Progettazione di Algoritmi
9/7/2020

Inviare l'elaborato a algoritmi2020@gmail.com

9 cfu nuovo

1	2	3	4	5	Totale
/18	/20	/22	/18	22/	/100

9 cfu vecchio

1	2	3	4	5	6	Totale
/18	/20	/22	/10	17/	/13	/100

6 cfu nuovo

3	4	5	Totale
/40	/32	/28	/100

6 cfu vecchio

3	4	5	6	Totale
/33	/22	/20	/25	/100

Si ricorda che per i punti che richiedono l'analisi di un algoritmo occorre fornire un limite superiore asintotico quanto migliore e` possibile al tempo di esecuzione dell'algoritmo giustificando la risposta.

1. Analisi degli algoritmi e notazione asintotica

a) Indicare quali delle seguenti affermazioni sono vere e quali sono false.
   1. $4n^4 = O(n^3)$ F
   2. $n^{\log n} = \Omega((\log n)^n)$ V
   3. $n(\log n)^{1/2} + n^{1/3} = O(\log n^n)$ V
   4. $n^3+10 = \Omega(n^3+10n)$ V
   5. $2^{\log_4 n} = \Theta(n)$ F

$n\sqrt{\log n} + \sqrt[3]{n} = O(n \log n)$

b) Si dimostri che se $1<f(n)=O(h(n))$ allora $(f(n))^a = O(h(n)^a)$, dove a è una costante positiva. Occorre utilizzare solo la definizione di O e nessuna altra proprieta`.

c) Si analizzi il tempo di esecuzione nel caso pessimo del seguente segmento di codice fornendo una stima asintotica **quanto migliore e` possibile** per esso. **Si giustifichi in modo chiaro la risposta.**

```
i=1
j=1
while(i<n and j<m)
  if (i <= j)
    i=i+1
  else
   j=j+1
```

$k/2 = n \rightarrow k = 2n$

$k/2 = m \rightarrow k = 2m$

$k = 2n-1$

$\frac{k}{2} = n \quad k = 2n-1$

$\frac{k}{2} = m \quad k = 2m$

$\frac{10}{2} = 5+1 = 6$

$\frac{10}{2} = 5 = 5$

$10 =$

$n^{\log n} = \Omega((\log n)^n) \iff \exists\ c>0,\ n_0 \geq 0 \mid n^{\log n} \geq c \log^n n\ \forall n \geq n_0$

$c \log^n n \leq n^{\log n} \qquad c \leq \frac{n^{\log n}}{\log^n n}$

b) Si dimostri che se $1<f(n)=O(h(n))$ allora $(f(n))^a = O(h(n)^a)$, dove a è una costante positiva. Occorre utilizzare solo la definizione di O e nessuna altra proprieta`.

$f(n) = O(h(n)) \iff \exists\ c'>0,\ n'_0 \geq 0 \mid f(n) \leq c' h(n)\ \forall n \geq n'_0$

$f(n)^a \leq c'^a\, h(n)^a \Rightarrow$ prendiamo $c'' = c'^a$ e $n''_0 = n'_0$

abbiamo che $f(n)^a \leq c'' h(n)^a\ \forall n \geq n''_0$ dunque dalla def di O abbiamo che $f(n)^a = O(h(n)^a)$

Progettazione di Algoritmi
9/7/2020

2. Divide et Impera

a) [9 cfu nuovo] Si descriva il comportamento dell'algoritmo per determinare la coppia di punti piu` vicina in un insieme di punti del piano.

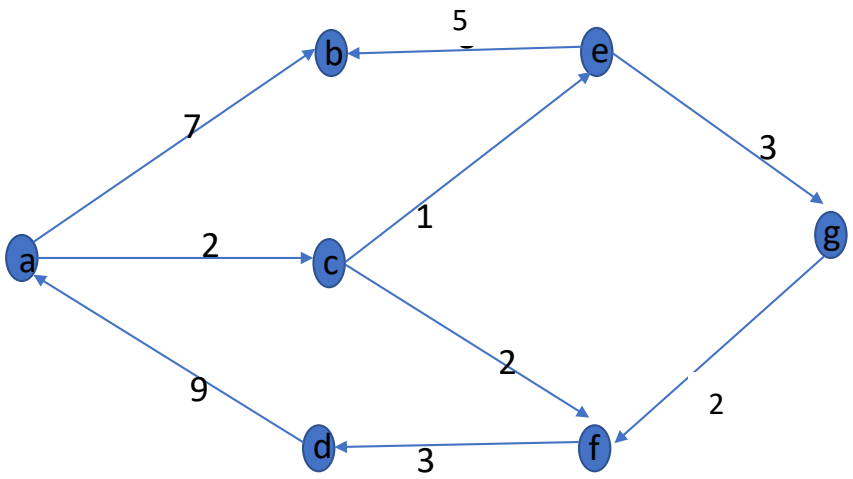
b) Si fornisca la relazione di ricorrenza che esprime un limite superiore al tempo di esecuzione dell'algoritmo QuickSelect nel caso pessimo. **Si giustifichi in modo chiaro la risposta.**

c) A partire dalla relazione di ricorrenza da voi fornita al punto b), si fornisca una funzione h(n) tale T(n)=O(h(n)). Giustificare la risposta usando il metodo iterativo.

