Contents

computational complexity

def: problema in computer science

un problema π é una relazione

$$\pi \subseteq I_\pi \times S_\pi$$

dove:

- $I_\pi =$ insieme delle istanze di input del problema
- $S_\pi =$ insieme delle soluzioni del problema

tipologie di problema

- decisione
  - si verifica se una data proprietá é valida per un determinato input
  - $S_\pi = \{true, false\}$ o semplicemente $S_\pi = \{0,1\}$ e la relazione $\pi \subseteq I_\pi \times S_\pi$ corrisponde ad una funzione

$$f : I_\pi \to \{0,1\}$$

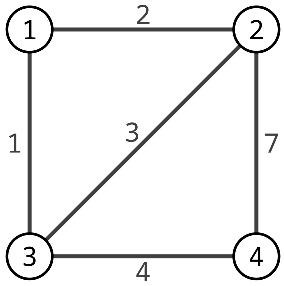
  - esempi: soddisfacibilitá, test di connettivitá di un grafo, etc....
- ricerca
  - data un'istanza $x \in I_\pi$, si chiede di determinare una soluzione $y \in S_\pi$ tale che la coppia $(x,y) \in \pi$ appartengono alla relazione che definisce il problema
  - esempi: soddisfacibilitá, clique, vertex cover, nei quali chiediamo in output un assegnamento di veritá soddisfacente, rispettivamente una clique o un vertex cover, invece di semplicemente "si" o "no"
- ottimizzazione
  - data un'istanza $x \in I_\pi$, si chiede di determinare una soluzione $y \in S_\pi$ ottimizzando una data misura della funzione costo
  - esempi: min spanning tree, max SAT, max clique, min vertex cover, min TSP, etc....

complessitá degli algoritmi e dei problemi

- espressa in funzione della taglia dell'input (denotata come $|x|, \forall x \in I_\pi$)
- taglia dell'istanza x
  - quantitá di memoria necessaria a memorizzare x in un computer
  - lunghezza $|x|_c$ della stringa che codifica x in un particolare codice naturale $c : I_\pi \to \Sigma$, dove Σ é l'alfabeto del codice c
- codice naturale
  - conciso: le stringhe che codificano le istanze non devono essere ridondanti o allungate inutilmente
  - numeri espressi in base ≥ 2

esempio: codice

- istanza: grafo G



- codice per G
  - $\Sigma = \{\{,\},,,0,1,2,3,4,5,6,7,8,9\}$ (simboli)
  - $c(G) = \{1,2,3,4,\{1,2\},\{1,3\},\{2,3\},\{2,4\},\{3,4\},2,1,3,7,4\}$
    * $\{1,2,3,4\}$ (nodi)
    * $\{\{1,2\},\{1,3\},\{2,3\},\{2,4\},\{3,4\}\}$ (archi)
    * $\{2,1,3,7,4\}$ (pesi)
  - $|G|_c = 49$

def: tempo di esecuzione dell'algoritmo A

sia $t_A(x)$ il tempo di esecuzione dell'algoritmo A per l'input x_i, allora il tempo di esecuzione nel caso peggiore di A é:

$$T_A(n) = \max\{t_A(x) \mid |x| \leq n\}, \qquad \forall n > 0$$

def: complessitá temporale dell'algoritmo A

l'algoritmo A ha complessitá temporale

- $O(g(n))$ se $T_A(n) = O(g(n))$, ovvero

$$\lim_{n\to\infty} \frac{T_A(n)}{g(n)} \leq c, \text{ per una costante } c > 0$$

- $\Omega(g(n))$ se $T_A(n) = \Omega(g(n))$, ovvero

$$\lim_{n\to\infty} \frac{T_A(n)}{g(n)} \geq c, \text{ per una costante } c > 0$$

- $\Theta(g(n))$ se $T_A(n) = \Theta(g(n))$, ovvero

$$T_A(n) = \Omega(g(n)) \text{ e } T_A(n) = O(g(n))$$

def: complessitá di un problema

un problema ha complessitá

- $O(g(n))$ se esiste un algoritmo che lo risolve avente complessitá $O(g(n))$
- $\Omega(g(n))$ se ogni algoritmo A che lo risolve ha complessitá $\Omega(g(n))$
- $\Theta(g(n))$ se ha complessitá $O(g(n))$ e $\Omega(g(n))$

problemi di decisione e classi di complessitá

i problemi di decisione sono solitamente descritti da un'istanza di input (o semplicemente INPUT) e da una DOMANDA sull'input esempi:

- soddisfacibilitá
  - INPUT: CNF (Conjunctive Normal Form) formula definita su un insieme di variabili
  - DOMANDA: esiste un assegnamento di veritá $\tau : V \to \{0,1\}$?
- clique
  - INPUT: un grafo non orientato $G = (V, E)$ di n nodi e un intero $k > 0$
  - DOMANDA: esiste in G una clique di almeno k nodi ($> k$), ovvero un sottoinsieme $U \subseteq V$ tale che $|U| \geq K$ e $\{u, v\} \in E$, $V \in U$?
- vertex cover
  - INPUT: un grafo non orientato $G = (V, E)$ di n nodi e un intero $k > 0$
  - DOMANDA: esiste in G una vertex cover di al massimo k nodi ($< k$), ovvero un sottoinsieme $U \subseteq V$ tale che $|U| \leq K$ e $u \in U$ o $v \in U$, $\forall \{u, v\} \in E$?

