03MNO Algoritmi e Programmazione

Appello del 13/09/2018 - 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 quickfind, 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. (2 punti)

Sia data la sequenza di interi, supposta memorizzata in un vettore:

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

- la si trasformi in un heap, ipotizzando di usare un vettore come struttura dati. Si riportino graficamente i diversi passi della costruzione dell'heap ed il risultato finale. Si ipotizzi che, alla fine, nella radice dell'heap sia memorizzato il valore massimo
- si eseguano su tale heap i primi 2 passi dell'algoritmo di heapsort.

NB: la sequenza è già memorizzata nel vettore e rappresenta una configurazione intermedia per cui la proprietà di heap non è ancora soddisfatta.

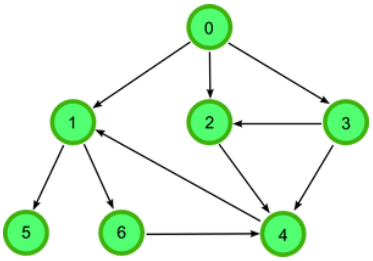
3. (2 punti)

Si effettuino, secondo l'ordine specificato, le seguenti inserzioni in foglia su un Interval BST supposto inizialmente vuoto:

[2,12] [3, 6] [0,5] [15,21] [8,13] [1,8] [11,23] [4,15] [7,12]

4. (2 + 1.5 + 1.5 punti)

Sia dato il seguente grafo orientato:

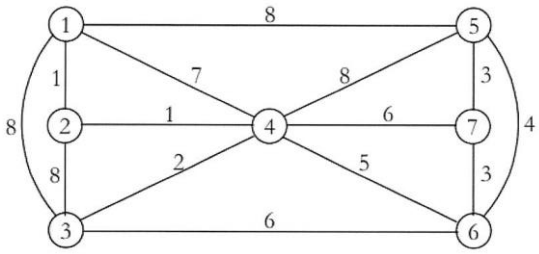


- se ne effettui una visita in profondità, considerando **0** come vertice di partenza (**2 punti**)
- lo si ridisegni, etichettando ogni suo arco come T (tree), B (back), F (forward), C (cross), considerando **0** come vertice di partenza (**1.5 punti**)
- se ne effettui una visita in ampiezza, considerando **0** come vertice di partenza (**1.5 punti**)

Qualora necessario, si trattino i vertici secondo l'ordine numerico.

5. (2 punti)

Sia dato il seguente grafo pesato (lo si consideri non orientato trascurando i versi degli archi):



se ne determini un minimum spanning tree applicando l'algoritmo di Prim a partire dal vertice **1**, disegnando l'albero e ritornando come risultato il valore del peso minimo. Si esplicitino i passi intermedi.

03MNO Algoritmi e Programmazione

Appello del 13/09/2018 - Prova di programmazione (12 punti)

1. (**2 punti**)

In un campionato *n* squadre giocano per *m* giornate. Sia data una matrice *C* di $n \times m$ numeri interi, ognuno dei quali può valere soltanto 0, 1 o 3. Ogni riga della matrice rappresenta i punti acquisiti dalle *n* squadre nelle partite disputate nelle *m* giornate del campionato: 3 punti per le partite vinte, 1 punto per quelle pareggiate e 0 punti per le sconfitte. I risultati della giornata *k*-esima sono contenuti nelle righe della colonna di indice *k*. Si scriva una funzione C con il seguente prototipo

```
void displRanking(int **C, int n, int m);
```

che, per ogni giornata del campionato, stampi l'indice (il numero di riga corrispondente) della squadra capolista in quella giornata.

Esempio:

Con la seguente matrice di 4 righe e 3 colonne, l'output sarà:

3	1	0
0	1	1
1	1	1
1	1	3

0	0	3

2. (**4 punti**)

Sia data una lista di interi, cui si accede mediante puntatore alla testa `link1 head`. Si scriva una funzione C che costruisca una nuova lista "compressa": per ogni elemento della prima lista si memorizza nella seconda lista l'elemento stesso e il numero di ripetizioni consecutive nella prima. Si definisca opportunamente il nodo della seconda lista. Il prototipo sia:

```
link2 comprimi(link1 head);
```

Esempio: se la prima lista è (3, 3, 3, 3, 2, 2, 3, 5, 5, 5), la seconda lista sarà ((3, 4), (2, 2), (3, 1), (5, 3)).

