

Interação e Concorrência

Exercicios Propostos

A98980 Eduardo Cunha

2024

**Exercício 1**

Uma imagem com texto, captura de ecrã, número, Tipo de letra

Descrição gerada automaticamente

Figura 1 - Exercício 1

Para resolver este problema, vamos definir a função booleana em “Qiskit” e, em seguida, implementar o algoritmo de “Deutsch-Jozsa” para identificar se se trata de uma função balanceada ou constante com apenas uma única chamada à função.

Analisando a tabela, foi possível perceber que a função retorna 1 para todas as entradas cujo primeiro bit é 0 e retorna 0 para todas as entradas cujo primeiro bit é 1, sendo, portanto, balanceada.

Após a execução do circuito de “Deutsch-Jozsa”, medimos a função e, se obtivermos:

* O estado |0⟩ - então é a função é constante
* O estado |1⟩ - então é a função é balanceada

Sendo assim, tendo em conta tudo o que foi referido, é de esperar obter 1 após a execução do algoritmo, dado que se trata de uma função balanceada.

Resolução do exercício:

Definimos a função booleana então da seguinte forma:

Uma imagem com Tipo de letra, texto, Gráficos, captura de ecrã

Descrição gerada automaticamente

Figura 2 - fun\_exc1

A função "fun\_exc1" recebe um circuito quântico ("circuit") como argumento e aplica uma porta "CNOT" ("CX") ao circuito, utilizando o "qubit" 0 como controle e o qubit 3 (“ancilla”) como alvo.

Posto isto e tendo em conta o código estudado nas aulas práticas, foi aplicado o algoritmo de “Deutsch Josza” da seguinte forma:

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 3 - deutsch\_josza

Com este código geramos o seguinte circuito:

Uma imagem com texto, captura de ecrã, file, diagrama

Descrição gerada automaticamente

Figura 4 - Circuito gerado

E obtivemos as seguintes medidas:

(A função “execute\_circuit” foi fornecida nas aulas práticas)

(Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente

Figura 5 - Medidas obtidas

Como é possível observar, obtivemos o resultado esperado (1), confirmando que se trata de uma função balanceada.

**Exercício 2**

**a)**

**Uma imagem com texto, Tipo de letra, captura de ecrã, file

Descrição gerada automaticamente**

Figura 6 - Exercício 2 a)

A implementação do algoritmo de “Grover” foi realizada da seguinte forma:

Primeiramente, a implementação da função "Oracle" tem como objetivo marcar o estado desejado. Se esse estado corresponder à solução, o oráculo altera o seu sinal.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 7 - Função “oracle”

Seguidamente foi implementada a função "diffusion\_operator" que tem como objetivo amplificar a amplitude do estado marcado pelo oráculo.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 8 - Função “diffusion\_operator”

Posto isto implementamos o circuito inteiro, de realçar que o número ótimo de iterações é dado por:

**k = floor(pi/4 \* sqrt(N))**

Neste caso, N=16 (porque temos 4 qubits, resultando em 2^4 resultados possíveis).Então:

**k = floor(pi/4 \* sqrt(16)) = floor(pi) = 3**

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 9 - Circuito completo

Através do código anterior foi gerado o seguinte circuito.

Uma imagem com texto, file, Paralelo, diagrama

Descrição gerada automaticamente

Figura 10 - Circuito gerado

E obtivemos os seguintes resultados:

Uma imagem com texto, captura de ecrã, ecrã, número

Descrição gerada automaticamente

Figura 11 - Resultados Obtidos

Tendo em conta os slides teóricos relativos ao algoritmo de “Grover”, obtemos uma medição na base computacional que fornece a solução correta com uma probabilidade de (N-1)/N, ou seja, 15/16, o que é aproximadamente igual a 0.9375. Obtive o resultado de 0.969, o qual considero ser um valor bastante próximo do esperado.

**b)**



Figura 12 - Exercício 2 b)

Para encontrar um dos seguintes elementos fiz as seguintes alterações no código:

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 13 - Função "Grover"

Como se pode verificar, agora, desenvolvi uma função que gera o circuito completo. Esta função recebe uma “string” indicando qual das soluções se pretende procurar e passa esta “string” como parâmetro da função "oracle" previamente definida.

Uma imagem com texto, captura de ecrã, ecrã, número

Descrição gerada automaticamente

Figura 14 - Medição do circuito com entrada "0000"

Uma imagem com texto, captura de ecrã, ecrã, número

Descrição gerada automaticamente

Figura 15 - Medição do o circuito com entrada "0101”

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

Figura 16 - Medição do o circuito com entrada "1101”

Uma imagem com texto, captura de ecrã, ecrã, software

Descrição gerada automaticamente

Figura 17 - Medição do o circuito com entrada "1110”

Estes foram os resultados obtidos e como se pode verificar, ao ser executado três vezes (como calculado na pergunta anterior), todas as medições se aproximaram bastante do valor esperado previamente calculado (0.9375), o que está de acordo com a teoria ensinada nas aulas. Isso confirma a eficácia e eficiência do algoritmo de “Grover”.

**c)**



Uma imagem com texto, captura de ecrã, Tipo de letra, diagrama

Descrição gerada automaticamente

Figura 18 - Exercício 2 c)

Primeiramente implementamos o oráculo em questão:

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura 19 - Função "oracle"

Este oráculo aplica uma operação CNOT onde o último “qubit” é o “qubit” de controle e o “qubit ancilla” é o alvo. Isso significa que o oráculo inverte o estado do qubit “ancilla” se e somente se o último “qubit” for igual a 1, desta forma tenta marcar todos os estados terminados por 1 ou seja: 0001, 0011,0111,1111,0101,1001,1011,1101.

Desta forma, com a aplicação do algoritmo de “Grover”, seria de esperar obter percentagens mais altas do que as dos demais e bastante idênticas para esses estados.

Codificação do restante circuito:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 20 - Função "diffusion\_operator"

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura 21 - Criação do circuito completo

Então, ao criar o circuito, com apenas uma alteração no número ideal de iterações, dado que agora há 8 estados possíveis de resposta, obtemos o seguinte circuito:

Uma imagem com texto, captura de ecrã, file, diagrama

Descrição gerada automaticamente

Figura 22 - Esquema do circuito gerado

Medimos os seguintes resultados:

Uma imagem com texto, captura de ecrã, número, Paralelo

Descrição gerada automaticamente

Figura 23 - Medição dos resultados

Isto vai contra o esperado, visto que todos os estados possuem uma percentagem alta. O problema parece estar no facto de que o oráculo atual apenas verifica o último “qubit”, sem considerar o estado dos outros “qubits”. Para que o algoritmo de “Grover” funcione corretamente, o oráculo precisa marcar especificamente o estado alvo (ou estados alvos) que desejamos amplificar e, desta forma, não está a marcar nenhum estado específico. Para marcar mais do que um estado, seria necessário marcar exatamente um estado de cada vez, como no exemplo a seguir, dada a função oráculo definida na alínea anterior.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Figura 24 - Marcação de mais que 1 estado

Desta forma, caso fossem marcados todos os estados esperados seria possível obter os resultados inicialmente esperados.