.04 % |
| $\|1,2\rangle\|4\rangle$ | 0,288 | 0,288 | 0,288 | 0.161 | 2.57 % |
| $\|1,3\rangle\|2\rangle$ | 0,288 | -0,288 | -0.482 | 0.545 | 29.74 % |
| $\|1,3\rangle\|4\rangle$ | 0,288 | -0,288 | -0.482 | 0.545 | 29.74 % |
| $\|1,4\rangle\|2\rangle$ | 0,288 | 0,288 | 0,288 | 0.161 | 2.57 % |
| $\|1,4\rangle\|3\rangle$ | 0,288 | 0,288 | -0.095 | -0.225 | 5.04 % |
| $\|2,3\rangle\|1\rangle$ | 0,288 | 0,288 | -0.095 | -0.225 | 5.04 % |
| $\|2,3\rangle\|4\rangle$ | 0,288 | 0,288 | 0,288 | 0.161 | 2.57 % |
| $\|2,4\rangle\|1\rangle$ | 0,288 | 0,288 | 0,288 | -0.225 | 5.04 % |
| $\|2,4\rangle\|3\rangle$ | 0,288 | 0,288 | 0,288 | -0.225 | 5.04 % |
| $\|3,4\rangle\|1\rangle$ | 0,288 | 0,288 | -0.095 | -0.225 | 5.04 % |
| $\|3,4\rangle\|2\rangle$ | 0,288 | 0,288 | 0,288 | 0.161 | 2.57 % |

Tabela 4.1: Valores das Amplitudes de cada Vértice ao longo da Execução para $L = \{1, 2, 1, 3\}$ e a suas probabilidades de medição.

No início, durante o Passo 2 (Figura 4.4), onde a preparação do estado superposto inicial é concluída, todos possuem a mesma probabilidade. Devido ao fato, para este exemplo, do algoritmo efetuar duas vezes a caminhada quântica (algoritmo 2) a tabela somente apresenta as amplitudes antes desta caminhada ser iniciada, sendo representadas pelas colunas Passo 3 ($1^o$ passagem), também representado pela Figura 4.4, e Passo 3 ($2^o$ passagem), representado pela Figura 4.5. O Passo 4 (Figura 4.6) representa o fim

da execução do simulador, apresentando assim as amplitudes finais dos estados. Como pode-se observar, a amplitude do estado $|1,3\rangle$, onde de fato encontram-se os elementos não-distintos, é a maior de todas.

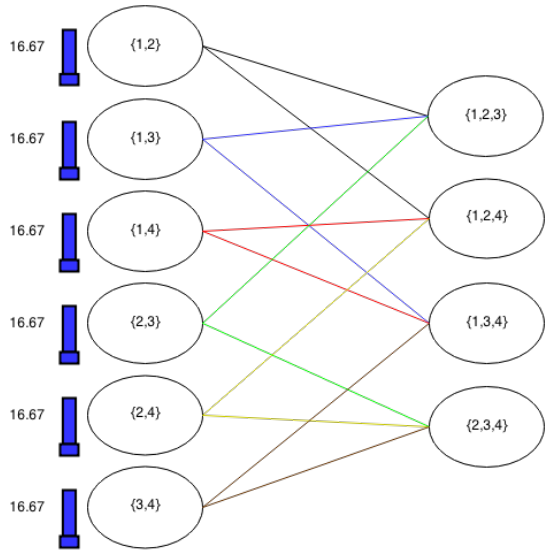


Figura 4.4: Grafo de Johnson com a probabilidade de cada vértice, referente ao Passo 2 e Passo 3 ($1^o$ passagem), do algoritmo principal. Este grafo é gerado a partir da lista $L = \{1, 2, 1, 3\}$.

