03MNO Algoritmi e Programmazione

Appello del 21/06/2018 - Prova di Teoria (12 punti)

1. (1 punto)

Sia data la seguente sequenza di coppie, dove la relazione i-j indica che il nodo i è adiacente al nodo j: 2-7, 5-3, 1-7, 6-2, 5-9, 5-6, 10-9, 3-5, 6-8, 10-0. Si applichi un algoritmo di on-line connectivity con **weighted** quickunion, riportando a ogni passo il contenuto del vettore e la foresta di alberi al passo finale. I nodi sono denominati con interi tra 0 e 10.

2. (1 punto)

Si ordini in maniera ascendente mediante **bottom-up** merge-sort il seguente vettore di interi:

21 19 2 14 43 3 79 23 29 17 51 10 15 16 8 101

Si indichino i passaggi rilevanti.

3. (2 punti)

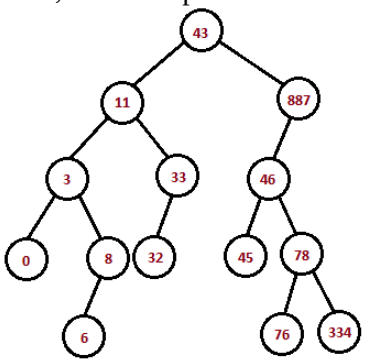
Sia data una coda a priorità implementata mediante uno heap di dati. I dati siano formati da una coppia di interi dove il secondo rappresenta la priorità. Nella radice dello heap si trovi il dato a priorità **minima**. Si inserisca la seguente sequenza di dati nella coda a priorità supposta inizialmente vuota:

(1,20) (4,32) (5,19) (3,51) (7, 28) (8,74) (9,9) (0,81) (10,17) (6,41) (2,37)

Si riporti la configurazione della coda a priorità dopo ogni inserzione. Al termine si cambi la priorità del dato in posizione 4 in 11 e si disegni la configurazione risultante della coda a priorità.

4. (1.5 punti)

Si visiti il seguente albero binario in preorder, inorder e postorder:

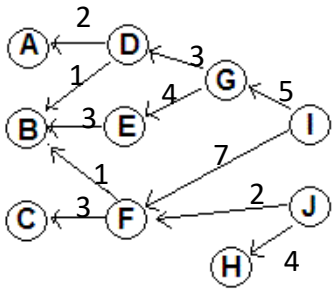


5. (2 punti)

Sia data la sequenza di chiavi intere: 12 145 267 31 98 100 4 207 13 99 181. Si riporti il contenuto di una tabella di hash di dimensione 19, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si usi l'open addressing con double hashing, definendo le opportune funzioni di hash.

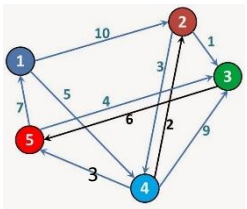
6. (2.5 punti)

Sia dato il seguente DAG pesato: considerando **I** come vertice di partenza, si determinino i cammini **massimi** tra **I** e tutti gli altri vertici. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.



7. (2 punti)

Sia dato il seguente grafo orientato pesato:



Si determinino i valori di tutti i cammini minimi che collegano il vertice **1** con ogni altro vertice mediante l'algoritmo di Dijkstra. Si assuma, qualora necessario, un ordine numerico per i vertici e gli archi.

03MNO Algoritmi e Programmazione

Appello del 21/06/2018 - Prova di programmazione (12 punti)

1. (**2 punti**)

Sia data una stringa *S* di caratteri. Si scriva una funzione C che conti quante sottostringhe di *n* caratteri in essa contenute hanno 2 vocali.

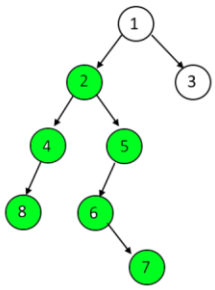
Esempio: se S=”forExample” e n= 4, le sottostringhe di lunghezza 4 con 2 vocali sono 4 e sono “forE”, “orEx”, “rExa” e “Exam”.

2. (**4 punti**)

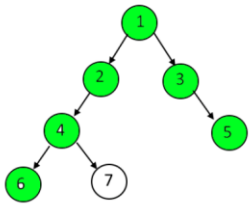
Il diametro di un albero binario è definito come la lunghezza del cammino più lungo tra una qualsiasi coppia di nodi. Si scriva una funzione C che, dato il puntatore alla radice dell’albero, ne determini il diametro:

```
int diameter(link root);
```

Esempi:



diametro = 5



diametro = 5

Si noti che il problema può essere ricondotto alla ricerca di un nodo per il quale sia massima la somma delle altezze del sotto-albero sinistro e di quello destro. È concesso estendere la struct del nodo per includere opportune informazioni aggiuntive utili.

3. **(6 punti**)

Sia data una sequenza di N interi memorizzata in un vettore $X = (x_0, x_1, \ldots x_{N-1})$. Si definisce **sottosequenza** di X di lunghezza k ($k \leq N$) un qualsiasi sottovettore Y degli elementi di X con indici crescenti $i_0, i_1, \cdots, i_{k-1}$ non necessariamente contigui.

