NP Completezza

venerdì 27 gennaio 2023 15:49

NP-COMPLETEZZA

CLASSI P E NP

Con l'eccezione della sezione su backtrack, finora abbiamo considerato solo problemi con soluzioni in tempo polinomoniale

- Il tempo di esecuzione è $O(n^k)$ per qualche k

È naturale classificare i problemi che si possono presentare secondo alcune caratteristiche:
- Dato un problema, esiste un algoritmo che lo risolve?
- Dato un problema tale che esista almeno un algoritmo che lo risolve, che complessità ha un tale algoritmo? E qual è l'algoritmo migliore possibile per quel problema?

Esistono problemi dove non esiste una soluzione: prendere in input un programma e decidere se il programma terminerà oppure no!

HALTING PROBLEM;

Problema dell'arresto
Dato un programma (espresso in un qualche linguaggio) ed i suoi dati di ingresso, la esecuzione del programma terminerà in un tempo finito?

Teorema dell'arresto (Turing)
Non può esistere nessun algoritmo che risolva il problema dell'arresto

Quindi, non tutti i problemi ammettono un algoritmo che li risolva.

IN QUALI CLASSI DI COMPLESSITÀ POSSIAMO ANDARE A PARTIZIONARE QUESTI ALGORITMI?

LA CLASSE DI COMPLESSITÀ P LA DEFINIAMO COME L'INSIEME DI TUTTI I PROBLEMI, TALI CHE, L'ALGORITMO CHE LI RISOLVE È ORDINE DI UN POLINOMIO

Complessità polinomiale (nel tempo): $\mathbb{P}$
La classe cui appartiene qualunque algoritmo che sia $O(p(n))$ dove p è un polinomio in n.

Il problema dell'ordinamento appartiene alla classe P, perché ci sono algoritmi che risolvono il problema in ordine polinomiale.

La ricerca del nodo A al nodo B del percorso minimo su un grafo è polinomiale!

Per molti problemi non si conosce alcun algoritmo polinomiale

Ma nessuno ha dimostrato che un tale algoritmo non possa esistere.

Problemi in $\mathbb{P}$
- Connettività: Stabilire se un grafo G è connesso;
- Cammini in un grafo: Dato un grafo orientato $G = (V, E)$ e due sottoinsiemi dei vertici $S, T \subseteq V$, esiste un cammino da un vertice di S ad un vertice di T?
- Matching: Dato un grafo G ed un intero k, esiste in G un matching di dimensione $\geq k$?
- Spanning tree: Dato un grafo $G = (V, E)$, una funzione di costo $d(E)$ ed un numero L, esiste un albero che ricopra G con un costo $\leq L$?
- Programmazione lineare Trovare la soluzione di

$$\begin{aligned} \min \quad & cx \\ Ax &= b \\ x &\geq 0 \end{aligned}$$

Cosa succede se non si trova un algoritmo polinomiale?

La classe $\mathbb{NP}$
È l'insieme dei problemi che sono risolubili in tempo *polinomiale* da una Macchina di Turing *nondeterministica*

LA MACCHINA SVOLGE TUTTE LE COMPUTAZIONI IN PARALLELO.

$$P \subseteq NP$$

- Maximum Clique: Dato un grafo $G = (V, E)$ ed un intero k, stabilire se il grafo contiene una *cricca* di dimensione k;
- Commesso viaggiatore: Dato un grafo (pesato) $G = (V, E)$, trovare, se esiste, un percorso che tocchi *tutti* i vertici, al costo più basso possibile;
- Soddisfacibilità: Data una espressione booleana, trovare se esiste una assegnazione delle sue variabili $x_1, \ldots, x_n$ tale che il valore della espressione sia VERO;
- Programmazione lineare a valori interi Trovare la soluzione di

$$\begin{aligned} \min \quad & cx \\ Ax &= b \\ x &\geq 0 \end{aligned}$$

dove A, x e b sono interi.

Per questi problemi qui nessuno conosce un algoritmo che non sia provarle tutte!
Ed è un metodo bruto "tentarle tutte".

RIDUCIBILITÀ

DATI DUE PROBLEMI DICIAMO CHE IL PROBLEMA 1 SI RIDUCE AL PROBLEMA 2, SE C'È UN ALGORITMO PER IL PROBLEMA 1 CHE AMMETTE COME SINGOLA ISTRUZIONE LA CHIAMATA ALL'ALGORITMO PER IL PROBLEMA 2.

