Exercício 4: Algoritmos de Deutsch and Grover

Objetivos: Ao completar esta sequência de tarefas, o estudante será capaz de:

- Entender como algoritmos de clássicos podem ser convertidos em quânticos
- Entender o uso do algoritmo de Deutsch.
- Entender o uso do algoritmo de Grover

Avaliação

Tão importante quanto escrever um código que funcione corretamente é escrever um código legível, que seja fácil de entender e possa ser facilmente reutilizado por outros ou por você mesmo. Por isso a avaliação é separada da seguinte forma:

- 60% conteúdo, se os resultados estão corretos.
- 40% apresentação, código bem comentado, cédulas texto bem escritas.

Referências:

1. https://learn.qiskit.org/course/ch-states/representing-qubit-states
2. https://learn.qiskit.org/course/ch-states/single-qubit-gates
3. https://qiskit.org/documentation/apidoc/visualization.html
4. https://learn.qiskit.org/course/introduction/grovers-search-algorithm
5. https://learn.qiskit.org/course/ch-algorithms/deutsch-jozsa-algorithm
6. Michael A. Nielsen e Isaac L. Chuang, Computação quântica e informação quântica

Tarefa 0

Importe as bibliotecas necessárias.

In [2]:
```
import numpy as np
from qiskit import QuantumCircuit, Aer
from qiskit.visualization import plot_histogram
#Seu código aqui (se precisar use mais de uma célula)
Aer.backends()
backend = Aer.get_backend('aer_simulator')
```

Tarefa 1:

O circuito abaixo (conhecido como Full Adder) realiza a adição de dois bits em um computador clássico. Com base nele, construa um circuito quântico que realiza a mesma operação usando as portas CNOT e Toffoli. (Note que a porta Toffoli pode ser utilizada para simular diversas portas clássicas de forma reversível)

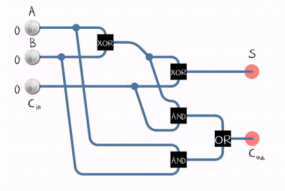

Figura de 'https://en.wikipedia.org/wiki/Adder_(electronics)'

In [ ]:
```
#Seu código aqui (se precisar use mais de uma célula)
```

Tarefa 2:

Considere uma função sobre os conjunto das sequências binárias de tamanho N em $0, 1$:

$$f(x) : \{0,1\}^N \to \{0,1\}$$

Mostre que está função pode ser calculada em um computador quântico através de um operador unitário U_f na forma do circuito abaixo::

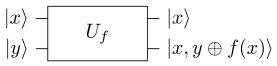

Figura de 'https://avalon-lang.readthedocs.io/en/latest/algorithms/deutsch.html'

onde x é uma sequência binária de tamanho N e y é um bit arbitrário.

Sua resposta aqui

Tarefa 3:

Na célula abaixo, a função oracle1 retorna um operador U_f que calcula uma função $f(x) : \{0,1\} \to \{0,1\}$ nos termos da Tarefa 2. Use o algoritmo de Deutsch para decidir se a função f é constante ou balanceada utilizando apenas uma chamada da função oracle1 em seu código. (Note que cada vez que rodar a célula abaixo, a função f muda)

In [ ]:
```
seed = np.random.randint(4)
#Está função recebe circuito de 2 q-bits, aplica o operador U_f, e retorna
def oracle1(qc: QuantumCircuit) -> QuantumCircuit:
    if seed == 0:
        qc.x(1)
    if seed == 1:
        qc.cnot(0,1)
    if seed == 2:
        qc.x(0)
        qc.cnot(0,1)
    return qc
```

In [ ]:
```
#Seu código aqui (se precisar use mais de uma célula)
```

Tarefa 4:

Seja f uma função teste sobre o conjunto das sequências de dois bits $\{00, 01, 10, 11\}$ no conjunto $\{0, 1\}$. Esta função retorna 1 (Verdadeiro) para apenas um item do domínio (o conjunto das sequências de dois bits). Como está função pode ser implementada em um computador quântico através de um operado U_f, pelo método da Tarefa 2, mostre que é possível determinar qual item retorna verdadeiro com apenas uma chamada do operador U_f com probabilidade de sucesso igual a 1.

Tarefa 5:

O operador U_f da tarefa 4 é implementado pela função oracle2 da célula abaixo. Construa um circuito quântico utilize esta função como o oracle no algoritmo de Grover para descobrir qual sequência de dois bits retorna o valor verdadeiro. Rode o circuito no simulador e obtenha esta sequência a partir dos resultados das medidas.

Sua resposta aqui

In [ ]:
```
seed = np.random.randint(4)
#Está função recebe circuito de 3 q-bits, aplica o operador U_f, e retorna
def oracle2(qc: QuantumCircuit) -> QuantumCircuit:
    qc.ccx(0,1,2,seed)
    return qc
```

In [ ]:
```
#Seu código aqui (se precisar use mais de uma célula)
```

Tarefa 6:

Agora considere uma função f sobre o conjunto das sequências de 8 bits em um bit, onde $f(x) = 1$ para a apenas uma sequência. Calcule quantas iterações do algoritmo de Grover são necessárias para se encontrar esta sequência de 8 bits. Use a função oracle3 abaixo como a função oráculo a ser utilizada no algoritmo de Grover. Construa o circuito quântico com o número de iterações do operador oráculo e operador de Grover necessárias. Obtenha então a sequências de bits com base na execução do algoritmo no computador quântico e resultados das medidas obtidas.

In [30]:
```
#Esta função recebe um c
n_bits = 8
seed = np.random.randint(2**n_bits)
binary = bin(seed)[2:].zfill(n_bits)

#Está função recebe circuito de 9 q-bits, aplica o operador oráculo U_f, 
def oracle3(qc: QuantumCircuit) -> QuantumCircuit:
    for i in range(n_bits):
        if binary[i] == '1':
            qc.x(i)
    qc.mcx(list(range(n_bits)), n_bits,mode='noancilla')
    for i in range(n_bits):
        if binary[i] == '1':
            qc.x(i)
    return qc
```

In [ ]:
```
#Seu código aqui (se precisar use mais de uma célula)
```

Desafio:

Rode o seu código da Tarefa 5 em um computador quântico real.