Deutsch Algorithm

The Deutsch Algorithm, proposed by David Deutsch (1985), is the first quantum algorithm that demonstrates how a quantum computer can outperform a classical computer.

It is used to determine whether a function $f(x)$, with input $x \in \{0,1\}$, is:

- Constant: $f(0) = f(1)$
- Balanced: $f(0) \neq f(1)$

A classical computer requires two evaluations of $f(x)$, but a quantum computer can determine the result with one evaluation using superposition and interference.

2. Steps of the Algorithm

Step 1: Initialize qubits
Prepare two qubits:

$$|\psi_0\rangle = |0\rangle|1\rangle$$

The first qubit represents the input x, and the second assists in encoding the function.

Step 2: Apply Hadamard gates
Apply a Hadamard gate (H) to both qubits to create a superposition:

$$|\psi_1\rangle = \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$$

This represents both possible inputs (0 and 1) simultaneously.

Step 3: Apply oracle U_f
The oracle implements the function $f(x)$:

$$U_f|x, y\rangle = |x, y \oplus f(x)\rangle$$

This flips the phase of the first qubit depending on the function's output.

Step 4: Apply Hadamard to the first qubit again
A Hadamard gate is applied to the first qubit. Quantum interference ensures the amplitude of the first qubit reflects whether $f(x)$is constant or balanced.

Step 5: Measure the first qubit

- Result 0 → function is constant
- Result 1 → function is balanced

3. Minimal Math (Concept Only)

After the algorithm, the first qubit is in the state:

$$|\psi_{final}\rangle = \begin{cases} |0\rangle, & \text{if } f(0) = f(1) \text{ (constant)} \\ |1\rangle, & \text{if } f(0) \neq f(1) \text{ (balanced)} \end{cases}$$

Thus, only one measurement is needed to determine the answer.

4. Advantages

- Requires only one function evaluation, demonstrating quantum speed-up.
- Clearly illustrates quantum superposition and interference.
- Forms the foundation for Deutsch–Jozsa, Grover's, and Shor's algorithms.
- Demonstrates that quantum computing can outperform classical methods even for simple problems.

5. Disadvantages

- Applicable only to single-bit input functions.
- Mainly theoretical, with limited practical use.
- Requires perfect quantum gates and stable qubits to work reliably.
- Provides constant speed-up rather than exponential speed-up.

Deutsch–Jozsa Algorithm

The **Deutsch–Jozsa Algorithm** is a quantum algorithm designed to determine whether a given **n-bit function** $f(x)$is **constant** or **balanced**.

- **Constant function:** $f(x)$gives the same output (0 or 1) for all 2^npossible inputs.
- **Balanced function:** $f(x)$outputs 0 for exactly half of the inputs and 1 for the other half.

Classically, it may require up to $2^{n-1} + 1$**evaluations** to determine the type of function. Using quantum computing, the Deutsch–Jozsa Algorithm solves the problem with **only one evaluation** of the function.

2. Steps of the Algorithm

Step 1: Initialize qubits
Prepare $n + 1$qubits:

$$|\psi_0\rangle = |0\rangle^{\otimes n} |1\rangle$$

The first $n$qubits represent the input, and the last qubit is used to encode the function's output.

Step 2: Apply Hadamard gates
Apply a Hadamard gate to each qubit:

$$|\psi_1\rangle = H^{\otimes n+1} |\psi_0\rangle$$

The first $n$qubits now exist in a **superposition of all** 2^n**input states**.

Step 3: Apply oracle U_f
The oracle encodes the function into the quantum state:

$$U_f |x, y\rangle = |x, y \oplus f(x)\rangle$$

This applies a **phase shift** to the first $n$qubits depending on the function value.

Step 4: Apply Hadamard gates again to first n**qubits**
Applying Hadamard gates again causes **quantum interference**: the amplitude of $|0 \dots 0\rangle$depends on whether $f(x)$is constant or balanced.

