LATIN.SCIENCE XVI Congresso Latino-americano de Software Livre e Tecnologias Abertas LATINOWARE 2019

Quantum Development Kit: ambiente open source para estudo de Computação Quântica

Mateus Karvat Camara
Centro de Ciências Exatas e Tecnológicas
UNIOESTE
Cascavel, Brasil
Email: mateus.camara@unioeste.br

Adriana Postal
Centro de Ciências Exatas e Tecnológicas
UNIOESTE
Cascavel, Brasil
Email: adriana.postal@unioeste.br

***Abstract*—A Computação Quântica é uma área multidisciplinar que faz uso da Matemática, Física e Ciência da Computação, se valendo de conceitos com alto nível de complexidade em seus estudos. Tendo isso em vista, o presente trabalho busca elucidar os conceitos centrais dessa nova área do conhecimento, a fim de prover uma sólida base teórica a estudos posteriores acerca do tema. Os conceitos a serem estudados são: qubits, superposição, portas lógicas quânticas e circuitos quânticos. Visto que a implementação de tais conceitos se faz essencial a este estudo, o ambiente open source Quantum Development Kit será utilizado para analisar o comportamento de portas lógicas quânticas, qubits, da superposição de estados e do algoritmo de Grover. Este algoritmo receberá destaque em virtude do ganho significativo que ele apresenta em relação a algoritmos não quânticos que solucionam o mesmo problema.**

***Keywords*-Quantum Development Kit; Open source; Computação Quântica; Qubit;**

I. Introdução

Computação Quântica é um modelo computacional que tem como base a Mecânica Quântica em detrimento da elétrica utilizada em computadores tradicionais. Enquanto um computador convencional realiza seus cálculos utilizando bits (0 ou 1), um computador quântico realiza seus cálculos utilizando qubits (bits quânticos), os quais têm uma infinidade de estados possíveis, além dos clássicos estados de 0 e 1 [1].

Valendo-se de propriedades da Mecânica Quântica como superposição e entrelaçamento, os computadores quânticos permitem a realização de cálculos em tempo muito menor que o levado por computadores tradicionais (até mesmo supercomputadores) [1].

Computadores quânticos prometem revolucionar a criptografia atual, a qual é baseada na fatoração de grandes números. Enquanto um computador tradicional levaria muitos anos efetuando os cálculos necessários para descobrir a chave de um sistema de criptografia, um computador quântico realizaria essas operações em minutos [1]. Uma criptografia pós-quântica utilizaria dados quânticos para o compartilhamento de chaves, sendo um sistema ainda mais seguro que o utilizado atualmente [2].

Além disso, a simulação de sistemas quânticos (como moléculas e proteínas) promete ser enormemente aprimorada com o uso de computadores quânticos. Um exemplo utilizado por [2] é o de um sistema quântico composto por 50 qubits. Esse sistema é composto por 2^{50}, ou 10^{15}, amplitudes complexas. Atribuindo precisão de 128 bits para cada amplitude, seriam necessários 32 petabytes para simular esse sistema num computador tradicional. Em contrapartida, bastariam 50 qubits para simular esse sistema num computador quântico.

Visto que qubits são entidades matemáticas, computadores quânticos podem ser realizados por quaisquer sistemas físicos que tenham as mesmas propriedades que qubits. Em outras palavras, quaisquer sistemas físicos que apresentem propriedades quânticas [2].

O presente trabalho foi desenvolvido por meio de pesquisa bibliográfica à literatura existente acerca do tema e do uso do ambiente Quantum Development Kit para implementação de portas lógicas quânticas e do Algoritmo de Grover. Na Seção II, serão estudados os conceitos básicos da Computação Quântica: qubits e circuitos quânticos. Em seguida, na Seção III, o ambiente Quantum Development Kit será descrito, sendo seu funcionamento demonstrado pela implementação de circuitos quânticos básicos. Na Seção IV, será estudado o algoritmo de Grover, o qual apresenta ganho significativo em relação a algoritmos clássicos na resolução de problemas de busca não estruturada. Por fim, a Seção V analisa a implementação do algoritmo de Grover no Quantum Development Kit, ressaltando o ganho deste algoritmo em relação a algoritmos clássicos.

II. Conceitos básicos

A. *Qubits*

Qubits são entidades matemáticas, o que significa que eles podem ser estudados isoladamente, não importando o meio físico utilizado para sua realização [2].

Um qubit pode existir numa superposição dos estados clássicos 0 e 1. Essa superposição de estados colapsa ao

ser realizada uma medição, quando o estado se torna 0 ou 1 [1].

A representação mais simples para um qubit é pela notação de Dirac, dada por

$$|\psi\rangle = \alpha|0\rangle + \beta|0\rangle. \quad (1)$$

Nessa notação, α é a amplitude de $|0\rangle$ e β é a amplitude de $|1\rangle$. $|\alpha|^2$ representa a probabilidade de que se obtenha 0 ao medir o qubit e $|\beta|^2$ a probabilidade de se obter 1. É importante ressaltar que α e β são variáveis complexas e que $|\alpha|^2 + |\beta|^2 = 1$ [2].

Um qubit pode também ser representado como um vetor num espaço vetorial bidimensional complexo - noção importante para operações realizadas por portas lógicas quânticas. A notação de um qubit de forma vetorial se dá na forma $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, sendo α e β as mesmas variáveis da notação de Dirac [2].

A superposição, de modo contra intuitivo, não ocorre apenas para um qubit isolado, mas para um grupo de qubits [2]. Um par de qubits, por exemplo, pode existir em superposição, de modo que cada estado computacional possível tem uma amplitude associada:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle. \quad (2)$$

Assim, um par de qubits apresenta novos estados possíveis em relação a cada qubit tomado individualmente [3].

