Complessità Computazionale e Oracoli

1. Introduzione alla Complessità Computazionale

La complessità computazionale è una branca dell'informatica teorica che studia l'efficienza degli algoritmi e la difficoltà intrinseca dei problemi. Si occupa di classificare i problemi in base alla quantità di risorse computazionali richieste per risolverli, come il tempo (numero di passi) o lo spazio (quantità di memoria). Le principali classi di complessità che vengono introdotte sono P e NP.

Definizione di O grande

Sia f : N → N una funzione che restituisce il numero di passi di calcolo elementari per un algoritmo A, dato un input di dimensione n. Scriviamo f ∈ O(g(n)) se f cresce asintoticamente tanto velocemente o lentamente di g.

Common asymptotic functions

Function	designation	example algorithm
1	constant	calculate mod 2
$log(n)$	logarithmic	binary search (in sorted database)
n	linear	search in unsorted data
$n\,log(n)$	superlinear	merge sort
n^2	quadratic	multiplication of integers
n^3	cubic	matrix multiplication
n^k	polynomial ($k \in$ N fixed)	
2^n	exponential	naive calculation of the n-th Fibonacci number

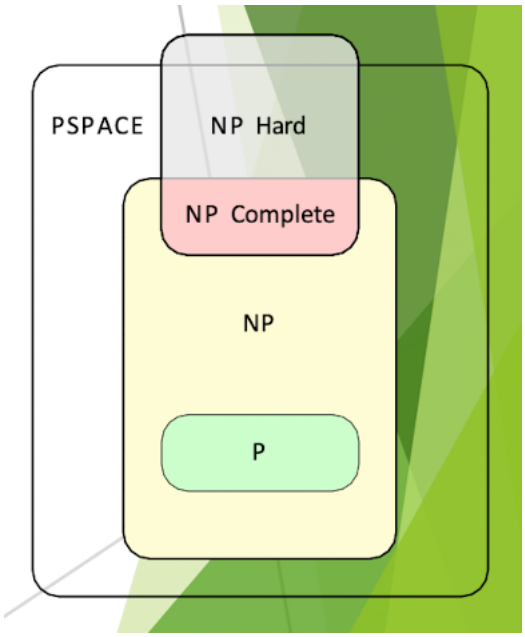
2. Classi di Complessità: P e NP

- **P**: insieme dei problemi decisionali (cioè che richiedono una risposta sì/no) risolvibili da un algoritmo in **tempo polinomiale** rispetto alla dimensione dell'input.
- **NP**: insieme dei problemi decisionali per cui, se qualcuno propone una soluzione, possiamo **verificarla** in tempo polinomiale.
- **NP-complete:**

1. È in **NP** (cioè possiamo verificare una soluzione in tempo polinomiale).
2. È **il più difficile tra i problemi in NP**, nel senso che **ogni altro problema in NP può essere ridotto a questo** in tempo polinomiale.

In altre parole, se trovassimo un algoritmo efficiente (polinomiale) per **anche uno solo** dei problemi NP-completi, **potremmo risolvere efficientemente tutti i problemi in NP**.

- **NP-hard**: È **almeno difficile quanto** i problemi in NP, ma **non è necessariamente in NP** (quindi magari **non possiamo neppure verificare** le soluzioni in tempo polinomiale).
- **PSPACE**: contiene tutti i problemi risolvibili usando **uno spazio di memoria polinomiale**, indipendentemente dal tempo impiegato.



Un esempio classico in NP è il problema del **commesso viaggiatore**: trovare il percorso minimo tra varie città. Verificare la validità e il costo di un percorso è facile, ma trovarlo potrebbe essere molto difficile.

La grande domanda aperta è: **P = NP?** Ovvero, ogni problema per cui possiamo verificare velocemente una soluzione, possiamo anche risolverlo velocemente?

BPP – Bounded-error Probabilistic Polynomial time

La classe **BPP (Bounded-error Probabilistic Polynomial time)** comprende tutti quei problemi decisionali che possono essere risolti da un **algoritmo randomizzato** in **tempo polinomiale**, con una **probabilità di errore limitata**.

Caratteristiche Principali

1. ✅ **Errore limitato**

   L'algoritmo può restituire un risultato sbagliato, ma con **probabilità inferiore a ½**.

2. 🎲 **Algoritmo randomizzato**

   Gli errori derivano **dall'uso di scelte casuali**: l'algoritmo prende decisioni basate sulla casualità, per ottenere il risultato.