Definizione

Riducibilità

Dati due problemi P_1 e P_2, diciamo (informalmente) che P_1 si riduce in tempo polinomiale a P_2 se esiste un algoritmo polinomiale A_1 che usa un altro algoritmo A_2 per il problema P_2 nel corso della soluzione di P_1.

IN QUESTO CASO IL PROBLEMA 1 È RIDUCIBILE IN TEMPO POLINOMIALE AL PROBLEMA 2.

Se un problema è riducibile ad un altro in tempo polinomiale, allora l'esistenza di un algoritmo polinomiale per il secondo problema implica l'esistenza di un algoritmo polinomiale per il primo

$\downarrow$

Definizione

$\mathbb{NP}$-completezza

Un problema (decisionale) si dice $\mathbb{NP}$-completo se

1. *Appartiene a $\mathbb{NP}$;*
2. *Tutti i problemi in $\mathbb{NP}$ sono riducibili ad esso in tempo polinomiale.*

↳ I problemi che stanno dentro NP, o sono tutti polinomiali o non lo è nessuno.

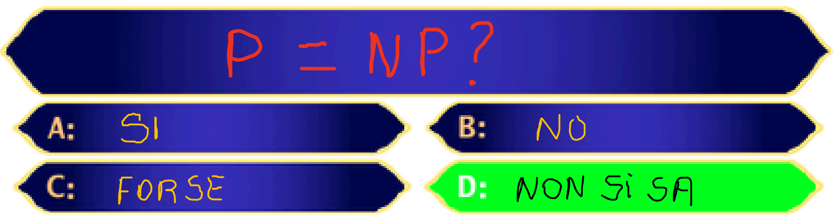
TEOREMA DI COOK

IL PROBLEMA DELLA SODDISFACIBILITÀ È NP-COMPLETO!

Altri problemi $\mathbb{NP}$-completi:

- Commesso viaggiatore (Traveling Salesman Problem);
- Programmazione Lineare a valori Interi;
- Partizione di un grafo
- Scheduling multiprocessore

P = NP?



Una definizione formale fa uso delle classi di complessità **P** e **NP**. La prima consiste di tutti quei problemi di decisione che possono essere risolti con una macchina di Turing deterministica in un tempo che è polinomiale rispetto alla dimensione dei dati di ingresso; la seconda consiste di tutti quei problemi di decisione le cui soluzioni positive possono essere verificate in tempo polinomiale avendo le giuste informazioni, o, equivalentemente, la cui soluzione può essere trovata in tempo polinomiale con una macchina di Turing non deterministica. Il problema delle classi P e NP si risolve quindi nella seguente domanda:

P è uguale a **NP**?

Un esempio per avere un'idea di cosa ciò vuole dire. Supponiamo di voler calcolare tutti i divisori (divisione intera, ovvero con resto pari a zero) di un numero **n**. Il problema, quindi, è *trovare tutti i numeri x tali che x è un divisore di **n***.

È abbastanza facile verificare che un certo numero x_0 è divisore di **n**; è sufficiente svolgere l'operazione di divisione e controllare il resto: se è pari a zero, il numero è un divisore, altrimenti non lo è. Il numero di passaggi richiesti per eseguire l'operazione di divisione è tanto maggiore quanto maggiore è il numero **n**, tuttavia essa risulta sempre abbastanza veloce perché il tempo da essa richiesto sia considerato *accettabile*.

Al contrario, potrebbe non essere altrettanto facile determinare l'insieme di *tutti* i divisori. Infatti, quasi tutti i metodi[1] finora ideati nel corso dei secoli richiedono un tempo che aumenta rapidamente al crescere del valore di **n**, troppo perché esso sia considerato *accettabile*.

Cosa si fa quando vi capita un problema che sembra esponenziale?
Si cerca di dimostrare se sia $\mathbb{NP}$-completo

E poi ? La $\mathbb{NP}$-completezza entra in gioco quando si voglia trovare *la soluzione esatta (ottima)* del problema.
Bisogna quindi "accontentarsi" (rilassare la richiesta):

- Si può risolvere un caso particolare del problema;
- Si può accettare un metodo che trovi la soluzione ottima con una probabilità del YY% (ma occasionalmente fallisca);
- Si può accettare un metodo che trovi una soluzione che sia ragionevolmente vicina ma non eguale a quella ottima;
- Si può accettare un metodo che con una probabilità di ZZ% trovi una soluzione *che sia*

ragionevolmente vicina a quella ottima.

Spesso ci affidiamo a delle euristiche, ossia delle tecniche di calcolo che funzionano (sufficientemente bene) in pratica ma per le quali non ci sono dimostrazioni rigorose di correttezza ed efficienza.