Step 5: Measure the first n**qubits**

- If the measurement is **|0…0⟩ → function is constant**
- If the measurement is **any other state → function is balanced**

3. Minimal Math (Concept Only)

The amplitude of the $|0 \dots 0\rangle$state after the second Hadamard gates is:

$$\text{Amplitude} = \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)}$$

- **Constant function:** All terms are the same → amplitude ≠ 0 → measurement gives $|0 \dots 0\rangle$
- **Balanced function:** Half terms +1, half -1 → amplitude = 0 → measurement gives other states

Thus, **only one function evaluation** is sufficient to determine the type.

4. Advantages

- Provides **exponential speed-up** over classical algorithms for large n.
- Evaluates **all** 2^n**inputs simultaneously** using superposition.
- Demonstrates **quantum interference and parallelism** clearly.
- Forms a foundation for **many other quantum algorithms** in computation and cryptography.

5. Disadvantages

- Mainly **theoretical**, with limited practical applications.
- Requires **ideal quantum hardware** (perfect gates and no decoherence).
- Applicable only to **promise problems** where the function is guaranteed to be either constant or balanced.

Simon's Algorithm

Simon's Algorithm is a quantum algorithm developed by **Daniel Simon (1994)**.
It is designed to find a **hidden binary string** $s$in a function $f: \{0,1\}^n \to \{0,1\}^n$that satisfies the following property:

$$f(x) = f(y) \iff x \oplus y = s$$

Here, $s$is an unknown string and $\oplus$denotes bitwise XOR.

The problem is called **Simon's Problem**. Classically, finding $s$requires **exponentially many queries**, whereas the quantum algorithm solves it **efficiently in polynomial time**.

2. Steps of Simon's Algorithm

Step 1: Initialize qubits
Prepare **two registers** of $n$qubits each:

$$|\psi_0\rangle = |0\rangle^{\otimes n} |0\rangle^{\otimes n}$$

The first register holds input x, and the second holds the output of the function $f(x)$.

Step 2: Apply Hadamard gates to the first register
Apply Hadamard gates to the first register to create a superposition of all possible inputs:

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle$$

Step 3: Apply oracle U_f
The oracle encodes the function $f(x)$into the second register:

$$U_f |x\rangle |0\rangle = |x\rangle |f(x)\rangle$$

This produces a **superposition of all input-output pairs**.

Step 4: Measure the second register
Measuring the second register collapses it to a particular function output $f(x_0)$.

The first register then becomes a superposition of **all inputs that map to that output**, which are related by the hidden string s:

$$|x_0\rangle + |x_0 \oplus s\rangle$$

Step 5: Apply Hadamard gates to the first register again
Apply Hadamard gates to the first register. Quantum interference ensures that measurement of the first register gives a string $y$such that:

$$y \cdot s = 0 (\mathrm{mod} 2)$$

(where $\cdot$is the bitwise dot product).

Step 6: Repeat measurements
Repeat steps 1–5 multiple times (about $n$times) to collect enough linear equations to determine $s$uniquely.

Step 7: Solve for s
Use the linear equations over $\mathbb{F}_2$(binary field) obtained from the measurements to compute the hidden string s.

3. Minimal Math (Concept Only)

- Superposition: $|x\rangle$over all 2^ninputs.
- Oracle: $U_f |x\rangle |0\rangle = |x\rangle |f(x)\rangle$
- After Hadamard on first register: measurement yields $y$such that $y \cdot s = 0$
- Solve system of linear equations in binary to find s.

4. Advantages

- Exponentially faster than classical algorithms (polynomial vs. exponential).
- Demonstrates **quantum parallelism** and **interference** in practical hidden-structure problems.

- Forms the foundation for **Shor's Algorithm**, which uses similar quantum period-finding principles.

5. Disadvantages

- Requires **ideal quantum hardware** (no decoherence or errors).
- Only applicable to **promise problems** where the function satisfies the Simon property.
- Mainly **theoretical**, not directly practical for general computation problems.