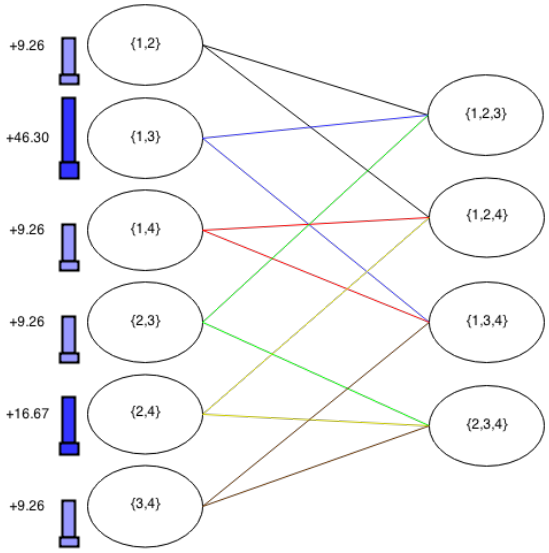


Figura 4.5: Grafo de Johnson com a probabilidade de cada vértice, referente ao Passo 3 ($2^o$ passagem), do algoritmo principal. Este grafo é gerado a partir da lista $L = \{1, 2, 1, 3\}$.

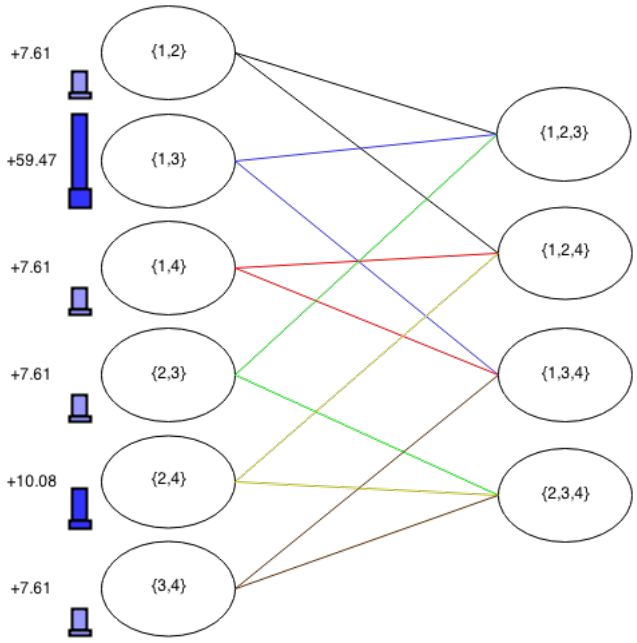


Figura 4.6: Grafo de Johnson com a probabilidade de cada vértice, referente ao Passo 4, do algoritmo principal. Este grafo é gerado a partir da lista $L = \{1, 2, 1, 3\}$.

Como visto na Tabela 4.2, para cada valor de $N$ existe uma probabilidade de medir um vértice marcado, e tal probabilidade é constante desde que o tamanho da lista se mantenha inalterado. Isto indica que a posição dos elementos na lista de entrada não interfere na probabilidade do algoritmo medir corretamente o estado que representa o vértice marcado.

| $N$ | Pr(.) |
|---|---|
| 4 | 59.46 % |
| 5 | 61.58 % |
| 6 | 55.56 % |
| 7 | 57.14 % |
| 8 | 53.22 % |
| 9 | 79.53 % |
| 10 | 90.63 % |
| 11 | 92.74 % |
| 12 | 78.66 % |
| 13 | 82.32 % |
| 14 | 82.41 % |
| 15 | 73.25 % |

Tabela 4.2: Tabela de Probabilidade do vértice marcado ser medido para valores arbitrários de $N$.

A Figura 4.7 mostra a mesma informação apresentada na Tabela 4.2, porém em gráficos em barra para melhor visualização.

