Dati e Algoritmi: A. Pietracaprina

Lezione dell'8 Gennaio 2026

Esempio Scritto 6: domanda 1

Siano A e B due algoritmi che risolvono lo stesso problema Π. Si definisca $t_X(n)$ la complessità al caso pessimo di X e $t_{X,i}$ il numero di operazioni eseguite da X per risolvere l'istanza i, per $X \in \{A, B\}$. Sapendo che $t_{B,i} \geq (1/10)t_{A,i}$ per ogni i, e che $t_A(n) \in \Theta\left(n^2\right)$, cosa si può dire di $t_B(n)$? Motivare la risposta.

$t_B(n) \in \Omega(n^2)$

Motivazione: Sia i^* = istanza peggiore per A

$\Rightarrow t_{A,i^*} = c \cdot n^2$ $c > 0$ costante

$t_{B,i^*} \geq \frac{c}{10} n^2 = c' \cdot n^2$ $c' = c/10 > 0$ costante

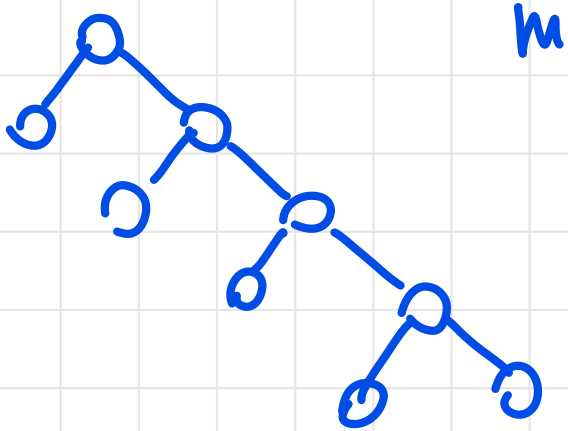
$\Rightarrow t_B(n) \in \Omega(n^2)$

Esempio Scritto 2: domanda 2

Sia T un albero binario proprio non vuoto di altezza h con n nodi di cui m sono foglie. Sapendo che $m \geq h + 1$:

1. Disegnare un albero T in cui il bound vale con uguaglianza ($h = 4$, $m = 5$).
2. Dimostrare che $n \geq 2h + 1$, usando, senza dimostrarla, la relazione nota tra numero di nodi interni e foglie.

1



$m = 5 \quad h = 4$

2 $n \geq 2h+1$ Sapendo che $m \geq h+1$

Relazione tra #nod interni e foglie

$m = n - m + 1 \Rightarrow$

$n = 2m - 1$

$\geq 2(h+1) - 1$ (dal test della domanda)

$= 2h + 1$

Esempio Scritto 2: domanda 3

Con riferimento alla Mappa:

1. Dare la specifica precisa del metodo `put`.
2. Supponendo di implementare la Mappa tramite lista position-based, indicare la complessità di `put` giustificando la risposta.

① Mappa M: M.put(k, x)

* Se M contiene già una entry (k, x')
  allora: - restituisce x'
  - sostituisce x' con x
* Se M non contiene alcuna entry

con chiave K allora

- restituisce null
- aggiunge a M una entry (K, x)

② Sia n il # entry nella mappa M la complessità di M.put(K, x) (al caso peggiore) è $\Theta(n)$ dato che se K non è presente in M devo guardare tutte le entry per accertarmi di questa situazione

Esempio Scritto 1: domanda 4

Si consideri un grafo $G = (V, E)$.

1. Dare la definizione di *spanning forest* di G.
2. Si supponga che G abbia 3 componenti connesse, G_1, G_2 e G_3. Detti n_i il numero di vertici e m_i il numero di archi di G_i, siano: $n_1 = 5, m_1 = 10$; $n_2 = 6, m_2 = 15$; $n_3 = 8, m_3 = 8$. Qual è il numero massimo di archi in una spanning forest di G? Motivare la risposta.

① Spanning forest =

* Spanning subgraph $G' = (V' = V, E' \subseteq E)$
* Senza cicli

② Sia $G_i = (V_i, E_i)$ $i = 1, 2, 3$

$V = V_1 \cup V_2 \cup V_3 \qquad E = E_1 \cup E_2 \cup E_3$

$G' = (V', E')$ è uno spanning forest di G

$\Rightarrow V' = V \qquad E' \subseteq E$ senza c.ch.