Shor's Algorithm

Shor's Algorithm, developed by **Peter Shor (1994)**, is a quantum algorithm for **factoring large integers** efficiently.

- Given a composite number N, the goal is to find **nontrivial factors** of N.
- Classically, integer factorization requires **sub-exponential time**, but Shor's Algorithm solves it in **polynomial time** using quantum computation.
- It combines **quantum period finding** with **classical number theory**.

2. Steps of the Algorithm

Step 1: Precheck & choose a number a

- Select a random integer $a$such that $1 < a < N$.
- Compute gcd (a, N).
- If gcd $(a, N) > 1$, it is a factor of N, and we are done.

Step 2: Reduce factoring to period finding

- Define the function:

$$f(x) = a^x \text{mod } N$$

- Goal: find the **period** $r$such that $a^r \equiv 1(\text{mod} N)$.

Step 3: Quantum part — prepare superposition

- Use a quantum register to create a superposition of all integers $x$from 0 to $Q - 1$(where $Q$is a power of 2, $Q > N^2$).
- Apply the oracle $U_f$to compute $f(x)$in superposition.

Step 4: Apply Quantum Fourier Transform (QFT)

- QFT is applied to extract information about the period $r$from the superposed states.

Step 5: Measure and classical post-processing

- Measure the first register to obtain an integer m.
- Use the **continued fraction algorithm** to approximate $m/Q \approx s/r$, recovering the candidate period r.

Step 6: Verify the period and compute factors

- If $r$is even and $a^{r/2} \not\equiv -1(\text{mod} N)$, compute:

$$\text{gcd } (a^{r/2} - 1, N) \text{and} \text{gcd } (a^{r/2} + 1, N)$$

- These give nontrivial factors of N.
- If factors are trivial, repeat with a different a.

3. Minimal Math (Concept Only)

1. Function for period-finding: $f(x) = a^x \text{mod } N$
2. Period: $r$such that $a^r \equiv 1(\text{mod} N)$
3. Factors: gcd $(a^{r/2} \pm 1, N)$

Quantum steps (superposition + QFT) **find** r**efficiently**, which is the core of the speed-up.

4. Advantages

- Exponentially faster than classical factoring algorithms.
- Can break classical cryptosystems like **RSA** by factoring large numbers efficiently.
- Demonstrates **quantum parallelism** and **quantum Fourier transform** applications.
- Polynomial time in the number of bits of N.

5. Disadvantages

- Requires **large-scale, fault-tolerant quantum computers**; not yet practical for real-world RSA numbers.
- Sensitive to **quantum decoherence** and gate errors.
- The algorithm works efficiently only when the number is **composite**; trivial for primes.

Grover's Algorithm

Grover's Algorithm, developed by **Lov Grover (1996)**, is a quantum algorithm designed for **searching an unsorted database** efficiently.

- Given an **unsorted database of Nitems**, classical search requires $O(N)$time.
- Grover's Algorithm finds the desired item in **O(√N) queries**, demonstrating a **quadratic speed-up** over classical search.
- It is widely used in **quantum search and optimization problems**.

2. Steps of the Algorithm

Step 1: Initialize qubits

- Prepare $n$qubits ($N = 2^n$) in the **|0⟩** state:

$$\mid \psi_0 \rangle = \mid 0 \rangle^{\otimes n}$$

Step 2: Apply Hadamard gates

- Apply Hadamard gates to all qubits to create **equal superposition** of all states:

$$\mid \psi_1 \rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \quad \mid x \rangle$$

Step 3: Oracle marking

- Apply the **oracle function** $O_f$that flips the phase of the target state:

$$O_f \mid x \rangle = \begin{cases} -\mid x \rangle & \text{if } x \text{ is the target} \\ \mid x \rangle & \text{otherwise} \end{cases}$$

Step 4: Apply Grover diffusion operator

- Amplifies the probability of the target state by reflecting all amplitudes about their average.
- The combination of **oracle + diffusion** is called **Grover iteration**.

