Algoritmi e Strutture Dati

a.a. 2022/23

Compito del 20/6/2023

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte I

(30 minuti; ogni esercizio vale 2 punti)

Avvertenza: Si giustifichino tecnicamente tutte le risposte. In caso di discussioni poco formali o approssimative gli esercizi non verranno valutati pienamente.

1. Si risponda vero o falso alle seguenti domande, motivando brevemente le risposte:
   a. la scelta dell'elemento pivot nella *Partition* non influenza la complessità asintotica di *QuickSort*.
   b. *Partition* può restituire valori diversi se le vengono forniti elementi pivot diversi.
   c. Dato un *maxHeap* incrementare il valore di una chiave di un elemento in posizione data ha costo O(n).
   d. *Counting Sort* è un algoritmo che ordina un qualunque array di *n* interi in tempo *O(n)*.

2. Si scriva la formula di aggiornamento delle matrici dell'algoritmo di Floyd-Warshall e si mostri che, in assenza di cicli negativi, la diagonale principale delle matrici prodotte durante l'esecuzione dell'algoritmo è sempre nulla.

3. Siano $\mathcal{P}$ e $\mathcal{Q}$ due problemi in NP con complessità $T_{\mathcal{P}}(n)$ e $T_{\mathcal{Q}}(n)$, rispettivamente, e si supponga che $\mathcal{P} \leq_P \mathcal{Q}$. Si stabilisca se le seguenti implicazioni sono vere o false:

   a) $T_{\mathcal{Q}}(n) = O(n^3) \Rightarrow T_{\mathcal{P}}(n) = O(n^3)$
   b) $T_{\mathcal{Q}}(n) = O(3^n) \Rightarrow T_{\mathcal{P}}(n) = O(3^n)$
   c) $\mathcal{Q} \in \mathrm{P} \Rightarrow \mathcal{P} \in \mathrm{P}$
   d) $\mathcal{P} \in \mathrm{NPC} \Rightarrow \mathcal{Q} \in \mathrm{NPC}$

Algoritmi e Strutture Dati

a.a. 2022/23

Compito del 20/6/2023

Cognome: _____ Nome: _____

Matricola: _____ E-mail: _____

Parte II

(2.5 ore; ogni esercizio vale 6 punti)

Avvertenza: Si giustifichino tecnicamente tutte le risposte. In caso di discussioni poco formali o approssimative gli esercizi non verranno valutati pienamente.

1. Sia *T* un albero binario di ricerca (BST) avente *n* nodi.
   a. Si scriva una funzione **efficiente** in C o C++
      `PTree Modify_key(PTree t, PNode x, int key)` che modifica $x \rightarrow key$ con *key* e ritorna *t* se *t* è ancora un BST, *nullptr* altrimenti.
   b. Valutare e giustificare la complessità della funzione.
   c. Specificare il linguaggio di programmazione scelto e la definizione di `PTree` e `PNode`.

2. Sia *A* un vettore in cui ogni elemento contiene due campi: *A[i].value* contiene un numero intero ed *A[i].color* contiene un colore *(BIANCO o NERO)*. Gli elementi di *A* sono ordinati in ordine crescente rispetto al campo *value*.
   a. Si consideri il problema di ordinare gli elementi di *A* rispetto al campo *color* secondo l'ordinamento *BIANCO<NERO*, facendo in modo che gli elementi dello stesso colore siano ordinati rispetto al campo *value*. Si scriva lo pseudocodice di un algoritmo **efficiente** per risolvere il problema proposto. Si valuti e giustifichi la complessità.
   b. Si consideri il problema di ordinare gli elementi di *A* rispetto al campo *value*, facendo in modo che gli elementi che hanno stesso *value* siano ordinati rispetto al campo *color* (sempre con la convenzione *BIANCO<NERO*). Si scriva lo pseudocodice di un algoritmo **efficiente** per risolvere il problema proposto. Si valuti e giustifichi la complessità.

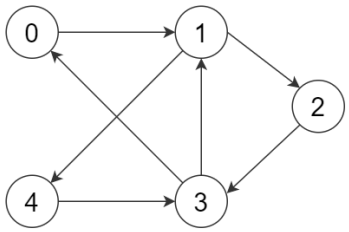
3. Si stabilisca quale problema decisionale risolve il seguente algoritmo, che riceve in ingresso un grafo orientato $G = (V, E)$ e un intero positivo *k* (si assuma per semplicità che il grafo non contenga cappi):

```
MyAlgorithm(G, k)
1.  m = +∞
2.  for each u ∈ V[G]
3.     m = min {m, MyFunction(G, u)}
4.  if m = k then
5.     return TRUE
6.  else
7.     return FALSE                    /* continua alla pagina successiva */
```

```
MyFunction(G, x)
1.  for each u ∈ V[G] \ {x}
2.     d[u] = +∞
3.  d[x] = 0
4.  Q = V[G]                          /* Q è una coda con priorità con campo chiave d */
5.  for i = 1 to |V[G]|
6.     u = ExtractMin(Q)
7.     for each v ∈ V[G]
8.        if (u,v) ∈ E[G] and d[u] + 1 < d[v] then
9.           d[v] = d[u] + 1
10.  a = +∞
11.  for each u ∈ V[G] \ {x}
12.     if (u,x) ∈ E[G] then
13.        a = min {a, d[u] + 1}
14.  return
```

Si dimostri la correttezza dell'algoritmo e si determini la sua complessità computazionale.

Cosa restituisce l'algoritmo in presenza del grafo sottostante con k=2, k=3 e k=4? Perché?



Si dica inoltre cosa restituisce `MyFunction` in corrispondenza dei 5 vertici del grafo.

4. Il sindaco di Venezia ha deciso di avviare un'operazione di ristrutturazione dei ponti della città al fine di tutelare la sicurezza dei suoi cittadini. Una limitata disponibilità finanziaria, tuttavia, impedisce di ristrutturare tutti i ponti e per questo impone le seguenti due condizioni:

    1. Deve essere possibile per i cittadini spostarsi da un punto all'altro della città attraversando *soltanto* ponti ristrutturati;

    2. Il costo complessivo della ristrutturazione deve essere il minore possibile.

Si formuli il problema in termini di ottimizzazione su grafi (indicando con precisione cosa rappresentano i nodi, gli archi e i pesi sugli archi) e si descriva un algoritmo per la sua risoluzione.

Si dimostri la correttezza dell'algoritmo utilizzato e si determini la sua complessità.

Infine, si simuli accuratamente la sua esecuzione su una versione semplificata del problema riguardante soltanto i sei sestieri di Venezia. A tal fine si utilizzi la tabella dei costi (in migliaia di euro) riportata di seguito (il simbolo "∞" significa che, tra due sestieri, non esiste un ponte):

	Cannaregio	Castello	Dorsoduro	Santa Croce	San Marco	San Polo
Cannaregio	--	56	∞	150	25	∞
Castello	56	--	∞	∞	30	∞
Dorsoduro	∞	∞	--	45	170	60
Santa Croce	150	∞	45	--	∞	40
San Marco	25	30	170	∞	--	240
San Polo	∞	∞	60	40	240	--