$E'_1 = E' \cap E_1 \qquad E'_2 = E' \cap E_2 \qquad E'_3 = E' \cap E_3$

$|E'_1| \leq n_1 - 1$

$|E'_2| \leq n_2 - 1$

$|E'_3| \leq n_3 - 1$

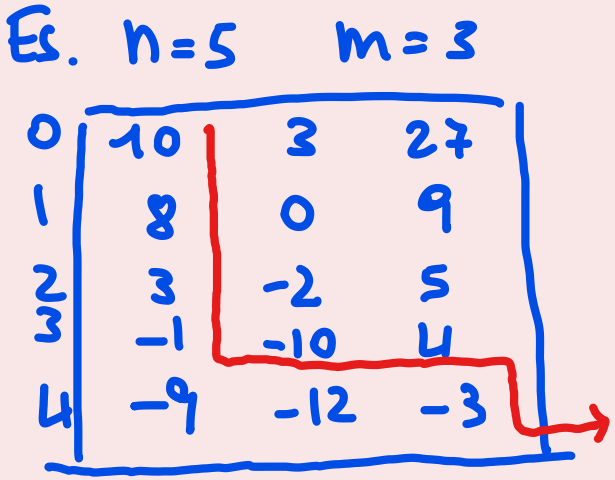
} perché un grafo senza cd. ha al più #vertici − 1 archi.

$$\Rightarrow |E'| = \sum_{i=1}^{3} |E'_i| \leq \left(\sum_{i=1}^{3} n_i \right) - 3 = n - 3 = 16$$

Esempio Scritto 1: esercizio 1

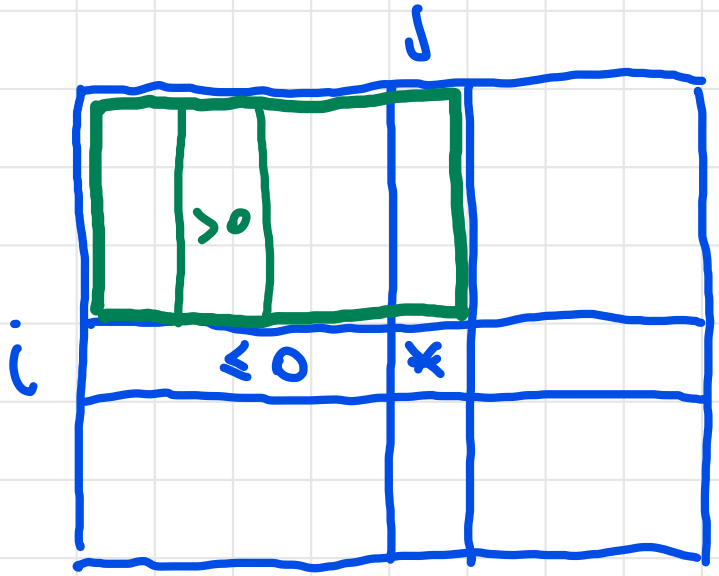
Sia A una matrice $n \times m$ con valori interi, con la proprietà che i valori in ogni colonna sono ordinati in senso decrescente dalla riga 0 alla riga $n-1$. Il seguente algoritmo determina il massimo numero di valori > 0 in una colonna.

```
i ← 0; j ← 0;
while (i < n) AND (j < m) do{
   if (A[i,j] ≤ 0) then j ← j + 1;
   else i ← i + 1;
}
return i;
```



Trovare un opportuno invariante per il ciclo, che serva per provare la correttezza dell'algoritmo (la prova di correttezza non è richiesta).

Invariante che deve valere alla fine di una generica iterazione del while, quando il prossimo elemento da guardare è A[i,j]



* Nella sottomatrice (in verde) formata dagli elementi nelle prime i righe (0, 1, ..., i-1)

e nelle prime $j+1$ colonne $(0, 1, \ldots, j)$ di A c'è una colonna con i elementi > 0

* $A[i, l] \leq 0 \quad \forall\, 0 \leq l < j$

Esempio Scritto 5: esercizio 1

Sia T un albero binario di ricerca che implementa una mappa le cui entry sono coppie (t, i_t), dove t (la chiave) rappresenta un istante di tempo in cui è avvenuto un terremoto, e i_t (il valore) rappresenta l'intensità del terremoto. Per ogni nodo $v \in T$ esiste un campo $v.\texttt{max}$ che riporta la massima intensità dei terremoti rappresentati dalle entry di T_v (sottoalbero con radice v).

