03MNO Algoritmi e Programmazione

Appello del 21/02/2020 - 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: 2-6, 5-2, 0-7, 4-3, 3-8, 3-7, 9-8, 2-5, 5-8, 8-0. 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 9.

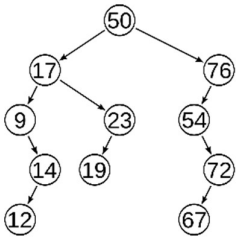
2. (2.5 punti)

Si risolva la seguente equazione alle ricorrenze mediante il metodo dello sviluppo (unfolding):

$$T(n) = 3T(n/3) + n/3 \qquad n>1$$
$$T(1) = 1 \qquad n=1$$

3. (2 punti)

Si inseriscano in **radice** nel BST di figura in sequenza le chiavi 6, 29 e 22 e poi si cancelli la chiave 19. Si disegni l'albero ai passi significativi.



4. (2 punti)

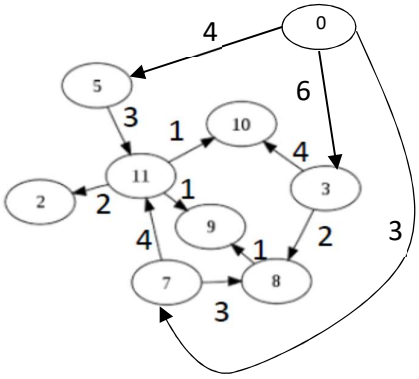
Sia data una coda a priorità implementata mediante uno heap di dati. I dati siano formati da una coppia di stringa-intero dove il secondo rappresenta la priorità. Nella radice dello heap si trovi il dato a priorità **massima**. Si inserisca la seguente sequenza di dati nella coda a priorità supposta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

(ab,24) (cfd,62) (FF,39) (aAd,51) (rt, 81) (PUy,74) (S,19) (ttt,18) (mn,27) (qwr,61) (ws,59)

Si riporti la configurazione della coda a priorità dopo ogni inserzione. Al termine si cambi la priorità del dato che si trova nello heap all'indice 4 in 11 e si disegni la configurazione risultante della coda a priorità.

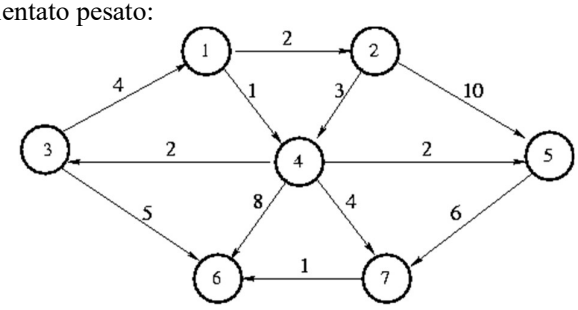
5. (2.5 punti)

Sia dato il seguente DAG pesato: considerando **0** come vertice di partenza, si determinino i cammini **massimi** tra **0** e tutti gli altri vertici. Qualora necessario, si trattino i vertici secondo l'ordine numerico si assuma che la lista delle adiacenze sia anch'essa ordinata numericamente.



6. (2 punti)

Sia dato il seguente grafo orientato pesato:



Si determinino i valori di tutti i cammini minimi che collegano il vertice **3** con ogni altro vertice mediante l'algoritmo di Dijkstra.

3MNO Algoritmi e Programmazione

Appello del 21/02/2020 - Prova di programmazione (18 punti)

1. (**18 punti**)

Un ristorante offre una selezione di piatti tra cui è possibile scegliere, riportati in un file testuale `piatti.txt`. Nella prima riga compare il numero `N` di piatti. Su ognuna delle `N` righe successive compare la descrizione di ogni piatto come quaterna `<nome> <portata> <tipologia> <costo>`. I primi tre campi sono stringhe senza spazi di massimo 100 caratteri. Il quarto campo è un numero positivo con al massimo 2 cifre decimali.

Esempio di `piatti.txt`:

10

Carbonara	Primo	Pasta	8.50
Amatriciana	Primo	Pasta	7.80
Caprese	Contorno	Vegetariano	5.50
VerdureGrigliate	Contorno	Vegetariano	4
Nizzarda	Contorno	NonVegetariano	8.25
FilettoAllaWellington	Secondo	Carne	17
Opera	Dessert	Dolce	4.50
Tiramisu	Dessert	Dolce	4.00
Cotoletta	Secondo	Carne	6.50
VitelloTonnato	Antipasto	NonVegetariano	4.00

Il ristorante vuole generare l'elenco di tutti i possibili menu di `P` piatti, dove `P` è un numero fisso. Per menu si intende un elenco di `P` piatti in cui:

- l'ordine dei piatti non conta. In altri termini tutte le permutazioni di piatti dello stesso menu sono identiche e basta considerarne una sola
- un piatto può eventualmente comparire due volte (per indicare un "bis").

