Progettazione di Algoritmi 15/9/2022

Cognome e Nome:
Numero di Matricola:

Spazio riservato alla correzione

1	2	3	4	5	Totale
/20	/17	/23	/22	18	/100

1. Grafi

a) Si scriva lo pseudocodice di un algoritmo BFS **senza** coda FIFO che abbia tempo di esecuzione O(n+m) .

b)

I. Si descriva un algoritmo che prende in input un grafo G non orientato e restituisce true se il grafo G e` bipartito e false altrimenti. L'algoritmo deve avere tempo di esecuzione O(n+m).

II. Si descriva come modificare l'algoritmo al punto I in modo tale che nel caso in cui il grafo G non sia bipartito, l'algoritmo fornisca una prova del fatto che G è effettivamente un grafo non bipartito. La prova deve essere convincente anche per una persona che **non** sa come funziona l'algoritmo al punto I e sa **solo** verificare che uno o piu` archi appartengano al grafo.

b) Si mostri l'esecuzione dell'algoritmo per l'ordinamento topologico sul DAG formato dai seguenti archi (a,b),(a,g), (a,e) , (b,c), (b,d), (c,d), (e,c), (g,e). Si disegnino i grafi su cui vengono invocate le chiamate ricorsive e alla fine si fornisca l'ordinamento topologico del grafo.

2. Algoritmi greedy

a) Fornire un'istanza del partizionamento di intervalli di dimensione n=9 e con valore della soluzione ottima uguale a 4. Indicare inoltre in quali istanti l'algoritmo greedy che trova la soluzione ottima incrementa il valore della soluzione **per la vostra istanza**. **Non è sufficiente fornire un disegno che descriva l'input.**

b) Si consideri il problema della minimizzazione dei ritardi. Si dimostri che in scheduling senza inversioni e senza idle time, job con la stessa scadenza vengono eseguiti uno dopo l'altro.

c) Si mostri l'esecuzione dell'algoritmo di Dijkstra sul grafo direzionato contenente i seguenti archi (a,b)6, (a,c)3,(a,d)5, (a,e)8 , (b,e)2,(c,d)1,(c,e)4,(c,f)3, (d,f)1. I numeri rossi sono i pesi degli archi. La radice dell'albero dei cammini minimi deve essere il nodo a. **Per ogni passo occorre mostrare l'albero costruito fino a quel momento e il contenuto della coda a priorita` (indicando anche le chiavi).**

3. **Programmazione dinamica [25 minuti]**

a) Si consideri il problema del problema **minimum coin change** problem.
   I. Si dica per quale input OPT(i,v) rappresenta il valore della soluzione ottima.
   II. Si dica cosa è una soluzione per il problema e cosa è una soluzione ottima.
   III. In quale caso accade che il valore della soluzione ottima è $OPT(i,v-v_i) + 1$? Perche' in questa espressione il primo parametro di OPT è i?

b) Scrivere lo pseudocodice dell'algoritmo di programmazione dinamica per il problema **minimum coin change problem**. **Analizzare il tempo di esecuzione giustificando nel dettaglio la risposta (non serve dare solo il tempo di esecuzione).**

c) Si fornisca la tabella dei valori computati dall'algoritmo di programmazione dinamica che calcola il valore della soluzione ottima per il problema della sottosequenza comune piu` lunga quando le due sequenze input sono x=BACD e y=BAAD. Alla fine si fornisca la soluzione ottima e si indichino con un cerchio le entrate sui cui indici vengono effettuate le chiamate ricorsive dall'algoritmo che stampa la soluzione ottima.

d) Fornire la tabella costruita dall'algoritmo che computa il valore della soluzione ottima per il problema dello zaino quando l'istanza input è $w_1=1$, $w_2=5$, $w_3=6$, $w_4=5$, $w_5=4$, $v_1=3$, $v_2=10$, $v_3=16$, $v_4=5$, $v_5=18$, $W=6$. Al termine fornire la soluzione ottima e indicare con un cerchietto le entrate della tabella sui cui indici vengono effettuate le chiamate ricorsive dell'algoritmo che stampa la soluzione ottima.

4. **Analisi degli algoritmi e notazione asintotica [20 minuti]**

a) Indicare quali delle seguenti affermazioni sono vere e quali sono false.
   1. $n^2 \log^{10} n = O(n^3)$
   2. $n^{1/4} = \Omega(n^{1/4} + \log n)$
   3. $n^{1/4} = \Omega(n^{1/4} + n^{1/3} \log n)$
   4. $n^3+10n^2+8=O(n^3)$
   5. $\log(\log^3 n)=O((\log n)(\log n))$

b) Si dimostri che se $0<f(n)=O(h(n))$ e $0<g(n)= O(p(n))$ e **a** e **b** sono costanti positive allora $\mathbf{a}f(n)+\mathbf{b}g(n)= O(h(n)+p(n))$. Occorre utilizzare solo la definizione di O e nessuna altra proprieta`.

c) Dimostrare che la seguente affermazione e` vera giustificando la risposta. Occorre fornire le costanti c ed n_0.
$n^2+ n\log n=O(n^2)$

d) Dimostrare che la seguente affermazione e` falsa giustificando **matematicamente** la risposta.
$n^3= \Omega(n^3 \log n)$

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

```
FOR(i=n; i>=1; i=i/4){
  FOR(j=1; j<500; j=j*3) {
      print(j);
   }
  print(i)
}
```

5. **Divide et Impera [20 minuti]**

a) Si scriva lo pseudocodice di un algoritmo ricorsivo che prende in input un array (**ed eventualmente altri input**) e computa la lunghezza della sottosequenza che contiene il massimo numero di occorrenze consecutive dello stesso elemento (l'elemento non e` dato in input). È sufficiente che l'algoritmo abbia tempo O(nlog n). Bonus se l'algoritmo è lineare ma non ci perdete tempo e fornite **un'unica versione dell'algoritmo!**
Evitate di invocare algoritmi ausiliari (Algoritmi che usano algoritmi ausiliari saranno valutati con punteggio inferiore. Tra questi algoritmi ausiliari non rientrano quelli che effettuano il lavoro di decomposizione o di ricombinazione che quindi possono essere usati nella versione a punteggio pieno).

b) Si fornisca la relazione di ricorrenza che esprime il limite superiore al tempo di esecuzione dell'algoritmo al punto a). Si giustifichi in modo chiaro la risposta.

c) A partire dalla relazione di ricorrenza da voi fornita al punto b) per il caso pessimo, si fornisca una funzione h(n) tale che T(n)=O(h(n)). **Giustificare la risposta** usando il metodo iterativo o quello della sostituzione (induzione). Se proponete la versione con bonus pensata bene a quale dei due metodi vi convenga usare.