1. Progettare un algoritmo ricorsivo `CheckGG` che, dati un tempo t e un valore di intensità i, restituisce `yes` se c'è stato un terremoto di intensità $\geq i$ in un istante di tempo $\geq t$, altrimenti restituisce `no`. Specificare chiaramente l'input e l'output dell'algoritmo.
2. Analizzare la complessità dell'algoritmo del punto precedente.

Algoritmo CheckGG (T, v, t, i)

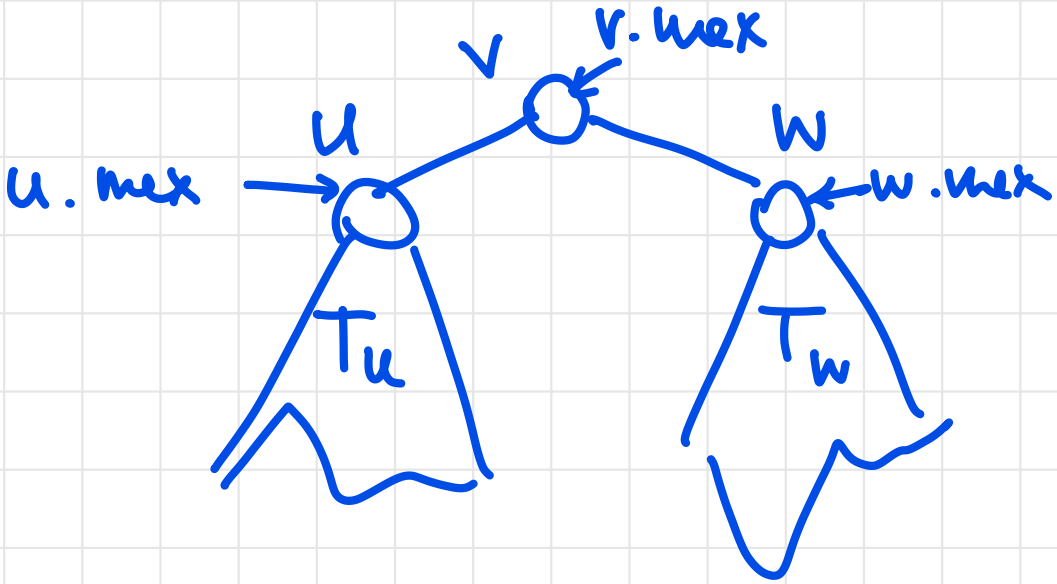
input nodo $v \in T$, istante t, intensità i

output yes, se T_v contiene una entry (t', i')

con $t' \geq t$ e $i' \geq i$

no, altrimenti

Prima invocazione: $v = T.root()$



Sia (t_v, i_v) la entry in v

Se $v.mex < i \Rightarrow$ return no

Altrimenti $(v.mex \geq i)$:

* Se $t_v < t \Rightarrow$ return CheckGG(T, w, t, i)
* Se $t_v \geq t$ allora
  - Se $i_v \geq i$ or $w.mex \geq i \Rightarrow$ return yes
  - Altrimenti return CheckGG(T, u, t, i)

Pseudocodice: esercizio

② Complessità $\Theta(h)$ h altezza di T

Motivazione: Stessa struttura della TreeSearch

Esempio Scritto 4: esercizio 2

Progettare un algoritmo che, date due sequenze ordinate S_1, S_2, ciascuna contenente n interi distinti, determini il numero di elementi $k \in S_1$ per cui S_2 contiene k^2. L'algoritmo deve avere complessità $\Theta(n)$.
(Suggerimento: modificare Merge)

Esempio: $n = 4$

$S_1 = 3 \quad 10 \quad 11 \quad 15$

$S_2 = 6 \quad 9 \quad 121 \quad 367$

} Output $= 2$

S_1^2 = sequenza S_1 con le singole elementi al quadrato

$\Rightarrow$ nell'esempio: $S_1^2 = 9 \quad 100 \quad 121 \quad 225$

Alla fine quello che dobbiamo calcolare è $|S_1^2 \cap S_2|$

l'intersezione tra 2 sequenze ordinate è stata già affrontata in un esercizio (Problema 3 delle dispense di esercizi svolti)