Esempio: se X = 0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15, Y = 0, 8, 12, 10, 14, 1, 7, 15 è una sottosequenza di lunghezza k=8 di X.

Si scriva in C una funzione che, ricevuti come parametri il vettore X e la sua lunghezza N, calcoli una qualsiasi delle sottosequenze **STRETTAMENTE CRESCENTI** di X a lunghezza massima e restituisca come risultato la sua lunghezza k:

```
int maxSubSeq(int *X, int N, int *Y);
```

Esempio: se X = 0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15, le sottosequenze strettamente crescenti Y di X a lunghezza massima (k = 6) sono:

0, 2, 6, 9, 11, 15
0, 4, 6, 9, 11, 15
0, 2, 6, 9, 13, 15
0, 4, 6, 9, 13, 15

PER ENTRAMBE LE PROVE DI PROGRAMMAZIONE (18 o 12 punti):

- ***indicare nell’elaborato e nella relazione nome, cognome e numero di matricola.***
- ***se non indicato diversamente, è consentito utilizzare chiamate a funzioni standard, quali ordinamento per vettori, funzioni su FIFO, LIFO, liste, BST, tabelle di hash, grafi e altre strutture dati, considerate come librerie esterne.***
- ***gli header file devono essere allegati all’elaborato (il loro contenuto riportato nell’elaborato stesso). Le funzioni richiamate, inoltre, dovranno essere incluse nella versione del programma allegata alla relazione. I modelli delle funzioni ricorsive non sono considerati funzioni standard.***
- ***consegna delle relazioni (per entrambe le tipologie di prova di programmazione): entro domenica 24/06/2018, alle ore 23:59, mediante caricamento su Portale. Le istruzioni per il caricamento sono pubblicate sul Portale nella sezione Materiale). QUALORA IL CODICE CARICATO CON LA RELAZIONE NON COMPILI CORRETTAMENTE, VERRÀ APPLICATA UNA PENALIZZAZIONE. Si ricorda che la valutazione del compito viene fatta, senza la presenza del candidato, sulla base dell’elaborato svolto in aula. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

03MNO Algoritmi e Programmazione

Appello del 21/06/2018 - Prova di programmazione (18 punti)

1. (**18 punti**)

Un gioco si svolge su una scacchiera rettangolare di `RxC` caselle. In ogni casella deve essere posizionata una tessera su cui sono disegnati due segmenti di tubo, uno in orizzontale e uno in verticale. Le diagonali non sono contemplate. Ogni segmento è caratterizzato da un colore e da un punteggio (valore intero positivo).

Per ottenere i punti associati ai vari segmenti, è necessario allineare lungo un'intera riga (colonna) tubi in orizzontale (verticale) dello stesso colore. Le tessere possono essere ruotate di 90°. Le tessere sono disponibili in copia singola. Si assuma esistano abbastanza tessere per completare la scacchiera.

Lo scopo del gioco è ottenere il massimo punteggio possibile posizionando una tessera in ogni cella della scacchiera.

Su un primo file di testo (`tiles.txt`) è riportato l'elenco delle tessere disponibili nel seguente formato:

- sulla prima riga è presente il numero `T` di tessere
- seguono `T` quadruple nella forma `<coloreT1><valoreT1> <coloreT2> <valoreT2>` a descrivere la coppia di tubi presenti sulla tessera in termini di rispettivo colore e valore.

Si assuma che a ogni tessera sia associato un identificativo numerico nell'intervallo 0..`T`-1

Su un secondo file di testo (`board.txt`) è riportata una configurazione iniziale per la scacchiera di gioco. Il file ha il seguente formato:

- sulla prima riga è presente una coppia di interi `R C` a rappresentare il numero di righe e colonne della superficie di gioco
- seguono `R` righe riportanti `C` elementi ciascuna a definire la configurazione di ogni cella
- ogni cella è descritta da una coppia t_i`/r` dove t_i è l'indice di una tessera tra quelle presenti in `tiles.txt` e `r` rappresenta l'eventuale sua rotazione (es: 7/0 oppure 3/1 per rappresentare rispettivamente una tessera non ruotata e una ruotata). Una cella vuota è rappresentata come -1/-1.

Si scriva un programma in C che, una volta acquisiti in opportune strutture dati l'elenco delle tessere disponibili e la configurazione iniziale della scacchiera:

- legga da file due possibili completamenti per la scacchiera letta in precedenza e valuti quale dei due porti a un punteggio maggiore. I file siano caratterizzati da un numero (non precisato, ma compatibile con le celle libere della scacchiera) di righe nella forma `<`t_i`>/<rotazione> <posR> <posC>` a indicare quale tessera e con quale orientamento vada posizionata in una data cella [`posR`][`posC`] della scacchiera.
- generi la soluzione a punteggio massimo possibili a partire dalla configurazione iniziale letta da file.

Esempio:

`tiles.txt`	`board.txt`	scacchiera iniziale	scacchiera finale

`9`
`A 3 B 2`
`A 2 V 1`
`A 2 V 2`
`B 1 N 2`
`A 3 G 3`
`V 1 G 2`
`R 1 G 6`
`V 1 B 1`
`V 11 B 3`
	`3 3`
	`0/0 -1/-1 2/0`
	`3/1 -1/-1 -1/-1`
	`-1/-1 6/0 -1/-1`