3. 🔁 **Riduzione dell'errore via ripetizione**

La probabilità di errore può essere **resa arbitrariamente piccola** eseguendo **più volte l'algoritmo** e scegliendo il risultato più frequente (strategia del **voto di maggioranza**).

4. *BPP rappresenta il confine della fattibilità per i computer classici.*

- **P ⊆ BPP** (tutti gli algoritmi deterministici sono anche algoritmi probabilistici che non usano casualità).
- Si ritiene probabile che **BPP = P**, ma non è ancora dimostrato formalmente.
- **BPP ⊆ PSPACE**, cioè tutto ciò che si può fare in BPP si può fare anche con memoria polinomiale.

BQP – Bounded-error Quantum Polynomial time

BQP è la classe dei problemi risolvibili **in tempo polinomiale da un computer quantistico**, con **errore limitato** (come BPP).

Un problema decisionale E appartiene alla classe **BQP** (*Bounded-error Quantum Polynomial time*) se esiste un algoritmo quantistico A tale che:

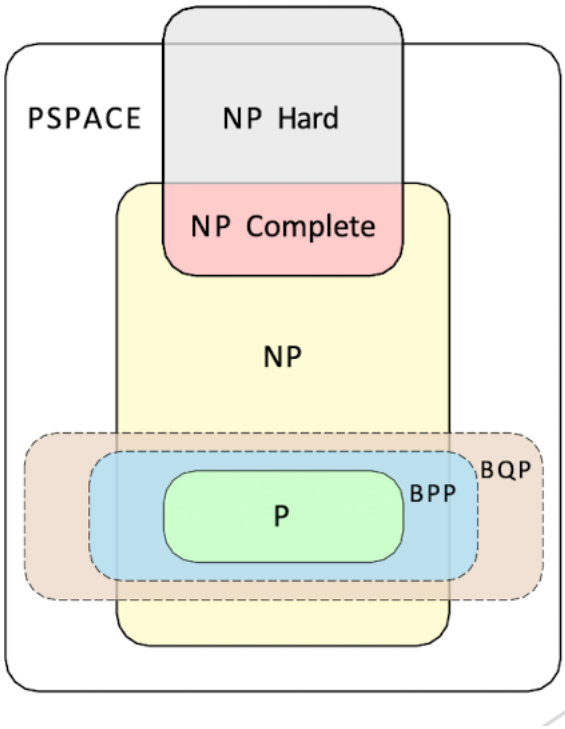
1. **Se la risposta corretta è E(x) = 1 per un input x, allora l'algoritmo restituisce A(x) = 1 con probabilità maggiore di 1/2.**
2. **Se la risposta corretta è E(x) = 0 per un input x, allora l'algoritmo restituisce A(x) = 0 con probabilità maggiore di 1/2.**
3. **L'algoritmo può essere implementato con circuiti quantistici uniformi di dimensione polinomiale.**

-La tripletta **{NOT, AND, OR}** è effettivamente sufficiente per costruire qualunque circuito logico classico; quindi, è un **insieme universalmente completo** per la computazione classica.

-La **porta di Toffoli** (nota anche come **CCNOT**) è una porta logica **reversibile** e può essere effettivamente **implementata nei computer quantistici**. È spesso usata per simulare circuiti classici in ambito quantistico.

-*Ogni circuito classico può essere trasformato in un circuito quantistico (reversibile) con un overhead al più polinomiale.*

-Tutti i problemi che una macchina classica può risolvere in tempo polinomiale (**P**) possono essere risolti anche da un computer quantistico nello stesso tempo massimo (**BQP**). (*La classe P è contenuta nella classe BQP.*)



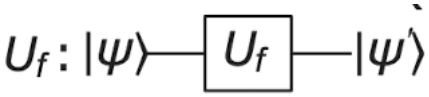
3. Macchine di Turing e Oracoli

Una **macchina di Turing** è un modello matematico che rappresenta il funzionamento astratto di un calcolatore. Una **macchina di Turing con oracolo** è una versione potenziata, in grado di fare domande a una "scatola nera" (l'oracolo), che risponde istantaneamente su un particolare problema.

Questa macchina permette di esplorare cosa accadrebbe **se avessimo risposte istantanee a problemi complessi**.

ORACOLI

1. *Un oracolo nella teoria della complessità risolve un problema in un singolo passo, cioè in tempo O(1).*
2. Un oracolo è trattato come una **scatola nera**: **non sappiamo come funziona**, ma possiamo **interrogarlo** fornendo input e osservando l'output. Questo è detto **modello a scatola nera** (*black box model*).
3. nella **computazione quantistica**, un oracolo è modellato come un **operatore unitario** Uf che agisce su uno stato quantistico psi.