Step 5: Repeat Grover iterations

- Repeat the Grover iteration approximately $\frac{\pi}{4}\sqrt{N}$times to maximize the probability of the target state.

Step 6: Measurement

- Measure the qubits.
- The result will be the **target item** with high probability.

3. Minimal Math (Concept Only)

- Superposition: $\mid \psi \rangle = \frac{1}{\sqrt{N}} \sum \mid x \rangle$
- Oracle flips phase of target: $\mid x \rangle \to -\mid x \rangle$
- Diffusion amplifies target amplitude: reflection about mean

- Repeating $O(\surd N)$times gives high probability of correct output

4. Advantages

- Quadratic speed-up over classical search ($O(\surd N)$ vs $O(N)$).
- Works for **any unstructured search problem**.
- Demonstrates **quantum parallelism** and **amplitude amplification**.
- Useful in **optimization, database search, and cryptography**.

5. Disadvantages

- Only provides **probabilistic success**; repeated runs may be needed.
- Requires **ideal quantum hardware** to avoid decoherence and errors.
- Less dramatic improvement than Shor's Algorithm (quadratic vs exponential).
- Limited to problems where **oracle can be efficiently implemented**.

Phase Kickback

Phase Kickback is a quantum computing phenomenon where the **phase of a control qubit** is modified based on the state of a target qubit after a controlled operation.

- It is widely used in **quantum algorithms**, such as **Shor's Algorithm**, **Deutsch–Jozsa Algorithm**, and **Quantum Phase Estimation**.
- Phase kickback allows information about a function or operation to be **encoded in the phase** of a qubit rather than its amplitude.

2. Explanation

Consider a **controlled-unitary operation** CUapplied on two qubits:

$$\mid c\rangle \mid t\rangle \xrightarrow{CU} \mid c\rangle U^c \mid t\rangle$$

- Here, $\mid c\rangle$is the **control qubit** and $\mid t\rangle$is the **target qubit**.
- If the target qubit is in an eigenstate of $U$with eigenvalue $e^{i\phi}$, then applying CUeffectively **adds a phase $\phi$to the control qubit**:

$$\mid c\rangle \mid t\rangle \rightarrow e^{ic\phi} \mid c\rangle \mid t\rangle$$

This effect, where the **phase of the control qubit "kicks back"** from the target qubit, is called **phase kickback**.

- Phase kickback is crucial in **quantum phase estimation**, where eigenvalues of a unitary operator are encoded in the phase of control qubits.
- It allows quantum algorithms to **extract global information** efficiently without measuring the target qubits directly.

3. Minimal Math (Concept Only)

1. Controlled-unitary: $CU \mid c\rangle \mid u\rangle = \mid c\rangle U^c \mid u\rangle$
2. Target eigenstate: $U \mid u\rangle = e^{i\phi} \mid u\rangle$
3. Resulting control qubit: $\mid c\rangle \rightarrow e^{ic\phi} \mid c\rangle$

- The **phase of control qubit** now carries information about the eigenvalue ϕ.

4. Advantages

- Enables **efficient extraction of function or operator information** in quantum algorithms.
- Key component in **Shor's Algorithm** for factoring large numbers.
- Allows **phase encoding** without disturbing the target qubit.

- Facilitates **quantum parallelism** and **interference-based computations**.

5. Disadvantages

- Requires **target qubit to be in an eigenstate** of the unitary operator.
- Implementation depends on **accurate controlled-unitary operations**; hardware errors can affect phase.
- Concept is abstract and may be **difficult to visualize**.
- Sensitive to **decoherence** and noise in quantum systems.

Factoring Integers in Quantum Computing

In **quantum computing**, **factoring integers** refers to using a quantum algorithm to **efficiently find nontrivial factors** of a large composite number N.

