

Exercice 03 : Algorithme de Deutsch-Jozsa

Cet exercice implémente l'algorithme de Deutsch-Jozsa (DJ), le premier algorithme quantique qui a démontré un avantage de vitesse théorique sur son équivalent classique. L'objectif est de déterminer si une fonction binaire $f : \{0, 1\}^n \rightarrow \{0, 1\}$ est **constante** (elle renvoie toujours 0 ou toujours 1) ou **balancée** (elle renvoie 0 pour la moitié des entrées et 1 pour l'autre moitié) en n'utilisant qu'une seule requête à l'Oracle quantique.

1. L'Avantage Quantique

Classiquement, pour déterminer avec certitude si une fonction f est constante ou balancée, il faut au moins $\frac{N}{2} + 1$ évaluations de la fonction (où $N = 2^n$ est le nombre total d'entrées).

- par exemple prenons la fonction f avec deux variable (x, y) tel que:

$$\begin{array}{ccc} f : & \{0, 1\}^2 & \rightarrow \{0, 1\} \\ & (x, y) & \rightarrow f(x, y) \end{array}$$

Il y a 4 entrées possible : $[0, 0], [0, 1], [1, 0], [1, 1]$.

On doit tester au moins 2 couples.

L'algorithme de Deutsch-Jozsa réalise cette tâche en **une seule évaluation** de la fonction f grâce à l'utilisation combinée de la **superposition** et de l'**interférence quantique**.

2. Structure de l'Algorithme

L'algorithme de Deutsch-Jozsa (DJ) se décompose en cinq étapes principales. Le circuit utilise n qubits de données $|x\rangle$ et un qubit auxiliaire $|y\rangle$.

A. Initialisation et Préparation

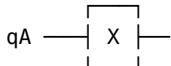
1. **Initialisation** : Le registre de données $|x\rangle$ (composé de n qubits) est initialisé à $|0\rangle^{\otimes n}$. Le qubit auxiliaire $|y\rangle$ est initialisé à $|1\rangle$.

Pour notre exemple :

```
q_0 : |0> -->
      |-> donnée d'entrée de la fonction
q_1 : |0> -->
```

```
q_A : |1> --> qubit auxiliaire
```

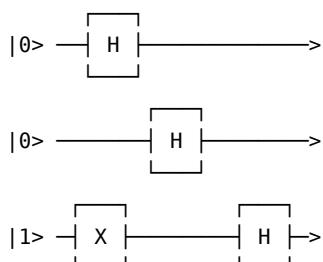
À noter que dans le simulateur ou l'ordinateur quantique on utilisera une porte X pour que le qubit auxiliaire puisse être $|1\rangle$, car il sont initialisé en $|0\rangle$



2. **Préparation de Phase** : La porte **H** (Hadamard) est appliquée au qubit auxiliaire $|y\rangle$. Ceci prépare le qubit auxiliaire à l'état $|-> = \mathbf{H}|1\rangle$, ce qui permet à l'Oracle de coder le résultat $f(x)$ sous forme de **phase**.

3. **Superposition** : La porte **H** est appliquée à tous les n qubits du registre $|x\rangle$. Ceci crée une superposition uniforme de tous les états d'entrée possibles : $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$.

pour notre exemple, on a donc:



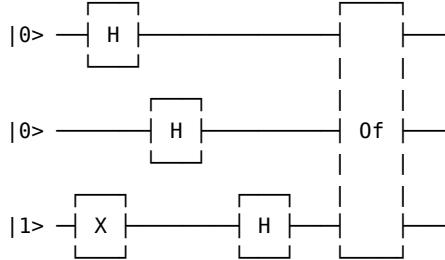
B. L'Oracle Quantique

4. **Application de l'Oracle (O_f)** : L'Oracle quantique O_f est appliqué. Il utilise la propriété de l'état $|-\rangle$ pour transformer l'action :

$$O_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle$$

L'information $f(x)$ est maintenant encodée dans la **phase** du registre de données $|x\rangle$.

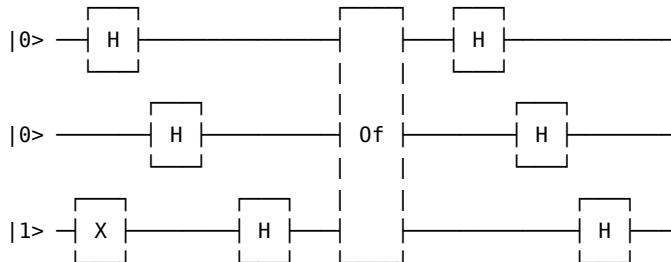
exemple :



C. Interférence et Résultat

5. **Interférence** : La porte **H** est appliquée à nouveau à tous les qubits du registre $|x\rangle$. Ce deuxième ensemble de portes Hadamard provoque une **interférence constructive** si l'Oracle est constant, et une **interférence destructive** si l'Oracle est balancé.

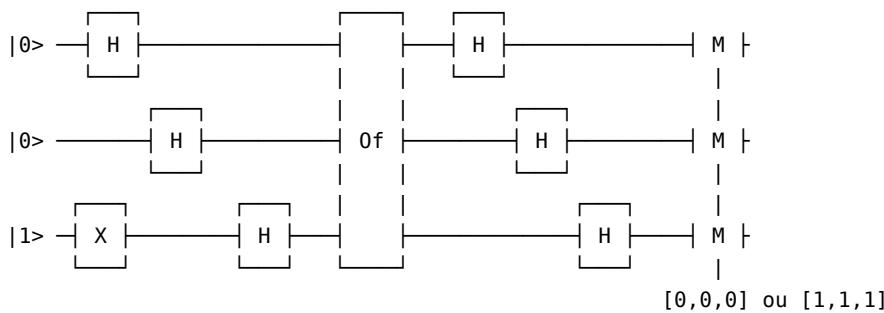
exemple :



6. **Mesure** : Le registre de données $|x\rangle$ est mesuré.

- **Si l'Oracle est Constant** : La mesure donne toujours l'état $|0\rangle^{\otimes n}$ (c'est-à-dire la chaîne '00...0').
- **Si l'Oracle est Balancé** : La mesure donne un état **non-nul** (une chaîne contenant au moins un '1').

exemple :



3. Structure du Code

Le programme est divisé en deux fichiers pour séparer la logique de l'algorithme et la définition de l'Oracle.

oracle.py (L'interface)

Ce fichier définit la classe **Base_Oracle** et les implémentations spécifiques des Oracles :

- **Base_Oracle** : Assure que tous les Oracles respectent l'interface requise : ils sont initialisés avec **n_qubits** et exposent la méthode **append_to_circuit(main_circuit)** pour insérer leur logique dans le circuit principal.

- **Classes spécifiques** : Contient des exemples d'Oracles Constant et Balancé pour les tests.

deutsch_jozsa.py (L'algorithme)

Ce fichier contient la fonction **run_deutsch_jozsa(oracle_obj)** qui :

1. Initialise le circuit quantique de taille ($n + 1$).
2. Applique **Transformation d'Hadamard **H** sur tous les *qubits*.
3. Appelle **oracle_obj.append_to_circuit(qc)** pour insérer le circuit de l'Oracle.
4. Applique les portes **H** finales.
5. Lance la mesure et conclut si l'Oracle est **CONSTANT** ou **BALANCÉ**.