Um estado importante para a Computação Quântica é o chamado estado de Bell (também conhecido como par EPR, em homenagem a Einstein, Podolsky e Rosen, que o descobriram) [2]. Dois qubits constituintes de um par EPR têm uma forte relação entre si, que é utilizada em vários algoritmos quânticos. Tomando, por exemplo, o estado

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \quad (3)$$

A probabilidade de se obter o estado 00 na medição é de 50% e a de se obter o estado 11 também é de 50%. Essa correlação permite o uso de propriedades únicas, inexistentes fora da Mecânica Quântica [2]. Além do estado β_{00}, há também os estados

$$|\beta_{10}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \quad (4)$$

$$|\beta_{01}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \quad (5)$$

$$|\beta_{11}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \quad (6)$$

B. Circuitos quânticos

Tal qual na computação tradicional, a Computação Quântica faz uso de portas lógicas para realizar operações em seus dados [3]. Como definido anteriormente, os qubits podem ser representados em forma vetorial. Assim, as portas lógicas quânticas podem ser descritas como matrizes, tal que o funcionamento de uma porta lógica corresponde à multiplicação da matriz correspondente a ela pelo vetor que representa o qubit. Algumas portas quânticas bastante importantes são as portas NOT (7), Z (8) e Hadamard (9).

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (7)$$

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

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (8)$$

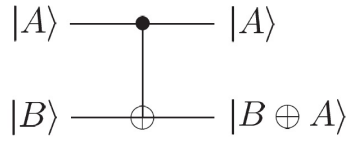
transforma $\alpha|0\rangle + \beta|1\rangle$ em $\alpha|0\rangle - \beta|1\rangle$,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (9)$$

transforma $\alpha|0\rangle + \beta|1\rangle$ em $\alpha\frac{|0\rangle+|1\rangle}{\sqrt{2}} + \beta\frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

Para portas que agem sobre múltiplos qubits, é mais conveniente representá-las através de um circuito. Na representação de um circuito quântico, uma linha representa a passagem do tempo para cada qubit, enquanto quadrados representam portas lógicas. À esquerda da linha tem-se o estado do qubit de entrada, enquanto à direita, tem-se o estado de qubit de saída, após a realização das operações [2].

Figure 1. Porta CNOT (NOT Controlado)



Fonte: [2].

No circuito da Figura 1, o círculo fechado indica que aquele é um qubit de controle. Ele realizará a operação apenas se tiver valor igual a 1. Já o símbolo $\oplus$ representa adição módulo 2 (ou seja, o resto da divisão da soma de A e B por 2). Assim, a porta CNOT inverte B caso o valor de A seja 1, e se o valor de A for 0, B mantém-se intacto [3].

Para exemplificar a passagem de dois qubits por uma porta CNOT, o estado

$$|\psi_0\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \quad (10)$$

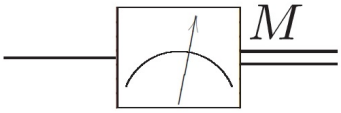
se tornaria o estado

$$|\psi_1\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|11\rangle + \alpha_{11}|10\rangle. \quad (11)$$

Percebe-se que, para as amplitudes α_{10} e α_{11}, em que o primeiro qubit é 1, houve inversão do segundo qubit, de acordo com o princípio de funcionamento da porta CNOT.

Na representação de circuitos quânticos, indica-se a medição de qubits pelo símbolo da Figura 2. Nele, as linhas duplas após o símbolo de medição indicam que o qubit colapsou para seu estado clássico, isto é, para um bit [2].

Figure 2. Medição de qubit



Fonte: [2].

III. Quantum Development Kit

O Quantum Development Kit é um ambiente de programação e simulação de algoritmos quânticos open source desenvolvido pela Microsoft e lançado em Dezembro de 2017 [4]. Ele foi desenvolvido não apenas para que iniciantes à programação quântica possam implementar algoritmos quânticos, mas também para que especialistas possam desenvolver novos algoritmos. O ambiente conta com [5]:

- Linguagem Q# e compilador: A linguagem Q# foi desenvolvida exclusivamente para expressar algoritmos quânticos, sendo utilizada para escrever subprogramas que executam num simulador quântico operando em um computador tradicional;
- Biblioteca Q#: Contém operações e funções que suportam os algoritmos quânticos;
- Simulador quântico local: Um simulador completo de estados quânticos otimizado para simulação rápida e precisa;
- Simulador quântico *trace*: o simulador *trace* não simula o ambiente como o simulador quântico local. Ele é usado para estimar os recursos necessários para execução de um programa quântico e permitir *debugging* mais rápido;
- Estimador de recursos: Estima os recursos necessários para executar uma determinada instância de uma operação em Q# num computador quântico;
- Extensão do Visual Studio e Visual Studio Code: Contém os padrões para os arquivos e projetos Q# bem como suporte para realce sintático.

O passo a passo da instalação do Quantum Development Kit é descrito em detalhes em sua página oficial [6].

A. Portas lógicas no Quantum Development Kit

O site oficial do Quantum Development Kit contém um tutorial para a implementação de portas lógicas simples, como a porta NOT, Hadamard e CNOT, além da realização da medição dos qubits [7]. Analisaremos o funcionamento das portas citadas no ambiente tendo tal tutorial como base.

O projeto contém dois arquivos: o arquivo *Operations.qs*, que contém o código em Q#, e o arquivo *Driver.cs*, que contém o *driver* em C# responsável por construir o simulador quântico, computar os argumentos necessários à execução do algoritmo quântico, rodar o algoritmo quântico e processar os resultados da operação. Cada operação em Q# gera uma classe homônima em C#, a qual contém um método chamado *Run* que executa de forma assíncrona a operação (visto que um computador tradicional é incapaz de realizar operações sobre um grande número de qubits simultaneamente).