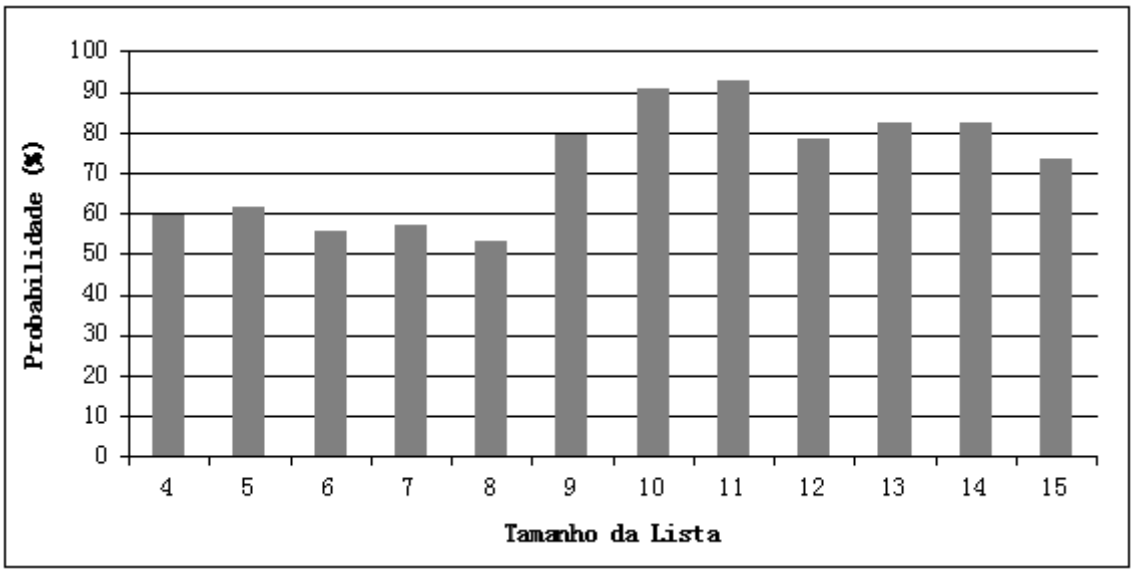


Figura 4.7: Gráfico de probabilidade do vértice marcado ser medido para valores arbitrários de $N$.

# 5 Conclusões

Neste trabalho foi apresentado um simulador numérico para o algoritmo de distinção de elementos proposto por Ambainis, cuja proposta principal é servir de ferramenta de apoio para os estudos e análises das técnicas e conceitos usados por este algoritmo. O grande problema de uma abordagem analítica para compreender o algoritmo de Ambainis está no crescimento exponencial a medida que a lista de entrada aumenta, tornando esta abordagem impraticável para caso onde a quantidade de elementos na lista de entrada não chega a ser muito grande. A simulação de algoritmos quânticos em computadores clássicos, de maneira geral, também possui complexidade exponencial, tanto em consumo de memória quanto em tempo necessário para se concluir uma simulação. Todavia, a execução de simulações de propósito educacional permite que seja possível estudar o comportamento dos algoritmos quânticos em situações diversas. Por este motivo, muitas ferramentas de simulação foram desenvolvidas, no entanto, nenhuma delas aborda especificamente o algoritmo de distinção de elementos.

O algoritmo de Ambainis faz uso de conceitos importantes e complexos do paradigma quântico, como caminhadas quânticas sobre o grafo de Johnson e busca quântica em grafos. Em virtude destes fatos, compreender o algoritmo de Ambainis pode contribuir no desenvolvimento de novas técnicas e algoritmos quânticos para o paradigma quântico. Neste trabalho foi desenvolvido um simulador numérico com a proposta de que sirva de ferramenta educacional no auxílio aos estudos de como os estados do algoritmo de distinção de elementos evolui ao longo de sua execução. Primeiramente, cada passo do algoritmo foi estudado, trabalhando de forma analítica na execução de cada passo do algoritmo para que posteriormente fosse implementado cada passo de forma que potenciais estudantes pudessem compreender o funcionamento tanto do simulador quanto do algoritmo.