IDEA: Simulare il merge di S_1^2 e S_2 senza memorizzare una sequenza di output memorizzando traccia del valore $|S_1^2 \cap S_2|$

```
Algorithm Count(S1, S2)
input ...
output ...
i ← 0; j ← 0; C ← 0;
while ((i < n) AND (j < n)) do {
  if ((S1[i])^2 = S2[j]) then {C++; i++; j++}
  else if ((S1[i])^2 < S2[j]) then i++
       else j++
}
return C
```

Complexità : $\Theta(n)$

Selezione di domande dai Quiz

Sia A un algoritmo per il problema computazionale $\Pi_1 = (\mathcal{I}_1, \mathcal{S})$, e si supponga che la sua complessità temporale al caso pessimo sia $\Theta(n^2)$. Sia B un altro algoritmo che risolve il problema computazionale $\Pi_2 = (\mathcal{I}_2, \mathcal{S})$, con $\mathcal{I}_2 \subset \mathcal{I}_1$, e che per ogni istanza $i \in \mathcal{I}_2$, chiama $A(i)$, ed esegue un numero costante di altre operazioni elementari. Per ogni $\mathcal{I}_2$, si può dimostra che la complessità temporale al caso pessimo $t_B(n)$ di B è

Select one choice.

- $\Theta(n^2)$
- $\mathcal{O}(n^2)$ ✗
- $\Omega(n^2)$

Sia A un algoritmo per un problema computazionale Π. Si supponga che per ogni istanza di taglia n, A esegue $\leq 10n^2$ operazioni. Se la complessità di A fosse $\Theta(n^2)$, quale altra proprietà, tra le seguenti, deve *sicuramente* valere?

Select one choice.

- Non esiste alcuna istanza di taglia n per cui A esegue $10n \log n$ operazioni
- X Esiste un'istanza di taglia n per la quale A esegue $\geq cn^2$ operazioni, con $c > 0$ un'opportuna costante
- Per ogni istanza di taglia n, A esegue $\geq n^2$ operazioni
- Per ogni istanza di taglia n, A esegue almeno $c_1 n^2$ e al più $c_2 n^2$ operazioni con $c_2 > c_1 > 0$ opportune costanti

Sia T un albero in cui ciascun nodo $v \in T$ contiene un intero $v\,.\,\texttt{val}$.
Il seguente algoritmo

Algoritmo $\texttt{minVal}(v)$
Input: $v \in T$
Output: $\min\{u\,.\,\texttt{val} : u \in T_v\}$
$s \leftarrow v\,.\,\texttt{val};$
foreach $w \in T\,.\,\texttt{children}(v)$ **do**
    $s \leftarrow \min\{s, \texttt{minVal}(w)\};$
return $s;$

segue lo schema di visita

Select one choice.

- preorder
- postorder
- inorder
- disorder

Si consideri un algoritmo che, dato in input un albero T di n nodi, invoca $\texttt{depth}(v)$ da ciascun nodo interno $v \in T$, ritornando il massimo valore restituito $+1$. Dire quale delle seguenti affermazioni è *vera*.

Select one choice.

- L'algoritmo restituisce la massima altezza di una foglia di T in tempo $\Theta(n)$
- L'algoritmo restituisce l'altezza della radice di T in tempo $\Theta(n^2)$
- L'algoritmo restituisce il numero di foglie di T in tempo $\Theta(n)$
- L'algoritmo restituisce la massima altezza di una foglia di T in tempo $\Theta(n^2)$

La condizione $\frac{n-1}{2} < h \leq n-1$ è verificata:

Select one choice.

- solo da alcuni alberi con n nodi e altezza h, in cui ogni nodo interno ha esattamente 3 figli
- X solo da alcuni alberi binari non propri con n nodi e altezza h
- solo da alcuni alberi binari propri con n nodi e altezza h
- mai

Sia T un albero binario proprio con n nodi, m foglie, e altezza h. Indicare, tra le seguenti, l'unica relazione che *non può essere mai verificata*:

Select one choice.

- $h = (n-1)/4$
- X $h = n - m + 1$
- $m = (n+1)/2$
- $m = 2^h$

h = max profondità di una foglia che è $\leq$ # nod interni = $n - m$