- Classical algorithms take **exponentially long** for large numbers.
- Quantum algorithms, particularly **Shor's Algorithm**, can factor integers in **polynomial time** by exploiting **superposition, entanglement, and quantum interference**.

2. Importance in Quantum Computing

- **Cryptography:** RSA encryption relies on the difficulty of factoring large integers; quantum factoring can **break classical cryptosystems**.
- **Algorithmic demonstration:** Shows that quantum computers can **solve problems exponentially faster** than classical computers.
- **Quantum number theory applications:** Enables **efficient period finding**, modular arithmetic, and other number-theoretic computations.

3. Quantum Method: Shor's Algorithm

Step 1: Reduce factoring to period finding

- For a composite number $N$and randomly chosen $a < N$:

$$f(x) = a^x \bmod N$$

- Goal: find the **period** $r$such that $a^r \equiv 1(\mathrm{mod} N)$.

Step 2: Prepare superposition

- Use a quantum register to create a **superposition of all possible** x**values**.

Step 3: Apply quantum oracle U_f

- Compute $f(x)$in superposition:

$$|x\rangle|0\rangle \xrightarrow{U_f} |x\rangle|f(x)\rangle$$

Step 4: Apply Quantum Fourier Transform (QFT)

- Extract the **period** $r$from the amplitudes using interference patterns.

Step 5: Classical post-processing

- Measure the first register, use **continued fractions** to find r, then compute factors:

$$\gcd(a^{r/2} \pm 1, N)$$

4. Advantages in Quantum Computing

- Provides **exponential speed-up** over classical factoring methods.
- Exploits **quantum parallelism** to evaluate all possible inputs simultaneously.
- Can potentially **break RSA and other classical cryptosystems**.
- Demonstrates key quantum concepts: **superposition, entanglement, phase kickback, and QFT**.

5. Disadvantages / Challenges

- Requires **fault-tolerant, large-scale quantum computers**.
- Sensitive to **decoherence and gate errors**, which can affect results.
- Only practical for **composite numbers**; quantum factoring is currently limited by hardware size.

Probabilistic Versus Quantum Algorithms

Probabilistic Algorithms:

- Algorithms that use **randomness** as part of their logic to solve a problem.
- Output may **vary for the same input** and usually provides a **correct answer with high probability**.
- Examples: **Randomized quicksort, Monte Carlo algorithms, Pollard's rho algorithm**.

Quantum Algorithms:

- Algorithms that leverage **quantum mechanics principles** like **superposition, entanglement, and interference**.
- Solve problems by **exploring many possibilities simultaneously** and manipulating probability amplitudes.
- Examples: **Shor's Algorithm, Grover's Algorithm, Deutsch–Jozsa Algorithm**.

Feature	Probabilistic Algorithms	Quantum Algorithms
Basis	Classical randomness	Quantum mechanics
Computation	Processes one possibility at a time	Processes **superpositions** of all possibilities simultaneously
Output	Correct with high probability	Correct with high probability, can leverage interference for exact or amplified probabilities
Speed	Often faster than deterministic classical algorithms, but still limited	Can achieve **exponential** or **quadratic speed-up** over classical algorithms
Examples	Monte Carlo, Las Vegas algorithms	Shor's, Grover's, Deutsch–Jozsa

3. Minimal Math / Concept

- **Probabilistic:** Uses random variable X and probability $P(X = x)$ to guide computation.
- **Quantum:** Uses **quantum states** $|\psi\rangle = \sum \alpha_i \mid i\rangle$ and **probability amplitudes** $|\alpha_i|^2$ for measurement outcomes.

5. Advantages of Quantum Algorithms

- Exploit **superposition and entanglement** to explore multiple solutions simultaneously.
- Can provide **exponential speed-up** (Shor) or **quadratic speed-up** (Grover) over classical methods.
- Solve certain problems **impossible for classical computers in reasonable time**.