Algoritmi e Strutture Dati
a.a. 2018/19

Compito del 12/09/2019

Cognome: ______________________ Nome: ______________________

Matricola: ______________________ E-mail: ______________________

Parte I
(30 minuti; ogni esercizio vale 2 punti)

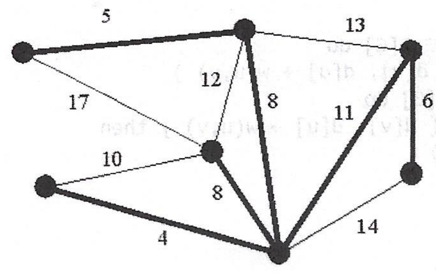
1. Scrivere l'algoritmo build-Max-Heap e simulare la sua esecuzione sull'array <-10, -3, -7, 15, 12, 36>

2. Il Prof. H. A. Milton sostiene di aver sviluppato un algoritmo di complessità

$$T(n) = 9T\left(\frac{n}{3}\right) + n^2$$

che riceve in ingresso un grafo non orientato G con n vertici e risponde TRUE se esiste in G un ciclo che passa per tutti i suoi vertici, e FALSE in caso contrario. Si dica, **giustificando tecnicamente la risposta**, se l'affermazione è verosimile.

3. Si dica, **giustificando tecnicamente la risposta**, se nel grafo sottostante gli archi indicati in grassetto formano o meno un albero di copertura minimo.



Algoritmi e Strutture Dati
a.a. 2018/19

Compito del 12/09/2019

Cognome: ______________________ Nome: ______________________

Matricola: ______________________ E-mail: ______________________

Parte II
(2.5 ore; ogni esercizio vale 6 punti)

1. Sia T un albero binario di ricerca. Si vogliono stampare le chiavi di T memorizzate in nodi il cui sottoalbero radicato nel figlio sinistro contiene un numero pari di chiavi e il sottoalbero radicato nel figlio destro contiene un numero dispari di chiavi.
   a. Si rappresenti un albero binario di ricerca la cui visita in pre-order ha come risultato 10, 5, 1, 20, 15, 25. Si mostri quali chiavi verrebbero stampate in base alla condizione sopra descritta.
   b. Scrivere una **procedura in C efficiente**, di nome stampaNodi(u), che data la radice di un albero binario di ricerca, stampa le chiavi dei nodi che soddisfano la condizione specificata. Valutarne la complessità, **indicando eventuali relazioni di ricorrenza.**

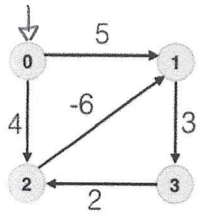
2. Siano dati in input k vettori $A_1, \ldots, A_k$ di numeri naturali, ognuno ordinato in modo decrescente. Sia n la quantità di elementi presenti complessivamente nei vettori, ovvero $n = \sum_{i=1}^{k} A_i.length$. Si consideri il problema di produrre in output il vettore ordinato in modo decrescente B, unione con ripetizioni di $A_1, \ldots, A_k$.
   a. Scrivere tramite pseudo-codice una procedura **efficiente** per risolvere il problema proposto nel caso in cui k sia costante rispetto ad n. Si determini la complessità.
   b. Scrivere lo pseudo-codice di una procedura per risolvere il problema proposto nel caso generico avente complessità $O(n \log k)$. Si determini la complessità.

3. Si stabilisca quale problema risolve il seguente algoritmo, che accetta in ingresso un grafo orientato e pesato $G = (V, E)$, la sua funzione peso $w : E \to \mathbb{R}$, e un vertice $s \in V$:

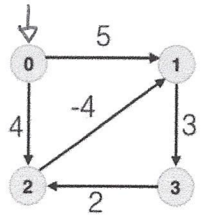
```
MyAlgorithm(G, w, s)
n = |V[G]|
for each u ∈ V[G] \ {s}
   d[u] = +∞
d[s] = 0
for i = 1 to n - 1
   for each (u,v) ∈ E[G] do
      d[v] = min { d[v], d[u] + w(u,v) }
for each (u,v) ∈ E[G] do
   if d[v] != min { d[v], d[u] + w(u,v) } then
      print("Yes")
   otherwise
      print("No")
return
```

Si dimostri la correttezza dell'algoritmo e si determini la sua complessità computazionale. Cosa restituisce l'algoritimo in presenza dei seguente grafi, ponendo $s = 0$?

YES



(a)

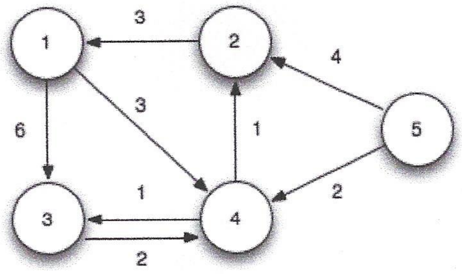


(b)

YES

Nota: si fornisca una dimostra "completa" della correttezza dell'algoritmo e si giustifichino tecnicamente tutte le risposte. In caso di discussioni poco formali o approssimative l'esercizio non verrà valutato pienamente.