Si scriva un programma che, ricevuto come argomento sulla riga di comando il valore `P`:

- acquisisca dal file `piatti.txt` l'elenco dei piatti disponibili, caricandoli in un vettore `ElencoPiatti`. I piatti sono ordinati secondo l'ordine con cui compaiono nel vettore. Un piatto x è minore di un piatto y se x compare prima di y nel file e quindi nel vettore
- utilizzi un quasi ADT per il menu, contenente l'elenco dei `P` piatti del menu (interi, indici nel vettore di cui al punto precedente) e il prezzo complessivo del menu. I piatti vanno ordinati in base al criterio di cui sopra
- contenga una funzione di confronto tra menu `MENUcompare` che, dati due menu, li confronti in base al prezzo e, a parità di prezzo, in base all'elenco dei piatti: un menu A precede un menu B se il prezzo di A è minore del prezzo di B. A parità di prezzo si confrontano per ognuno dei due menu i piatti che compaiono per primi in essi. Se persiste la parità si procede con la seconda coppia di piatti e così via
- generi, mediante funzione ricorsiva basata su un opportuno modello del calcolo combinatorio, l'elenco di tutti i possibili menu, immagazzinandoli in un BST, ordinato secondo la funzione `MENUcompare`
- stampi i menu disponibili (secondo l'ordine crescente individuato dalla funzione `MENUcompare`), mediante una visita del BST. Per ogni menu si stampino le informazioni complete (quaterna) di ogni piatto.

Nota: le funzioni di creazioni e di inserimento nel BST possono essere considerate di libreria. E' richiesta la funzione di visita con stampa dei dati.

Esempio: se `P`=5, i due menu A e B seguenti presentano bis di piatto, i piatti sono ordinati secondo l'ordine del file, sono a pari prezzo (25,00) il primo precede il secondo nella `MENUcompare` in quanto il piatto `Tiramisu` precede il piatto `VitelloTonnato` in `piatti.txt` (i piatti precedenti sono identici):

Menu A	**Menu A**
Carbonara Primo Pasta 8.50	Carbonara Primo Pasta 8.50
VerdureGrigliate Contorno Vegetariano 4.00	VerdureGrigliate Contorno Vegetariano 4.00
VerdureGrigliate Contorno Vegetariano 4.00	VerdureGrigliate Contorno Vegetariano 4.00
Opera Dessert Dolce 4.50	Opera Dessert Dolce 4.50
Tiramisu Dessert Dolce 4.00	VitelloTonnato Antipasto NonVegetariano 4.00

03MNO Algoritmi e Programmazione

Appello del 21/02/2020 - Prova di programmazione (12 punti)

1. (**2 punti**)
Sia data una matrice di caratteri `A` di dimensioni `n` × `m` ed una stringa. Si scriva una funzione C che conti quante volte la stringa è contenuta nelle righe e nelle colonne della matrice. La stringa va considerata in orizzontale da sinistra a destra o in verticale dall'alto al basso. Il prototipo della funzione sia:

```
int conta(char **matrice, int n, int m, char *stringa);
```

Esempio: la stringa `cat` è contenuta 3 volte nella seguente matrice `A` di `n` = 4 righe e `m` = 5 colonne (nella seconda colonna, nella quarta colonna e nella quarta riga):

$$A = \begin{pmatrix} x & c & e & c & a \\ w & a & e & c & q \\ d & t & p & a & z \\ p & c & a & t & f \end{pmatrix}$$

2. (**4 punti**)
Sia data una lista concatenata `L`. Sia dato un intero `k`. Si scriva una funzione C che scambi il `k`-esimo nodo dalla testa della lista con il `k`-esimo nodo dalla coda della lista

```
void swap(list L, int k);
```

Oltre che l'implementazione della funzione `swap`, si richiede anche l'esplicita definizione del tipo `list` e dei nodi usati all'interno delle liste. La lista sia un ADT di I classe. **Ai fini dell'esercizio, non si può fare uso di funzioni di libreria.** Negli scambi è lecito o scambiare i valori o scambiare i nodi

Esempio: se la lista contiene 10, 20, 30, 40, 50, 60, 70 e k=0 oppure k=6, dopo lo scambio conterrà **70**, 20, 30, 40, 50, 60, **10**. Se la lista contiene 10, 20, 30, 40, 50, 60, 70 e k=2, dopo lo scambio conterrà 10, 20, **50**, 40, **30**, 60, 70. Per k > 6 lo scambio non avviene.

3. **(6 punti**)
Sia data un insieme S = $\{S_0, S_1, \ldots, S_{n-1}\}$ di stringhe di lettere maiuscole. Si determini mediante un algoritmo ricorsivo il numero massimo di stringhe appartenenti ad S mutuamente disgiunte, cioè che non presentano nessuna lettera in comune.

Esempio: dato un insieme S che contiene le seguenti 7 stringhe
S_0 = ABGCIEF, S_1 = BA, S_2 = CD, S_3 = FE, S_4 = GHBD, S_5 = JKLGHI, S_6 = FK

il numero massimo di stringhe mutuamente disgiunte è 4. Anche se le specifiche non richiedono che esse siano elencate, in quanto basta il loro numero, per una migliore comprensione si osservi che esse sono S_1 = BA, S_2 = CD, S_3 = FE, S_5 = JKLGHI.

La funzione abbia come prototipo:

```
int disgiunte(char **stringhe, int n);
```

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 lunedì 24/02/2020, alle ore 14:00, 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.***