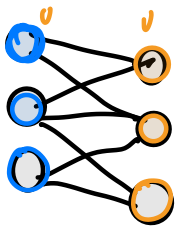
3. Grafi

a) Si scriva lo pseudocodice **dell'algoritmo BFS** che fa uso di una **coda FIFO** aggiungendo anche le linee di codice per la costruzione dell'albero BFS. Si analizzi il tempo di esecuzione dell'algoritmo proposto.

b) Si mostri l'esecuzione dell'algoritmo di Dijkstra sul seguente grafo a partire dal nodo sorgente **a. Per ogni passo si mostri il contenuto della coda a priorita`, incluse le chiavi degli elementi, e l'albero dei percorsi minimi costruito fino a quel passo.**



c) Si consideri l'algoritmo che determina se un grafo è bipartito. Si dimostri che se l'algoritmo colora due nodi adiacenti dello stesso colore allora il grafo non è bipartito.



arraycolor[u] = arraycolor[v]

sorgente/parametro

BFS(s) — d = discovered

init Q empty → O1
init T as BFS empty → O1
set d[s] = true → O1
e foreach d[v] = false → O(n)
enqueue Q[s] → O1
while Q != empty
  u = dequeque(Q) ↓ O(n·m)
  foreach n adiacenti a u
    se d[n] = false → O1
      d[n] = true → O1
      enqueque(v) → O1
      aggiungi arco u,v a T – O1

    end if
  end foreach
end while

$\sum_{u \in V} deg(u)$

$deg(u) = 2m = O(m)$

$O(n+m)$

pari discovered [s] = true
// discovered [v] per tutti i nodi del grafo = false $O(n)$
l(0) = s
BFS-T init
array color [v] per tutti i nodi = nulle $O(n)$
i = 0
while (i ≤ n-2)
    l(i+1) = ∅
    foreach u ∈ l(i)
        foreach n adiacente ad u
            se discovered [n] = false
                discovered [n] = true
                aggiungi n ad l(i+1) ←
                se i è pari
                    arraycolor [n] = red
                altrimenti
                    arraycolor [n] = blue
                aggiungi (u,v) ad BFS-T
                se arraycolor [u] == arraycolor [v]
                    allora non è bipartito
            endif
        endif
    endforeach
end foreach
i = i+1
end while

$deg(v) \geq 2m = m$

Progettazione di Algoritmi
9/7/2020

4. **Algoritmi greedy**

a) Si spieghi in che cosa consiste un'istanza (input) del problema del partizionamento di intervalli e qual è l'obiettivo del problema (output). Se dalla risposta a questo punto si evincera` che lo studente non sa in cosa consiste il problema, i punti successivi dell'esercizio non saranno valutati.

b) Si scriva lo pseudocodice dell'algoritmo greedy che restituisce il valore della soluzione ottima per il problema del partizionamento di intervalli. SI SCRIVA LO PSEUDOCODICE IN ITALIANO: le uniche parole inglesi consentite sono le parole chiave if, for, ecc. .

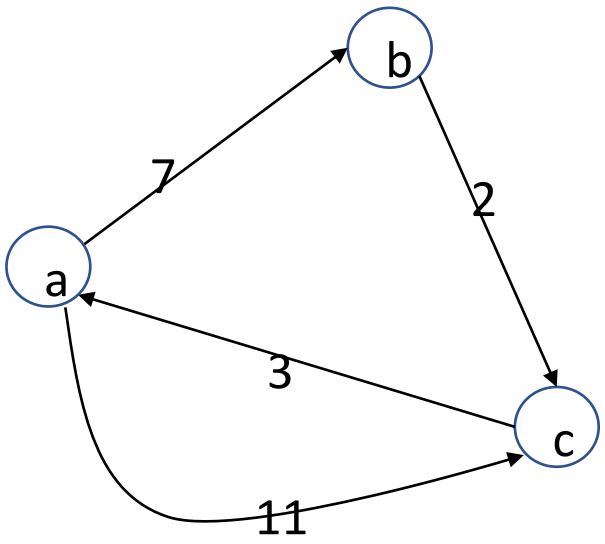
c) [6 cfu tutti e 9 cfu nuovo] Si fornisca un'istanza del problema **dell'Interval Scheduling** per la quale la strategia greedy "Fewest Conflicts" non fornisce una soluzione ottima. Occorre indicare i tempi di inizio e di fine di ogni job.

5. **Programmazione dinamica**

a) Scrivere lo psudocodice di un algoritmo basato sul principio di programmazione dinamica per computare la sottosequenza comune piu` lunga di due sequenze.

b) Fornire una formula per il calcolo del valore della soluzione ottima OPT del problema dei cammini minimi in termini di valori delle soluzioni ottime per sottoproblemi di taglia piu` piccola. Spiegare in modo chiaro
   1. cosa rappresenta la funzione OPT e cosa rappresentanto i suoi parametri
   2. come si arriva alla formula da voi fornita.

c) Si consideri il seguente grafo e **si disegni la matrice 3 x 3** dei valori computati dall'algoritmo di Bellman-Ford per il problema dei cammini minimi quando il nodo t ricevuto in input dell'algoritmo è il nodo **a**.