Uma interface gráfica está em fase de desenvolvimento, de forma a tornar o simulador mais intuitivo e fácil de ser utilizado. Como projetos futuros, tem-se a intenção de estender as funcionalidades do simulador para abranger os caso em que o número de elementos repetidos $k$ seja arbitrário, e o caso de múltiplas $k$-colisões.

# Referências Bibliográficas


[1] Aaronson, S., Shi, Y., *Quantum lower bounds for the collision and the element distinctness problems*. Journal of the ACM, 51:595-605, 2004.

[2] Abreu, A. S., Ferreira, M. M., Kowada, L. A. B., Marquezino, F. L., *Numeric Simulation of Quantum Algorithm for Elements Distinctness*. Submetido ao WECIQ V. 2014.

[3] Aharonov, Y., Davidovich, L., Zagury, N., *Quantum random walks*, Physical Review A, 48(2):1687–1690, 1993.

[4] Ambainis, A., *Quantum Walk Algorithm for Element Distinctness*. SIAM Journal on Computing, 37(1):210-239, 2007.

[5] Barbosa, A. A., Junior, B. L., Lima, A. F., *Zeno – Simulador Simbólico de Circuitos Quânticos*. WECIQ, 2007.

[6] Bartlett, S. D., Sanders, B. C., Braunstein, S. L., Nemoto, K., *Efficient Classical Simulation of Continuous Variable Quantum Information Processes*. 2002.

[7] Berry, S. D., Bourke, P., Wang, J. B., *qwViz: visualisation of quantum walks on graphs*. Computer Physics Communications. Elsevier, ISSN 0010-4655. Volume 182, Issue 10, pp 2295-2302, October 2011.

[8] Bookatz, A. D., Wocjan, P., Viola, L., *Hamiltonian Quantum Simulation with Bounded-Strength Controls*. New Journal of Physics. 16, 045021, 2014.

[9] Figueiredo, F. D. S., *Simulador de Circuitos Quânticos*, 2013. Dissertação (Mestrado em Engenharia Electrotécnica e de Computadores) - Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2013.

[10] Grover, L., *A fast quantum mechanical algorithm for database search*. Proceedings of STOC'96, pp.212-219.

[11] Kaye, P., Laflamme, R., Mosca, M., *An Introduction to Quantum Computing*, Editora Oxford, 2007.

[12] Kowada, L. A. B., *Construção de Algoritmos Reversíveis e Quânticos*, 2006. Tese (Doutorado em Ciências em Engenharia de Sistemas e Computação) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006.

[13] Marquezino, F. L., *Análise, Simulações e Aplicações Algorítmicas de Caminhadas Quânticas*, 2010. Tese (Doutorado em Ciências em Modelagem Computacional) - Laboratório Nacional de Computação Científica, Petropólis, Rio de Janeiro, 2010.

[14] Marquezino, F. L., Portugal, R., *The QWalk Simulator of Quantum Walks*. Computer Physics Communications. Volume 179, 2008

[15] Nielsen, M. A., Chuang, I. L., *Quantum Computation and Quantum Information*, Editora Cambridge, 2000.

[16] O'Connor, J. J., Robertson, E. F., John von Neumann, Disponível em: http://www-history.mcs.st-andrews.ac.uk/Biographies/Von_Neumann.html. Acesso em: 18 dez. 2013.

[17] Portugal, R., Lavor, C. C., Carvalho, L. M., Maculan, N., *Uma Introdução à Computação Quântica*, SBMAC, 2004.

[18] Santos, R. A. M., *Algoritmos baseados em cadeias de Markov quânticas*, 2014. Tese (Doutorado em Ciências em Modelagem Computacional) - Laboratório Nacional de Computação Científica, Petropólis, Rio de Janeiro, 2014.

[19] Szegedy, M., *Quantum speed-up of markov chain based algorithms*, In: Proce- edings of the 45th Symposium on Foundations of Computer Science, páginas 32–41, 2004.

[20] Viamontes, G. F., Markov, I. L., Hayes, J. P., *Quantum Circuit Simulation*, Springer Verlag, 2009.