O tutorial inicia com a porta NOT e a operação de Medição, as quais são implementadas na `Operação Set`:

`Operação Set`

```
1 operation Set (desired: Result, q1: Qubit) : Unit {
2         if (desired != M(q1)) {
3             X(q1);
4         }
5     }
```

Fonte: [7].

A `Operação Set` inicializa um qubit $q1$ no estado *desired*. Para tanto, ela realiza a medição do qubit (`Set` - Linha 2: $M(q1)$), o qual é invertido pela porta NOT caso o resultado seja distinto do valor do parâmetro *desired* (`Set` - Linha 3: $X(q1)$).

`Operação BellTest`

```
1  operation BellTest (count : Int, initial: Result) :
2          (Int, Int) {
3
4          mutable numOnes = 0;
5          using (qubit = Qubit()) {
6
7              for (test in 1..count) {
8                  Set(initial, qubit);
9                  let res = M(qubit);
10
11                 if (res == One) {
12                     set numOnes += 1;
13                 }
14             }
15             Set(Zero, qubit);
16         }
17
18         return (count-numOnes, numOnes);
19     }
```

Fonte: [7].

A `Operação BellTest` é utilizada para facilitar a visualização das operações realizadas. A partir dos

parâmetros *count* e *Init* inseridos no *driver*, ela inicializa 1000 qubits no estado 0 e 1000 qubits no estado 1. Para cada estado 1 medido (`BellTest` - Linha 9), a variável *numOnes* é incrementada, tal que, ao final da operação, o número de estados 0 e 1 medidos é exibido. A saída obtida demonstra que as operações foram realizadas com sucesso, visto que o parâmetro *Init* determina o estado de todos os qubits a serem medidos:

```
Init:Zero  0s=1000  1s=0
Init:One   0s=0     1s=1000
Press any key to continue...
```

O funcionamento da porta NOT é explicitado caso ela seja implementada (utilizando o comando $X(qubit)$) imediatamente antes da medição (`BellTest` - Linha 9). A saída obtida demonstra a inversão dos estados:

```
Init:Zero  0s=0     1s=1000
Init:One   0s=1000  1s=0
Press any key to continue...
```

Tendo isso em vista, ao implementar a porta Hadamard antes da medição (substituindo a implementação da porta NOT por $H(qubit)$), cria-se superposição e o resultado esperado é o qubit $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, onde há probabilidade de 50% de que os estados 0 e 1 sejam medidos. Novamente, a saída obtida comprova o sucesso da operação (vale ressaltar que a probabilidade ocorre para cada medição, tal que o valor obtido não é igual a 500, mas muito próximo a ele):

```
Init:Zero  0s=502   1s=498
Init:One   0s=496   1s=504
Press any key to continue...
```

Por fim, para demonstrar a propriedade do entrelaçamento, o tutorial acrescenta um segundo qubit $q1$, o qual é inicia-lizado no estado zero (`Entrelaçamento` - Linha 9). O primeiro qubit $q0$ é colocado em superposição (`Entrelaçamento` - Linha 11) e ambos passam por uma porta CNOT (`Entrelaçamento` - Linha 12). Após essas operações, os qubits $q0$ e $q1$ estarão entrelaçados, tal que as medições realizadas sobre ambos devem trazer os mesmos resultados. A fim de verificar esse resultado, atribui-se à variável *res* o resultado da medição do qubit $q0$ (`Entrelaçamento` - Linha 12), a qual é comparada com o resultado da medição do qubit $q1$ (`Entrelaçamento` - Linha 15), incrementando uma variável *agree* caso as medições sejam coincidentes.

`Entrelaçamento`

```
1  operation BellTest(count:Int, initial:Result):
2          (Int, Int, Int){
3          mutable numOnes = 0;
4          mutable agree = 0;
5          using ((q0, q1) = (Qubit(), Qubit())) {
6              for (test in 1..count)
7              {
8                  Set (initial, q0);
9                  Set (Zero, q1);
10
11                 H(q0);
12                 CNOT(q0,q1);
13                 let res = M (q0);
14
15                 if (M (q1) == res) {
16                     set agree += 1;
17                 }
18
19                 if (res == One) {
20                     set numOnes += 1;
21                 }
22
23             }
24
25             Set(Zero, q0);
26             Set(Zero, q1);
27         }
28
29         return (count-numOnes, numOnes, agree);
30     }
```

Fonte: [7].

A saída obtida apresenta o resultado esperado, com a varíavel *agree* apresentando valor 1000:

```
Init:Zero  0s=526   1s=474   agree=1000
Init:One   0s=512   1s=488   agree=1000
Press any key to continue...
```

Comprova-se, assim, o funcionamento do ambiente Quantum Development Kit na simulação de portas quânticas e circuitos simples.

IV. Algoritmo de Grover

O algoritmo de Grover é também chamado de algoritmo quântico de busca por realizar busca não estruturada, isto é, a busca em um conjunto de entradas sem ordenação prévia. Em um problema de busca não estruturada, tem-se uma função booleana $f : X \to \{0, 1\}$ atuando sobre um conjunto $X = \{x_1, x_2, ..., x_N\}$ de N elementos tal que o objetivo é encontrar os M elementos x^* pertencentes a X tal que $f(x^*) = 1$ [8]. Em outras palavras, tem-se M elementos que satisfazem uma propriedade desejada (indicados por x^*) em um conjunto de N elementos, tal que a função f retorna 1 para estes M elementos. Visto que a busca a ser realizada pelo algoritmo é não estruturada, algoritmos clássicos requerem um tempo de ordem $\Theta(N/M)$ para encontrar o elemento x^*. O algortimo de Grover, por sua vez, utiliza de propriedades quânticas e requer apenas um tempo de ordem $\Theta(\sqrt{N/M})$ para encontrar o elemento [2].