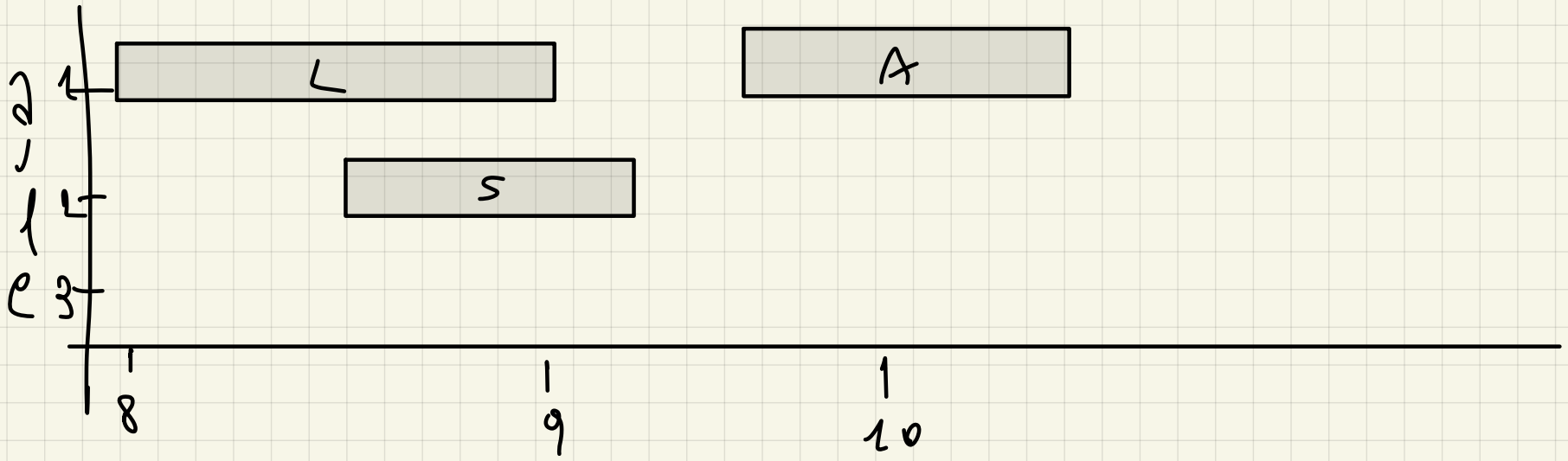
Il partizionamento degli intervalli consiste nel avere n-attività e delle risorse disponibili e riuscire ad utilizzare il minor numero di risorse per queste attività.

Come input del problema abbiamo un insieme n-intervalli $(start_1 - finish_1], [s_2 - f_2), \ldots, [s_n - f_n)$ che rappresentano gli intervalli di tempo le quali si svolgono le attività.

L'obiettivo (output) è far svolgere le n-attività con il minor numero di risorse avendo 2 vincoli

1) ogni attività può avere al + 1 risorsa

2) " " necessita di una sola risorsa



CONTINUA →

Ordino gli intervalli per tempo di inizio decrescente

$O(n \log n)$

Init $d = \emptyset$

foreach n in intervalli

se l'intervallo n può essere assegnato ad una risorsa già presente V

assegno V a n

altrimenti

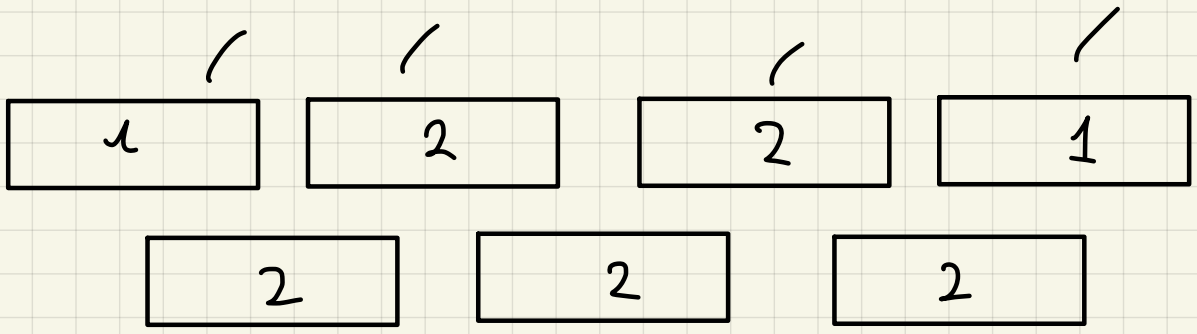
alloco una nuova risorsa $d+1$

assegno la risorsa $d+1$ all'intervallo j

$d = d + 1$

end foreach

Prendiamo come esempio il seguente



In questo caso l'algoritmo invece di selezionare 4 job ne selezionerà solo 2