Questo significa che l'oracolo **codifica la funzione f** in un circuito quantistico reversibile.

Quindi: *un oracolo quantistico è rappresentato da un operatore unitario* Uf *che agisce su registri quantistici e codifica la funzione* f *in modo reversibile.*

4. *La computazione quantistica permette di ridurre il numero di chiamate a un oracolo rispetto alla computazione classica.*

Algoritmo di Deutsch

Supponiamo di avere un **oracolo quantistico** che implementa una funzione:

f: {0, 1} →{0, 1}. Questa funzione prende un bit in input e restituisce un bit in output. Tuttavia, non conosciamo il comportamento interno della funzione: vogliamo determinare se è **costante** oppure **bilanciata**.

🔷 **1. Descrizione del problema**

Sappiamo che la funzione f può essere di due tipi:

- **Costante**, se restituisce sempre lo stesso valore per entrambi gli input: ad esempio, f(0) = f(1) = 0, oppure f(0) = f(1) = 1.
- **Bilanciata**, se restituisce un valore diverso per ciascun input: ad esempio, f(0) = 0 e f(1) = 1, oppure viceversa.

Il nostro obiettivo è determinare se la funzione è costante o bilanciata, effettuando **il minor numero possibile di chiamate all'oracolo**.

🔷 **2. Metodo classico**

Su un computer classico, l'unico modo per risolvere il problema è interrogare direttamente la funzione due volte:

- Si calcola f(0) e poi f(1),
- Si confrontano i due risultati.

Se i due output sono uguali, la funzione è costante. Se sono diversi, la funzione è bilanciata.

➡️ In sintesi: **sono necessarie due chiamate all'oracolo**.

🔷 **3. Metodo quantistico**

Con la **computazione quantistica**, è possibile risolvere il problema con una **sola** chiamata all'oracolo, grazie a tre strumenti fondamentali:

- **Sovrapposizione**, che permette di valutare la funzione su più input contemporaneamente,
- **Interferenza quantistica**, che consente di cancellare o amplificare stati,
- **Gate di Hadamard**, che trasformano stati base in combinazioni lineari (superposizioni).

Il procedimento è il seguente:

1. Si preparano due qubit: il primo nello stato |0>, il secondo nello stato |1>.

2. Si applica il **gate di Hadamard** a entrambi i qubit, creando una sovrapposizione.
3. Si esegue una sola chiamata all'oracolo quantistico Uf, che agisce su tutti gli input in parallelo.
4. Si applica di nuovo il **gate di Hadamard** al primo qubit (quello che conteneva l'input).
5. Infine, si misura il primo qubit:
    - Se si ottiene **0**, la funzione è costante.
    - Se si ottiene **1**, la funzione è bilanciata.

🔷 4. Vantaggio quantistico

Questo schema dimostra un concetto fondamentale della computazione quantistica: **un problema che richiede due valutazioni classiche può essere risolto con una sola valutazione quantistica**, sfruttando la **sovrapposizione e interferenza**.

➡️ Questo è il primo esempio storico di **vantaggio quantistico**, anche se in un caso molto semplice.

1. *L'oracolo quantistico* Uf *viene applicato una sola volta per determinare se* f *è costante o bilanciata.*

   *(*Questo è esattamente ciò che rende l'algoritmo di Deutsch un esempio di vantaggio quantistico: una sola chiamata all'oracolo è sufficiente per risolvere un problema che, in modo classico, richiede più interrogazioni.*)*

2. *Può essere esteso a funzioni booleane con dimensione di input arbitraria.*

   *(*L'estensione naturale dell'algoritmo di Deutsch è l'**algoritmo di Deutsch-Jozsa**, che lavora su funzioni f: {0,1}^n →{0,1}, cioè con input di lunghezza arbitraria n. Anche lì, una sola chiamata all'oracolo quantistico (con sovrapposizione di tutti gli input possibili) è sufficiente.*)*

3. *Vantaggio teorico rispetto all'equivalente classico, ma scarso valore pratico.*

   *(*L'algoritmo dimostra un chiaro **speedup teorico**, ma per input piccoli il guadagno è minimo, e il problema non è particolarmente utile in applicazioni reali. Tuttavia, è **fondamentale come base teorica** per comprendere algoritmi più avanzati.*)*

4. *Tuttavia, la maggior parte degli algoritmi quantistici con vantaggi dimostrabili si basa su oracoli quantistici combinati con interferenza.*