3. (6 punti)

Un vettore `paste` di *n* `struct` descrive le tipologie e le disponibilità dei pasticcini realizzati da un laboratorio di pasticceria. Ogni `struct` consta di 3 campi interi non negativi:

- `codice` identifica la tipologia di pasticcino (ad esempio 3 bigné, 5 crostatina, etc.)
- `peso` indica il peso in grammi di ogni pasticcino
- `quantita` indica il numero di pasticcini di quella tipologia disponibili.

Si scriva un programma C che, ricevuti come parametro il vettore, la sua lunghezza e un intero che rappresenta il peso del vassoio di pasticcini da comporre, calcoli l'insieme di pasticcini il cui peso si avvicini il più possibile (al più eguagli) al peso richiesto, tenendo conto delle disponibilità di ogni tipologia. Si proceda quindi a stampare tale insieme, specificando per ogni tipologia di pasticcini la sua cardinalità, ed il peso finale del vassoio. Nel caso di più soluzioni equivalenti è sufficiente stamparne una. Si supponga inoltre che l'utente non abbia alcuna preferenza sulla tipologia di pasticcini selezionati.

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 16/09/2018, 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 13/09/2018 - Prova di programmazione (18 punti)

1. (**18 punti**)

I titoli azionari delle aziende quotate in Borsa possono essere oggetto di transazioni (vendita/acquisto di un certo numero di titoli in una certa data/ora ad un certo valore su una certa piazza) in una o più piazze finanziarie in tutto il mondo. Si immagini di voler creare un sistema globale, cioè che tenga conto di tutte le Borse, per la memorizzazione e gestione dei titoli azionari.

Ogni Borsa invia al sistema con regolarità un file con un elenco di transazioni raggruppate per titolo. Sulla prima riga compare il numero di titoli, di seguito i titoli e le relative transazioni con il seguente formato:

- una prima riga contiene il `<titolo>` (codice alfanumerico univoco di al più 20 caratteri) seguito dal numero di transazioni
- sulle righe successive, una per riga, le transazioni sotto forma di quaterne `<data> <ora> <valore> <numero>`. Le date sono nella forma `aaaa/mm/gg`, le ore sono nel formato su 24 ore `hh:mm` riferite al tempo di Greenwich (GMT), mentre i valori dei titoli sono rappresentati con numeri reali non negativi, la quantità è un intero. Non si presupponga nessuna forma di ordinamento né sui nomi dei titoli, né sulle transazioni.

Il sistema acquisisce i file e ne memorizza i contenuti in un'opportuna struttura dati a 2 livelli che prevede una collezione di titoli e per ogni titolo una collezione delle sue quotazioni giornaliere. La quotazione giornaliera $\mathbf{Q_i}$ di un titolo in una certa data **i** è la media di tutti i valori $\mathbf{v_{ij}}$ di quel titolo in quella data **i** pesati sul numero di titoli scambiati $\mathbf{n_{ij}}$

$$\boldsymbol{Q_i = \frac{\sum_j v_{ij} \cdot n_{ij}}{\sum_j n_{ij}}}$$

Si realizzino:

- un quasi ADT per la data, implementato come struct con campi interi per anno, mese e giorno
- un ADT di I classe per titolo e collezione di titoli
- un ADT di I classe per quotazioni giornaliere e collezione di quotazioni giornaliere
- un client che fornisca le seguenti funzionalità (l'eventuale menu e il tipo di I/O sono a scelta dello studente):
  1. acquisizione in qualsiasi momento del contenuto di un file contenente un insieme di transazioni (*si noti che la possibilità di acquisire un nuovo file in qualsiasi momento richiede una struttura dati completamente dinamica*)
  2. ricerca di un titolo azionario, con complessità al più logaritmica nel numero di titoli
  3. ricerca, dato un titolo precedentemente selezionato, della sua quotazione in una certa data, con complessità al più logaritmica nel numero di quotazioni per quel titolo
  4. ricerca, dato un titolo precedentemente selezionato, della sua quotazione minima e massima in un certo intervallo di date.
  5. ricerca, dato un titolo precedentemente selezionato, della quotazione minima e massima lungo tutto il periodo registrato.

La complessità dell'operazione nei punti 4 e 5 non è vincolata, ma sarà usata come metro di giudizio. Si richiede di indicare esplicitamente la complessità della soluzione sviluppata. Non è richiesta un'implementazione completa degli ADT. Per le collezioni si realizzino, oltre alla definizione della struttura dati, le funzioni di inserimento e ricerca richieste dai punti precedenti. Per l'ADT data si realizzino le funzioni di lettura, scrittura e confronto. Nel caso si utilizzino BST non si ponga il problema di bilanciarli.