Deutsch-Jozsa Problem

- An extension of Deutsch algorithm to the case of an n qubit input, $f : \{0,1\}^{\otimes n} \longrightarrow \{0,1\}$
- The function is either a constant or balanced(i.e, exactly half of them gives 0 while the other half of them evaluates to 1).

1 Classical algorithm to solve the Deutsch-Jozsa Problem

1. Initialize a variable 'n' to the number of bits in the function's input
2. Initialize a variable 'x' to 0
3. Call the function with input 'x'. If the function returns 0, go to step-4. If the function returns 1, go to step-5
4. The function is constant. Return "constant" and stop
5. Initialize a variable 'y' to 1
6. Call the function with input 'y'. If the function returns 0, go to step-7. If the function returns 1, go to step-8
7. The function is balanced. Return "balanced" and stop
8. The function is constant. return "constant" and stop

This algorithm works by first querying the function with input 'x', which is always 0. If the function is constant, it will always return 0, and algorithm will correctly identify it as constant. If the function is balanced, it will return either 0 or 1 with equal probability, and the algorithm will proceed to step-5. In step-5, the algorithm queries the function with input 'y', which is always 1. If the function is constant, it will always return the same value as it did for input 'x'. If the function is balanced, it will return the opposite value to what it returned for input 'x'.

The Deutsch-Jozsa algorithm has a query complexity of O(1), meaning that it only requires a single query to the function to determine whether it is constant or balanced.

2 Quantum circuit to solve in a single evaluation

1. Initialize a quantum register with $'n+1'$ qubits, where 'n' is the number of bits in the function's input. The first 'n' qubits will be used to store the input to the function, and the last qubit will be used as an ancilla qubit
2. Initialize the input qubits to the value 0
3. Apply the Hadamard gate to all of the qubits. This will put the qubits into a superposition of all possible input states
4. Apply the function to the input qubits as a unitary operation. This will apply the function to all possible input states simultaneously
5. Apply the Hadamard gate to the ancilla qubit
6. Measure the ancilla qubit, if the measurement result is 0, the function is constant. If the measurement result is 1, the function is balanced

This quantum circuit works by using quantum superposition and interference to evaluate the function simultaneously on all possible input states. It has a query complexity of O(1), as it only requires a single evaluation to determine whether the function is constant or balanced. This is in contrast to classical algorithms, which typically have a query complexity of O(n), where 'n' is the number of bits in the function's input.

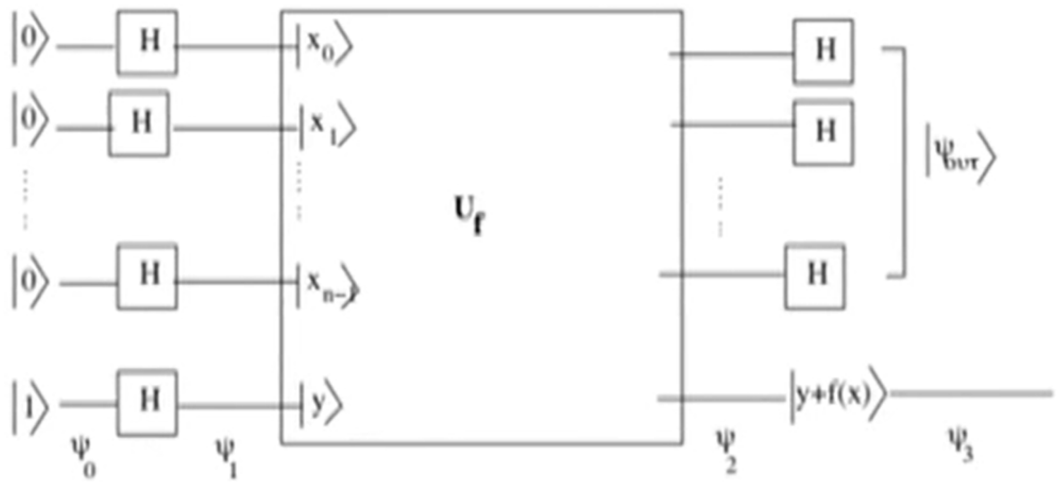

Fig: Quantum circuit for Deutsch-Jozsa Algorithm

- Input is a uniform linear combination of the n qubit computational basis states $\{x_{n-1}, x_{n-2}, ..., x_1, x_0\}$ with $x_i = 0, 1$
  Input: $|0\rangle^{\otimes n} \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2}} \sum_{x=0}^{2^n-1} |x\rangle$
- Target: $|1\rangle \xrightarrow{H} \frac{|0\rangle - |1\rangle}{\sqrt{2}}$
- Output(Entangled): $\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \left| f(x) - \overline{f(x)} \right\rangle$

- $\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} (-1)^{f(x)+k\cdot x} |k\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}$
- For balanced function, the coefficient of k=0 is $\sum_{x=0}^{2^n-1} (-1)^{f(x)} = 0$, because there are as many f(x)=1 as there are f(x)=0.
- First register=0 : Constant function
- First register anything else : balanced function

3 Classical randomized algorithm to solve Deutsch-Jozsa problem

1. Initialize a variable 'n' to the number of bits in the function's input
2. Generate a random binary string of length 'n' and call it 'x'
3. Call the function with input 'x'. If the function returns 0, go to step-4. If the function returns 1, go to step-5
4. The function is constant. Return "constant" and stop
5. Generate a random binary string of length 'n' and call it 'y'
6. Call the function with input 'y'. If the function returns 0, go to step-7. If the function returns 1, go to step-8
7. The function is balanced. Return "balanced" and stop
8. The function is constant. Return "constant" and stop

This algorithm works by generating random inputs 'x' and 'y' and calling the function with each of them. If the function is constant, it will always return the same value, and the algorithm will correctly identify it as constant. If the function is balanced, it will return either 0 or 1 with equal probability, and the algorithm will identify it as balanced.

The query complexity of this algorithm is O(1), as it only requires a single query to the function to determine whether it is constant or balanced. However, the success probability of the algorithm is only 50%, as it may generate the same input for both 'x' and 'y', in which case it will incorrectly identify the function as constant. To improve the success probability, you can repeat the algorithm multiple times and return the most common result. This will increase the query complexity to O(n), where 'n' is the number of repetitions.