4. Si scriva l'algoritmo di Floyd-Warshall, si dimostri la sua correttezza, si fornisca la sua complessità computazionale e si simuli accuratamente la sua esecuzione sul seguente grafo:



Nota: si giustifichino tecnicamente tutte le risposte. In caso di discussioni poco formali o approssimative l'esercizio non verrà valutato pienamente.

ESAME 12/09/19

PART I

1)
```
void buildMaxHeap (Heap A) {
    A.heapsize = A.length;
    for (int i = floor (A.length/2); i ≠ 0; i--)
        max_heapify(A, i);
}

void max_heapify (Heap A, int i) {
    int l = left(i)
    int r = right(i)
    int massimo;
    int temp;
    if (l ≤ A.heapsize && (A.arr[l] > A.arr[i]))
        massimo = l;
    else
        massimo = i;
    if r ≤ A.heapsize && A.arr[r] > A.arr[massimo])
        massimo = r;
    if (i ≠ massimo) {
        temp = A.arr[i];
        A.arr[i] = A.arr[massimo];
        A.arr[massimo] = A.arr[i];
        max_heapify (A, massimo);
    }
}
```

<-10, -3, -7, 15, 12, 36>

		i			
-10	-3	-7	15	12	36
1	2	3	4	5	6

left(i) → i*2

right(i) → i*2+1

$l=6 \quad r=7 \quad i=3$

max = 6

$\overset{i}{3} \neq \overset{max}{6}$ $\quad$ A[i] = -7

SWAP $\quad$ A[l] = A[massimo] = 36

-10	-3	36	15	12	-7

-10	-3	36	15	12	-7
1	2	3	4	5	6

-10	15	36	-3	12	-7

$l=4 \quad r=5 \quad i=2$

$max=4 \quad 2 \neq 4 \rightarrow$ SWAP

-10	15	36	-3	12	-7
1	2	3	4	5	6

36	15	-10	-3	12	-7

$i=1 \quad l=2 \quad r=3$

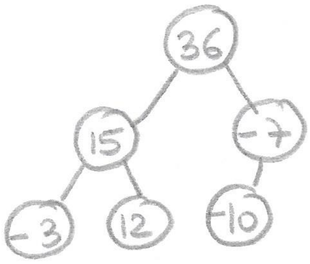
$max=3 \quad 1 \neq 3 \rightarrow$ SWAP

36	15	-10	-3	12	-7

36	15	-7	-3	12	-10

$i=3 \quad l=6 \quad r=7$

$max=6 \quad 3 \neq 6$



2) $T(n) = 9T\left(\frac{n}{3}\right) + n^2$

$a=9 \quad b=3 \quad \log_3 9 = 2 \rightarrow n^2$

$f(n) = n^2 \quad f(n) \approx g(n)$

$f(n) = \Theta(n^d) = \Theta(n^2)$

$T(n) = \Theta(n^d \log n) = \Theta(n^2 \log n)$

PROBLEMA DEL CICLO HAMILTONIANO $\rightarrow$ PROBLEMA NP

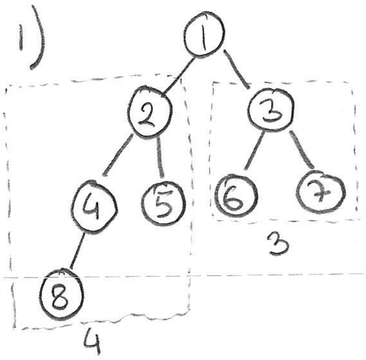
Se avesse trovato davvero questo algoritmo che risolve il problema del ciclo Hamiltoniano avremmo che $P \in P$ perche' $T(n)$ risulta polinomialmente. Ma essendo che e' stato dimostrato che Hamilton $\in$ NP, avremmo NP=P perche' Hamilton si troverebbe in entrambi gli insiemi. Cosa non verificata

ESAME 12/09/19

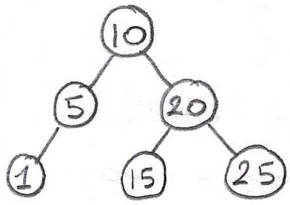
3) Si, perché per ogni vertice vengono presi in considerazione gli archi leggeri che lo unisce agli altri vertici.
Volendo simulare un algoritmo che sappiamo corretto, restituendo un MST (Kruscal o Prim), il MST risultante sarà equivalente a quello presentato.