nei problemi di decisione $I_\pi = Y_\pi \cup N_\pi$

- $Y_\pi =$ insieme di istanze positive, ovvero con soluzione 1
- $N_\pi =$ insieme di istanze negative, ovvero con soluzione 0

def: un algoritmo A risolve π

un algoritmo A risolve $\pi \iff \forall$ input $x \in I_\pi$, A risponde $1 \iff x \in Y_\pi$

def: classe dei problemi $TIME(g(n))$

$TIME(g(n)) =$ classe dei problemi di decisione con complessitá $O(g(n))$

algoritmi non-deterministici per i problemi di decisione

essi si compongono di 2 fasi

- fase 1
  - generano in modo non-deterministico un "certificato" y
- fase 2
  - partendo dall'input x e dal certificato y, verificano se x é un'istanza positiva

def: un algoritmo non-deterministico A risolve π

un algoritmo non-deterministico A risolve π se si ferma per ogni possibile certificato y ed esiste un certificato y per cui A risponde 1 (vero) $\iff x \in I_\pi$

- complessitá
  - costo della fase 2
  - espressa in funzione di $|x|$

def: classe dei problemi $NTIME(g(n))$

$NTIME(g(n)) =$ classe di problemi di decisione con complessitá non-deterministica $O(g(n))$

esempio: algoritmo non-deterministico per il problema della clique

- fase 1
  - dato in input il grafo $G = (V, E)$, genera non-deterministicamente un sottoinsieme $U \subseteq V$ di k nodi
- fase 2
  - verifica se U é una clique, ovvero se $\{u, v\} \in E, \forall u, v \in U$, e in tal caso risponde 1, altrimenti risponde 0
- chiaramente l'algoritmo risolve il problema della clique, in quanto si ferma per ogni possibile sottoinsieme U ed esiste un sottoinsieme U per il quale risponde 1 se e solo se esiste una clique di k nodi in G, ovvero $\iff (G, K) \in Y_{clique}$
- complessitá: $O(n^2)$, poiché $|U| \leq |V| = n$

osservazioni (algoritmi deterministici e non-deterministici)

- un algoritmo deterministico é meno potente di uno non deterministico poiché non puó eseguire la fase 1
- se esiste un algoritmo deterministico A che risolve π, allora esiste anche un algoritmo non-deterministico A' che risolve π con la stessa complessitá come segue:
  - esso esegue al fase 1 e coincide con A nella fase 2, ignorando il certificato y

corollario: $TIME(g(n)) \subseteq NTIME(g(n))$

$$TIME(g(n)) \subseteq NTIME(g(n))$$

- dove:
  - $TIME(g(n)) =$ classe dei problemi deterministicamente risolvibili in tempo $O(g(n))$
  - $NTIME(g(n)) =$ classe dei problemi non-deterministicamente risolvibili in tempo $O(g(n))$

efficienza e trattabilitá

- un problema é trattabile se puó essere risolto efficientemente (deterministicamente)
- sono considerati trattabili o efficientemente risolvibili tutti i problemi aventi complessitá limitata da un polinomio della dimensione dell'input

$$\text{TRATTABILITÁ} \equiv \text{EFFICIENZA} \equiv \text{POLINOMIALITÁ}$$

efficienza e trattabilitá: ragione 1

la crescita delle funzioni polinomiali rispetto a quelle esponenziali (sia per ció che riguarda il tempo di esecuzione sia per ció che riguarda la dimensione delle istanze risolvibili entro un certo tempo di esecuzione)

efficienza e trattabilitá: ragione 2

- la composizione di polinomi é un polinomio e dunque la risolvibilitá in tempo polinomiale di un problema é indipendente da
  - il codice naturale utilizzato, poiché tutti i codici naturali sono correlati in maniera polinomiale
  - il modello computazionale adottato, se ragionevole (cioé costruibile nella pratica o meglio in grado di eseguire un lavoro limitato costante per step), in quanto tali modelli sono polinomialmente correlati, ovvero possono simularsi l'un l'altro in tempo polinomiale

osservazione: macchina di turing non-deterministica

la macchina di turing non-deterministica non é un modello di calcolo ragionevole, poiché la quantitá di lavoro svolto in ogni fase (ciascun livello dell'albero delle computazioni) cresce in modo esponenziale

optimization problems body

approximation body

greedy body

local search body

rounding body

primal dual body

dynamic programming body

approximation schemes body

alternative approaches body

social networks and bibliography body

centrality measures body

spectral analysis and prestige index body

link analysis body

web structure body

search and advertising body

matching markets body

auctions body

vcg mechanism body

gsp mechanism body