Para realizar a busca, analisaremos os elementos com base em seus índices, de modo que o conteúdo destes elementos não é relevante para o algoritmo em si, mas sim para a função $f(x)$, que operará com base nos índices destes elementos e identificará aqueles que atendem à propriedade desejada. Assumindo que $N = 2^n$ (caso N não seja uma potência de 2, podemos acrescentar lixo ao conjunto até

que se tenha um n inteiro que satisfaça essa igualdade [8]), precisaremos de n qubits para armazenar os índices. Assim, a função descrita anteriormente é simplificada para $f : \{0,1\}^n \to \{0,1\}$.

O algoritmo inicia com n qubits inicializados em $|0\rangle$. O primeiro passo é criar uma superposição de todos os estados possíveis, a qual é realizada aplicando a porta Hadamard sobre esse conjunto de qubits [9]:

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \sum_{x=0}^{2^n-1} \frac{1}{\sqrt{N}}|x\rangle. \tag{12}$$

A seguir, realiza-se uma série de transformações conhecida como *Iteração de Grover* ou *Operador de Grover* [2]. Nela, cada iteração se dá pela aplicação de um oráculo que realiza a função f sobre os qubits superpostos, seguida da realização da *transformada difusa*, a qual realiza inversão em torno da média. A Iteração de Grover é descrita por diferentes autores utilizando as diferentes representações de qubits e operações sobre qubits: [10] e [11] utilizam a notação angular, [9] utiliza a notação algébrica, [8] utiliza a notação artimética e [2] aborda todas as três notações. Tomaremos como base a notação aritmética.

O oráculo a ser utilizado é uma "caixa-preta", isto é, um conjunto de operadores que transforma uma determinada entrada na saída esperada, sem que se faça necessário o conhecimento de seu funcionamento interno. A utilização do oráculo pelo algoritmo de Grover é um ponto positivo [10], pois ele pode ser aplicado a qualquer problema de busca, independente dos tipos de dados a serem analisados, visto que cada problema utilizaria um oráculo distinto. Utilizando as notações quânticas, pode-se descrever o oráculo como um operador O:

$$|\psi\rangle|q\rangle \xrightarrow{\text{O}} |\psi\rangle|q \oplus f(x)\rangle, \tag{13}$$

em que um qubit $|q\rangle$ é invertido se $f(x) = 1$ e mantém-se intacto caso contrário. Nota-se que o objetivo do oráculo é implementar a função $f(x)$ sobre o conjunto inicial de qubits, destacando os elementos desejáveis do conjunto de dados.

É, então, conveniente utilizar $|q\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$, pois a inversão resultará no estado $-|q\rangle$ (a inversão de $|q\rangle$ resulta em $\frac{-|0\rangle+|1\rangle}{\sqrt{2}} = -|q\rangle$). Tendo isso em vista, pode-se escrever a ação do oráculo na forma

$$|\psi\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \xrightarrow{\text{O}} (-1)^{f(x)}|\psi\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right). \tag{14}$$

Pode-se observar que o qubit $|q\rangle$ não é alterado pela atuação do oráculo, tal que ele pode ser omitido de (14), resultando em

$$|\psi\rangle \xrightarrow{\text{O}} (-1)^{f(x)}|\psi\rangle. \tag{15}$$

A partir de (15), dizemos que o oráculo *marca* as soluções para o problema ao inverter a fase da solução [2].

Após a aplicação do oráculo sobre o conjunto de qubits $|\psi\rangle$, realiza-se a *transformada difusa* [8], que é descrita aritmeticamente por

$$\sum_{x=0}^{2^n-1} \alpha_x|x\rangle \mapsto \sum_{x=0}^{2^n-1} (2\mu - \alpha_x)|x\rangle. \tag{16}$$

Em (16), α_x é a amplitude de um qubit x e μ é a média das amplitudes de todos os qubits [8], a qual é dada por

$$\mu = \frac{1}{N}\sum_{x=0}^{2^n-1} \alpha_x. \tag{17}$$

A partir disso, a *Iteração de Grover*, que consiste na aplicação sucessiva dos passos descritos em (15) e (16), deve ser realizada por um número R de vezes, sendo R definido por

$$R \leq \left\lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rceil. \tag{18}$$

Por fim, realiza-se a medição do estado resultante [2].

Para melhor compreender o algoritmo, tomemos o exemplo dado por [9] de um sistema contendo $N = 8$ estados em que se busca um estado x^* de índice 011. Visto que utilizaremos o Algoritmo de Grover para encontrar tal elemento, devemos verificar se N é potência de 2, ou seja, se existe n tal que $N = 2^n$. Como $2^3 = 8$, então $n = 3$ e não é necessário acrescentar lixo ao conjunto. Temos, então, uma função booleana $f : \{0,1\}^3 \to \{0,1\}$, em que $f(011) = 1$ e $f(i) = 0$ para todos os outros índices i diferentes de 011.