PART II

1)



left (ricor.)
right (ricor.)
cond left + cond right
stampa



OUT: 10

```
void stampaNodi (Node u) {
    int sum = 0;
    stampaAux (u, sum);
}

int stampaAux (Node u, int& sum) {
    if (!u)
        sum=0;
    if (!u->left && !u->right)
        sum += 1;
    else {
        int l=0;
        int r=0;
        stampaAux (u->left, l);
        stampaAux (u->right, r);
        if ((l % 2 == 0) && (r % 2 != 0))
            std:: cout << u->key;
        sum = l+r+1;
    }
    return sum;
}
```

$$T(n) = \begin{cases} c & \text{se } 0 \le n \le 1 \\ T(k) + T(n-k) + d & \text{se } n > 1 \end{cases} = \Theta(n)$$

Perché l'algoritmo scorre sempre una sola volta TUTTI i nodi dell'albero.

2) mergeVect $(A_1 \ldots A_k, K, m)$

```
/* Creo B[m] */
/* Creo C[k] */
For (j=1 to k)
    C[j] = 1
index ← 0
i ← 1
j ← 1
while (j ≤ m) {
    max = -∞
    while i ≤ k {
        if (max < A_i [C[i]]) {
            max ← A_i [C[i]]
            index ← i
        }
        i++.
    }
    C[index]++
    B[j] ← max
    j++
}
return B
}
```

k

m

$T(m) = \Theta(m \cdot k)$

3) BELLMAN-FORD

$G(V,E)$, $w: E \to \mathbb{R}$

TRUE → se non ci sono cicli negativi $d[v] = d[u] + w(u,v)$

FALSE → se ci sono cicli negativi $(x_0 = x_q)$ $\sum < 0$

$$x_1 + \ldots + x_q = x_0 + \ldots + x_{q-1} + \sum w(x_{i-1}, x_i)$$

$$0 \leq \sum w(x_{i-1}, x_i)$$

ESAME 12/09/19

4) Floyd-Warshall (W)

```
m = rows(W)
D^0 = W
for k=1 to m
    for i=1 to m
        for j=1 to m
            d_ij^k = min(d_ik^(k-1) + d_kj^(k-1), d_ij^(k-1))
return D^m
```

$T(m) = \Theta(m^3)$

CORRETTEZZA

Sia $\delta(i,j) = \min w(p),\ p \in D_{ij}$

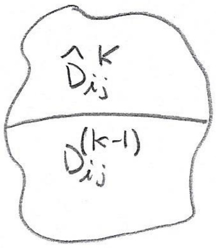
$d_{ij}^{k} = \min w(p),\ p \in D_{ij}^{k}$

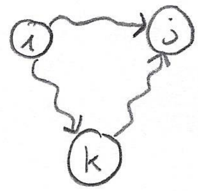
Dato k vertice $\{1 \dots m\}$, quando $k = |V| : d_{ij}^{m} = \delta(i,j)$

Per trovare il cammino minimo, posso trovare p che passano per k e p che non passano per k:

$\hat{D}_{ij}^{k} = \{ p \in \hat{D}_{ij}^{k} \mid p \text{ passa per } k \}$

$D_{ij}^{(k-1)} = \{ p \in D_{ij}^{k-1} \mid p \text{ non passa per } k \}$





$\Downarrow$

$D_{ij}^{k} = D_{ij}^{k-1} \cup \hat{D}_{ij}^{k}$ quindi:

$$d_{ij}^{(k)} = \min w(p),\ p \in D_{ij}^{(k)}$$

$$= \min \begin{cases} \min w(p),\ p \in \hat{D}_{ij}^{(k)} \\ \min w(p),\ p \in D_{ij}^{(k-1)} \end{cases} = \min \begin{cases} \min w(p),\ p \in \hat{D}_{ij}^{(k)} \\ d_{ij}^{(k-1)} \end{cases}$$

Posso separare il cammino che passa per k in (i,k) e (k,j):

$$\min w(p),\ p \in \hat{D}_{ij}^{(k)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$$

Posso ignorare $d_{kk}^{(k)}$ perché sono tutti cammini semplici:

$$d_{ij}^{k} = \min\{ d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \} \Rightarrow \begin{cases} w_{ij} & \text{se } k=0 \\ \min\{ d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \} & \text{se } k \geq 1 \\ \delta(i,j) & \text{se } k=m \end{cases}$$

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	∞	∞	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

D^0

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	∞	1	1	0	∞
5	∞	4	∞	2	0

D^1

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	4	1	1	0	∞
5	7	4	13	2	0

D^2

	1	2	3	4	5
1	0	∞	6	3	∞
2	3	0	9	6	∞
3	∞	∞	0	2	∞
4	4	1	1	0	∞
5	7	4	13	2	0

D^3

	1	2	3	4	5
1	0	4	4	3	∞
2	3	0	7	6	∞
3	6	3	0	2	∞
4	4	1	1	0	∞
5	6	3	3	2	0

D^4

	1	2	3	4	5
1	0	4	4	3	∞
2	3	0	7	6	∞
3	6	3	0	2	∞
4	4	1	1	0	∞
5	6	3	3	2	0

D^5