**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 , with input , is:

* Constant:
* Balanced:

A classical computer requires two evaluations of , 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:

The first qubit represents the input , 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:

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

Step 3: Apply oracle   
The oracle implements the function :

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

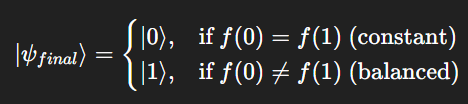
Step 4: Apply Hadamard to the first qubit again  
A Hadamard gate is applied to the first qubit.  
Quantum interference ensures the amplitude of the first qubit reflects whether is constant or balanced.

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:



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** is **constant** or **balanced**.

* **Constant function:** gives the same output (0 or 1) for all possible inputs.
* **Balanced function:** outputs 0 for exactly half of the inputs and 1 for the other half.

Classically, it may require up to **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 qubits:

The first 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:

The first qubits now exist in a **superposition of all input states**.

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

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

**Step 4: Apply Hadamard gates again to first qubits**  
Applying Hadamard gates again causes **quantum interference**: the amplitude of depends on whether is constant or balanced.

**Step 5: Measure the first 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 state after the second Hadamard gates is:

* **Constant function:** All terms are the same → amplitude ≠ 0 → measurement gives
* **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 .
* Evaluates **all 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** in a function that satisfies the following property:

Here, is an unknown string and denotes bitwise XOR.

The problem is called **Simon’s Problem**. Classically, finding 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 qubits each:

The first register holds input , and the second holds the output of the function .

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

**Step 3: Apply oracle**   
The oracle encodes the function into the second register:

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

**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 such that:

(where is the bitwise dot product).

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

**Step 7: Solve for**   
Use the linear equations over (binary field) obtained from the measurements to compute the hidden string .

**3. Minimal Math (Concept Only)**

* Superposition: over all inputs.
* Oracle:
* After Hadamard on first register: measurement yields such that
* Solve system of linear equations in binary to find .

**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 , the goal is to find **nontrivial factors** of .
* 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**

* Select a random integer such that .
* Compute .
* If , it is a factor of , and we are done.

**Step 2: Reduce factoring to period finding**

* Define the function:
* Goal: find the **period** such that .

**Step 3: Quantum part — prepare superposition**

* Use a quantum register to create a superposition of all integers from 0 to (where is a power of 2, ).
* Apply the oracle to compute in superposition.

**Step 4: Apply Quantum Fourier Transform (QFT)**

* QFT is applied to extract information about the period from the superposed states.

**Step 5: Measure and classical post-processing**

* Measure the first register to obtain an integer .
* Use the **continued fraction algorithm** to approximate , recovering the candidate period .

**Step 6: Verify the period and compute factors**

* If is even and , compute:
* These give nontrivial factors of .
* If factors are trivial, repeat with a different .

**3. Minimal Math (Concept Only)**

1. Function for period-finding:
2. Period: such that
3. Factors:

Quantum steps (superposition + QFT) **find 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 .

**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 items**, classical search requires 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 qubits () in the **|0⟩** state:

**Step 2: Apply Hadamard gates**

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

**Step 3: Oracle marking**

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

**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 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:
* Oracle flips phase of target:
* Diffusion amplifies target amplitude: reflection about mean
* Repeating times gives high probability of correct output

**4. Advantages**

* Quadratic speed-up over classical search ( vs ).
* 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** applied on two qubits:

* Here, is the **control qubit** and is the **target qubit**.
* If the target qubit is in an eigenstate of with eigenvalue , then applying effectively **adds a phase to the control qubit**:

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:
2. Target eigenstate:
3. Resulting control qubit:

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

**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 .

* 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 and randomly chosen :
* Goal: find the **period** such that .

**Step 2: Prepare superposition**

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

**Step 3: Apply quantum oracle**

* Compute in superposition:

**Step 4: Apply Quantum Fourier Transform (QFT)**

* Extract the **period** from the amplitudes using interference patterns.

**Step 5: Classical post-processing**

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

**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 and probability to guide computation.
* **Quantum:** Uses **quantum states** and **probability amplitudes** 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**.