O algortimo de Grover inicia com n qubits inicializados em $|0\rangle$, tal que, sendo $n = 3$, teremos o estado

$$|\psi_0\rangle = 1|000\rangle. \tag{19}$$

O próximo passo é aplicar a porta Hadamard ao estado $|\psi_0\rangle$, o que criará uma superposição de todos os estados possíveis para 3 qubits, tendo cada estado a mesma amplitude. Conforme (12), o estado resultante será igual ao somatório dos estados x com amplitude $\frac{1}{\sqrt{N}}$. Como $N = 8$, $\sqrt{N} = 2\sqrt{2}$. Além disso, x varia de 0 a 2^n-1, de modo que temos x variando de 0 a 7 (que são representados, em bits, como 000 e 111). Logo, essa superposição é representada em

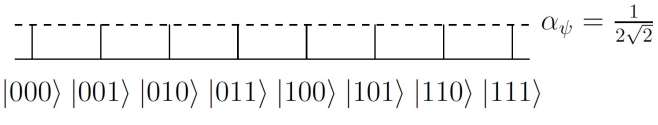
$$H^3|000\rangle = \frac{1}{2\sqrt{2}}|000\rangle + \frac{1}{2\sqrt{2}}|001\rangle + \ldots + \frac{1}{2\sqrt{2}}|111\rangle = \sum_{x=0}^{7}\frac{1}{2\sqrt{2}}|x\rangle = |\psi_1\rangle. \tag{20}$$

No estado $|\psi_1\rangle$, a média das amplitudes μ_1 pode ser calculada somando-se todas as amplitudes e dividindo pelo número de estados (conforme (17)):

$$\mu_1 = \frac{1}{8}\sum_{x=0}^{7}\alpha_x = \frac{1}{8}\frac{8}{2\sqrt{2}} = \frac{1}{2\sqrt{2}}. \quad (21)$$

Tendo isso em vista, o estado $|\psi_1\rangle$ é graficamente representado na Figura 3, em que a amplitude de cada estado é representado por uma barra vertical proporcional à amplitude daquele estado. Como todas as amplitudes são iguais, todas as barras verticais têm mesmo comprimento, o qual é equivalente a α_ψ.

Figure 3. Representação do estado $|\psi_1\rangle$



Fonte: [9].

Antes de iniciar as iterações de Grover, é importante verificar quantas iterações devem ser realizadas. Utilizando (18), tem-se que 2 iterações de Grover devem ser realizadas, ou seja,

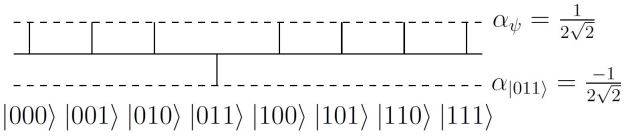
$$R \leq \left\lceil \frac{\pi}{4}\sqrt{\frac{8}{1}} \right\rceil = \left\lceil \frac{\pi 2\sqrt{2}}{4} \right\rceil \approx 2.22. \quad (22)$$

Iniciando a primeira iteração, aplica-se o oráculo sobre o estado $|\psi_1\rangle$, o qual, conforme (15), inverterá a amplitude de todos os elementos em que $f(x) = 1$ e manterá a mesma amplitude para todos os elementos em que $f(x) = 0$. Como descrito anteriormente, apenas o estado $|011\rangle$ satisfaz a função booleana do oráculo, originando o estado

$$|\psi_2\rangle = \frac{1}{2\sqrt{2}}|000\rangle + \frac{1}{2\sqrt{2}}|001\rangle + \frac{1}{2\sqrt{2}}|010\rangle - \frac{1}{2\sqrt{2}}|011\rangle + ... + \frac{1}{2\sqrt{2}}|111\rangle. \quad (23)$$

Nota-se, no estado $|\psi_2\rangle$, que a amplitude de todos os estados é $\frac{1}{2\sqrt{2}}$, exceto do estado $|011\rangle$, em que a amplitude é $-\frac{1}{2\sqrt{2}}$. Por esse motivo, a representação gráfica deste estado posicionará a barra vertical do estado $|011\rangle$ abaixo do eixo, o que pode ser observado na Figura 4. Percebe-se que os demais estados não foram alterados.

Figure 4. Representação do estado $|\psi_2\rangle$



Fonte: [9].

A mudança no estado $|011\rangle$ afeta, porém, o valor da média das amplitudes. Assim, a média das amplitudes μ_2 para o estado $|\psi_2\rangle$ é calculada por

$$\mu_2 = \frac{1}{8}\sum_{x=0}^{7}\alpha_x = \frac{1}{8}\frac{6}{2\sqrt{2}} = \frac{3}{8\sqrt{2}}. \quad (24)$$

A aplicação da transformada difusa (descrita em (16)) a $|\psi_2\rangle$ resulta em

$$\sum_{x=0}^{7}\alpha_x|x\rangle \mapsto \sum_{x=0}^{7}(2\frac{3}{8\sqrt{2}} - \alpha_x)|x\rangle. \quad (25)$$

Sabendo-se que α_x representa a amplitude do estado x e que todos os estados, exceto $|011\rangle$, têm amplitude $\frac{1}{2\sqrt{2}}$, a transformada difusa alterará todos esses estados igualmente, reduzindo sua amplitude:

$$2\frac{3}{8\sqrt{2}} - \frac{1}{2\sqrt{2}} = \frac{3}{4\sqrt{2}} - \frac{1}{2\sqrt{2}} = \frac{1}{4\sqrt{2}}. \quad (26)$$

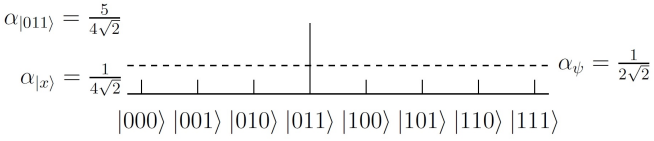
Todavia, para o estado $|011\rangle$, a amplitude α_{011} é igual a $-\frac{1}{2\sqrt{2}}$, tal que sua transformada difusa será distinta da transformada dos demais estados:

$$2\frac{3}{8\sqrt{2}} - \left(-\frac{1}{2\sqrt{2}}\right) = \frac{3}{4\sqrt{2}} + \frac{1}{2\sqrt{2}} = \frac{5}{4\sqrt{2}}. \quad (27)$$

Devido a isso, a aplicação da transformada difusa sobre $|\psi_2\rangle$ resulta em $|\psi_3\rangle$, o qual é representado na Figura 5 e por

$$|\psi_3\rangle = \frac{1}{4\sqrt{2}}|000\rangle + \frac{1}{4\sqrt{2}}|001\rangle + \frac{1}{4\sqrt{2}}|010\rangle + \frac{5}{4\sqrt{2}}|011\rangle + ... + \frac{1}{4\sqrt{2}}|111\rangle. \quad (28)$$

Figure 5. Representação do estado $|\psi_3\rangle$



Fonte: [9].