[21] Viamontes, G. F., Markov, I. L., Hayes, J. P., *Graph-based Simulation of Quantum Computation in the Density Matrix Representation*. Quantum Information & Computation. 5 (2), 113-130, 2005.

# Apêndice A

# Pseudocódigo do algoritmo de Ambainis

O algoritmo original como foi proposto por Ambainis [4] em seu artigo é apresentado neste apêndice, concentrando-se no caso em que existem apenas dois elementos iguais, que é o caso abordado neste trabalho. Primeiro é apresentado o **Algoritmo 1**, que representa o fluxo principal do programa onde ocorre a preparação do estado inicial para que seja feita a caminhada quântica, descrita no **Algoritmo 2**, e posteriormente a medição.

---

**Algoritmo 1:** Algoritmo de Solução Única.

---

1 Gere a superposição uniforme

$$\frac{1}{\sqrt{\binom{N}{r}(N-r)}} \sum_{|S|=r, y \notin S} |S\rangle |y\rangle .$$

2 Consulte todos os $x_i$ para $i \in S$. Isto faz com que o estado se transforme em

$$\frac{1}{\sqrt{\binom{N}{r}(N-r)}} \sum_{|S|=r, y \notin S} |S\rangle |y\rangle \bigotimes_{i \in S} |x_i\rangle .$$

3 Repita $t_1 = O((\frac{N}{r})^{k/2})$ vezes:
(a) Aplique o operador condicional de inversão de fase (Transforma $|S\rangle |y\rangle |x\rangle \rightarrow -|S\rangle |y\rangle |x\rangle$) para todo $S$ de tal modo que $x_{i_1} = x_{i_2} = ... = x_{i_k}$ para $k$ distintos $i_1, ..., i_k \in S$.

(b) Realize $t_2 = O(\sqrt{r})$ passos da Caminhada Quântica (algoritmo 2).

4 Efetue a medição do estado final. Verifique se $S$ contém uma k-colisões e responda "Existe uma k-colisões"ou "não existe k-colisões", de acordo com o resultado.

---

O **Algoritmo 2** apresenta a caminhada quântica, e esta está dividida em seis passos. Os três primeiros passos da caminhada são responsáveis por mover o caminhante do espaço de Hilbert $\mathcal{H}$ para o espaço $\mathcal{H}'$, enquanto os três últimos passos movem o caminhante de volta ao espaço inicial.

**Algoritmo 2:** Um Passo da Caminhada Quântica.

1 Aplique a transformação mapeando $|S\rangle\,|y\rangle$ para

$$|S\rangle\left(\left(-1+\frac{2}{N-r}\right)|y\rangle+\frac{2}{N-r}\sum_{y'\notin S,y'\neq y}|y'\rangle\right),$$

onde $S$ e $y$ são registradores do estado $\mathcal{H}$. (Essa transformação é uma variação da "transformação de difusão" em [10].)

2 Mapeie o estado de $\mathcal{H}$ para $\mathcal{H}'$, adicionando $y$ ao $S$ e alterando $x$ para um vetor de $k+1$ posições, introduzindo 0 no local correspondente ao $y$.

3 Busque por $x_y$ e insera-o no local de $x$ correspondente ao $y$.

4 Aplique a transformação mapeando $|S\rangle\,|y\rangle$ para

$$|S\rangle\left(\left(-1+\frac{2}{r+1}\right)|y\rangle+\frac{2}{r+1}\sum_{y'\in S,y'\neq y}|y'\rangle\right)$$

no registrador $y$.

5 Apague o elemento de $x$ correspondente ao novo $y$ usando a entrada para consultar $x_y$.

6 Mapeie o estado de volta para $\mathcal{H}$ removendo o componente 0 correspondente ao $y$ de $x$ e removendo $y$ de $S$.

A explicação mais detalhada a respeito deste algoritmo pode ser encontrada no Capítulo 3.