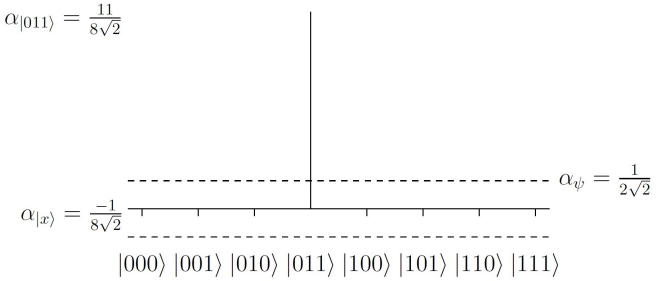
Encerrada a primeira iteração de Grover, realiza-se a segunda iteração. Assim como na primeira, efetua-se a aplicação do oráculo sobre o estado e então a transformada difusa. A segunda iteração tem como resultado o estado $|\psi_4\rangle$, representado na Figura 6 e por

$$|\psi_4\rangle = -\frac{1}{8\sqrt{2}}|000\rangle - \frac{1}{8\sqrt{2}}|001\rangle - \frac{1}{8\sqrt{2}}|010\rangle + \frac{11}{8\sqrt{2}}|011\rangle - ... - \frac{1}{8\sqrt{2}}|111\rangle. \quad (29)$$

Destaca-se a amplitude significativamente maior do estado $|011\rangle$ em relação aos demais estados.

Após as duas iterações, efetua-se a medição, na qual o estado $|011\rangle$ tem probabilidade $121/128 \approx 94.5\%$ de ser medido, enquanto a probabilidade de encontrar um estado incorreto é de $7/128 \approx 5.5\%$. Assim, a probabilidade de

Figure 6. Representação do estado $|\psi_4\rangle$



Fonte: [9].

se ter uma medição correta é 17 vezes maior de que a probabilidade de se obter uma medição incorreta. Ainda que o algoritmo de Grover seja probabilístico, para $N \geq 3$, o maior erro é de $\approx 5.5\%$ e ele tende a 0 conforme N tende ao infinito [11].

V. Algoritmo de Grover no Quantum Development Kit

Muito além de simples portas lógicas quânticas, o Quantum Development Kit pode executar algoritmos quânticos complexos como o Algoritmo de Grover. Uma implementação deste algoritmo é disponibilizado pela própria Microsoft [12] e será utilizada neste trabalho. A implementação disponibilizada foi ligeiramente modificada para efetuar 1000 tentativas para todos os algoritmos, bem como calcular o número de iterações e acessos ao oráculo necessários com base na literatura existente. Tal alteração só foi possível em virtude da natureza open source da plataforma. Considerando tal implementação, focaremos na saída apresentada pelo programa, a qual descreve satisfatoriamente o funcionamento do algoritmo de Grover no ambiente, comparando-o com um algoritmo clássico de busca não estruturada.

Primeiramente, o programa executa uma busca não estruturada utilizando um algoritmo clássico que seleciona aleatoriamente um dos elementos pertencentes ao conjunto em que se realiza a busca. Esse algoritmo não realizará ordenação dos elementos do conjunto nem eliminará os elementos indesejados de futuras buscas, de modo que o espaço de busca é sempre o mesmo, possibilitando uma melhor comparação entre o poder computacional dos computadores tradicionais com computadores quânticos.

O conjunto considerado tem 8 elementos, tal que a probabilidade de sucesso para cada tentativa é de 0.125, ou 12.5%. A partir disso, 1000 tentativas são realizadas, havendo a exibição do resultado a cada 100 tentativas, em que é exibido o status de sucesso da tentativa (One para sucesso e Zero para falha) e qual foi o elemento encontrado, com base em seu índice (a tentativa 99, por exemplo, encontrou o elemento de índice 111). A probabilidade exibida a cada 100 tentativas é a razão entre o número de tentativas bem sucedidas e o número total de tentativas realizadas. Nota-se que o valor da probabilidade se mantém próximo ao valor esperado, mas, se tratando de um cenário probabilístico, o valor nunca é exatamente o mesmo.

Busca clássica

```
Classical random search for marked element in
 database.
  Database size: 8.
  Success probability:  0,125

Attempt 99. Success: One,  Probability: 0,14
Found database index One, One, One
Attempt 199. Success: Zero,  Probability: 0,16
Found database index One, One, Zero
Attempt 299. Success: Zero,  Probability: 0,15
Found database index Zero, One, One
Attempt 399. Success: Zero,  Probability: 0,162
Found database index One, Zero, One
Attempt 499. Success: Zero,  Probability: 0,162
Found database index One, One, Zero
Attempt 599. Success: Zero,  Probability: 0,15
Found database index Zero, Zero, One
Attempt 699. Success: One,  Probability: 0,146
Found database index One, One, One
Attempt 799. Success: Zero,  Probability: 0,142
Found database index Zero, One, One
Attempt 899. Success: Zero,  Probability: 0,142
Found database index One, Zero, One
Attempt 999. Success: Zero,  Probability: 0,139
Found database index Zero, One, One


Press any key to continue...
```

Tendo em vista a natureza probabilística da busca realizada, convém executar o programa novamente a fim de verificar se os valores de probabilidade encontrados em cada tentativa são distintos, o que se comprova na `Busca clássica - Tentativa 02`:

Busca clássica - Tentativa 02

```
Classical random search for marked element in
 database.
  Database size: 8.
  Success probability:  0,125

Attempt 99. Success: Zero,  Probability: 0,19
Found database index Zero, One, Zero
Attempt 199. Success: Zero,  Probability: 0,16
Found database index One, Zero, Zero
Attempt 299. Success: Zero,  Probability: 0,167
Found database index Zero, Zero, One
Attempt 399. Success: Zero,  Probability: 0,15
Found database index Zero, Zero, Zero
Attempt 499. Success: Zero,  Probability: 0,146
Found database index One, Zero, One
Attempt 599. Success: Zero,  Probability: 0,137
Found database index One, One, Zero
Attempt 699. Success: One,  Probability: 0,139
Found database index One, One, One
Attempt 799. Success: Zero,  Probability: 0,135
Found database index One, Zero, One
Attempt 899. Success: Zero,  Probability: 0,14
Found database index One, Zero, One
```

```
Attempt 999. Success: Zero,  Probability: 0,137
Found database index Zero, One, Zero


Press any key to continue...
```

Tendo efetuado a `Busca clássica` em um espaço de busca pequeno, o programa parte para a realização de uma busca utilizando o algoritmo de Grover. O espaço considerado tem 64 elementos, necessitando de 13 iterações de Grover, conforme (18). A probabilidade de sucesso do algoritmo de busca clássico efetuando a busca de modo aleatório é de 1.5625%, enquanto a probabilidade de sucesso do algoritmo quântico é de $\approx 99.6586\%$. A probabilidade de sucesso do algoritmo quântico é dada por

$$P = sin^2\left(\left(2\left\lfloor\frac{\pi}{4}\sqrt{\frac{N}{M}}\right\rfloor + 1\right) sin^{-1}\left(\sqrt{\frac{M}{N}}\right)\right), \quad (30)$$

em que N representa o tamanho do espaço de busca e M representa o número de elementos que satisfazem a busca (ou seja, o número de elementos desejáveis) [10].

De modo similar à implementação da `Busca clássica`, o programa exibe os elementos encontrados a cada 100 iterações, em que são exibidas a condição de Sucesso e a Probabilidade descritas anteriormente, bem como o *Speedup*, que indica quantas vezes, em média, o algoritmo quântico opera mais rápido que o algoritmo clássico (levando em conta as probabilidades apresentadas e o número de iterações de Grover realizadas). O valor encontrado indica que o algoritmo quântico é aproximadamente 10 vezes mais rápido.

`Busca quântica por um elemento`

```
Quantum search for marked element in database.
  Database size: 64.
  Classical success probability: 0,015625
  Queries per search: 6
  Quantum success probability: 0,996585680786799

Attempt 99. Success: One,
Probability: 1 Speedup: 10,667
Found database index One, One, One, One, One, One
Attempt 199. Success: One,
Probability: 1 Speedup: 10,667
Found database index One, One, One, One, One, One
Attempt 299. Success: One,
Probability: 1 Speedup: 10,667
Found database index One, One, One, One, One, One
Attempt 399. Success: One,
Probability: 1 Speedup: 10,667
Found database index One, One, One, One, One, One
Attempt 499. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One
Attempt 599. Success: One,
Probability: 0,997 Speedup: 10,635
Found database index One, One, One, One, One, One
Attempt 699. Success: One,
Probability: 0,997 Speedup: 10,635
Found database index One, One, One, One, One, One
Attempt 799. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One
Attempt 899. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One
Attempt 999. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One


Press any key to continue...
```

Tal como na `Busca clássica`, obtém-se resultados ligeiramente diferentes ao se efetuar a `Busca quântica por um elemento - Tentativa 02`:

`Busca quântica por um elemento - Tentativa 02`

```
Quantum search for marked element in database.
  Database size: 64.
  Classical success probability: 0,015625
  Queries per search: 6
  Quantum success probability: 0,996585680786799

Attempt 99. Success: One,
Probability: 1 Speedup: 10,667
Found database index One, One, One, One, One, One
Attempt 199. Success: One,
Probability: 1 Speedup: 10,667
Found database index One, One, One, One, One, One
Attempt 299. Success: One,
Probability: 1 Speedup: 10,667
Found database index One, One, One, One, One, One
Attempt 399. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One
Attempt 499. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One
Attempt 599. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One
Attempt 699. Success: One,
Probability: 0,999 Speedup: 10,656
Found database index One, One, One, One, One, One
Attempt 799. Success: One,
Probability: 0,999 Speedup: 10,656
Found database index One, One, One, One, One, One
Attempt 899. Success: One,
Probability: 0,999 Speedup: 10,656
Found database index One, One, One, One, One, One
Attempt 999. Success: One,
Probability: 0,998 Speedup: 10,645
Found database index One, One, One, One, One, One


Press any key to continue...
```

Por fim, é realizada uma busca com o algoritmo de Grover num conjunto de 256 elementos em que há 4 elementos (de índices 0, 39, 101 e 234) que satisfazem a operação. Aqui, a probabilidade de sucesso do algoritmo quântico é a mesma que para a `Busca quântica por um elemento` tendo em vista que (30) apresenta o mesmo valor. Além disso, o algoritmo clássico para este conjunto

tem mesma probabilidade de sucesso que no exemplo anterior, visto que $\frac{4}{256} = \frac{1}{64} = 0,015625$.

Busca quântica por quatro elementos

```
Quantum search for marked element in database.
  Database size: 256.
  Marked elements: 0,39,101,234
  Classical success probability: 0,015625
  Queries per search: 6
  Quantum success probability: 0,996585680786799

Attempt 99. Success: One,
 Probability: 0,98 Speedup: 10,453
Found database index 234
Attempt 199. Success: One,
 Probability: 0,98 Speedup: 10,453
Found database index 101
Attempt 299. Success: One,
 Probability: 0,987 Speedup: 10,528
Found database index 39
Attempt 399. Success: One,
 Probability: 0,99 Speedup: 10,56
Found database index 39
Attempt 499. Success: One,
 Probability: 0,992 Speedup: 10,581
Found database index 101
Attempt 599. Success: One,
 Probability: 0,993 Speedup: 10,592
Found database index 234
Attempt 699. Success: One,
 Probability: 0,993 Speedup: 10,592
Found database index 234
Attempt 799. Success: One,
 Probability: 0,994 Speedup: 10,603
Found database index 234
Attempt 899. Success: One,
 Probability: 0,992 Speedup: 10,581
Found database index 234
Attempt 999. Success: One,
 Probability: 0,993 Speedup: 10,592
Found database index 234

Press any key to continue...
```

Novamente, a execução de uma segunda tentativa apresenta resultados ligeiramente diferentes em virtude da natureza probabilística do problema.

Busca quântica por quatro elementos - Tentativa 02

```
Quantum search for marked element in database.
  Database size: 256.
  Marked elements: 0,39,101,234
  Classical success probability: 0,015625
  Queries per search: 6
  Quantum success probability: 0,996585680786799

Attempt 99. Success: One,
 Probability: 0,99 Speedup: 10,56
Found database index 0
Attempt 199. Success: One,
 Probability: 0,99 Speedup: 10,56
Found database index 39
Attempt 299. Success: One,
 Probability: 0,99 Speedup: 10,56
Found database index 0
Attempt 399. Success: One,
 Probability: 0,99 Speedup: 10,56
Found database index 234
Attempt 499. Success: One,
 Probability: 0,99 Speedup: 10,56
Found database index 101
Attempt 599. Success: One,
 Probability: 0,992 Speedup: 10,581
Found database index 39
Attempt 699. Success: One,
 Probability: 0,993 Speedup: 10,592
Found database index 101
Attempt 799. Success: One,
 Probability: 0,994 Speedup: 10,603
Found database index 101
Attempt 899. Success: One,
 Probability: 0,994 Speedup: 10,603
Found database index 0
Attempt 999. Success: One,
 Probability: 0,995 Speedup: 10,613
Found database index 101

Press any key to continue...
```

VI. Conclusão

A partir da implementação do algoritmo de Grover no ambiente Quantum Development Kit, evidencia-se o poder da Computação Quântica sobre a computação tradicional e atesta-se o alto grau de relevância desta área. Durante a realização deste trabalho, a Computação Quântica ainda se encontrava em fase de pesquisa, sem possuir aplicações de impacto na sociedade. Acredita-se que tais aplicações surgirão após a Computação Quântica alcançar a chamada "supremacia quântica", em que computadores quânticos serão capazes de realizar operações que computadores clássicos são incapazes de realizar. Para se alcançar tal marco, muita pesquisa e investimento devem ser direcionados a essa área.

Tendo isso em vista, percebe-se a necessidade de utilização de softwares livres no estudo e desenvolvimento desta área emergente, a fim de garantir a disseminação do conhecimento acerca dela. O presente trabalho só foi possível graças à existência de um ambiente open source, o qual não apenas proporcionou acesso gratuito a um simulador quântico eficiente, mas também possibilitou o estudo aprofundado do código de implementação dos algoritmos quânticos por ele utilizados. Vale notar que o Quantum Development Kit conta com uma comunidade ativa de desenvolvedores colaborando para o aprimoramento da linguagem Q# e do próprio ambiente.

Agradecimentos

Agradecemos ao MEC-SESU, pelo financiamento deste projeto de iniciação científica a partir do Programa de Educação Tutorial (PET).

REFERENCES

[1] C. H. Bennett. (2016) Information is quantum. [Online]. Available: https://www.youtube.com/watch?v=EqXv40kCahM

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[3] I. L. Chuang and P. Shor. (2018) Quantum information science i, part 1. Massachussets Institute of Technology (MIT). [Online]. Available: https://courses.edx.org/courses/course-v1:MITx+8.370.1x+1T2018/course

[4] M. Q. Team. (2017) Announcing the microsoft quantum development kit. [Online]. Available: https://cloudblogs.microsoft.com/quantum/2017/12/11/announcing-microsoft-quantum-development-kit/

[5] Microsoft. (2017) Welcome to the microsoft quantum development kit preview. [Online]. Available: https://docs.microsoft.com/en-us/quantum/?view=qsharp-preview

[6] ——. (2017) Quantum development kit. [Online]. Available: https://www.microsoft.com/en-us/quantum/development-kit

[7] ——. (2017) Writing a quantum program. [Online]. Available: https://docs.microsoft.com/en-us/quantum/quickstart?view=qsharp-preview&tabs=tabid-vscode

[8] J. Wright, "Lecture 4: Grover's algorithm," Carnegie Mellon University, 2015. [Online]. Available: https://www.cs.cmu.edu/~odonnell/quantum15/lecture04.pdf

[9] E. Strubell, "An introduction to quantum algorithms," *COS498 - Chawathe*, 2011. [Online]. Available: https://people.cs.umass.edu/~strubell/doc/quantum_tutorial.pdf

[10] J. Watrous, "Lecture 13: Grover's algorithm (continued)," University of Calgary, 2006. [Online]. Available: https://cs.uwaterloo.ca/~watrous/LectureNotes/CPSC519.Winter2006/12.pdf

[11] ——, "Lecture 12: Grover's algorithm," University of Calgary, 2006. [Online]. Available: https://cs.uwaterloo.ca/~watrous/LectureNotes/CPSC519.Winter2006/13.pdf

[12] Microsoft. (2017) Microsoft quantum development kit samples. [Online]. Available: https://github.com/Microsoft/Quantum