03MNO Algoritmi e Programmazione

Appello del 22/02/2019 - Prova di teoria (12 punti)

1. (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 **minima**. Si inserisca la seguente sequenza di dati nella coda a priorità supposta inizialmente vuota:

(ab,20) (cfd,32) (FF,19) (aAd,51) (rt, 28) (PUy,74) (S,9) (ttt,81) (mn,17) (qwr,41) (ws,37)

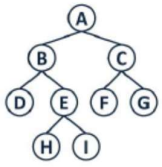
Si riporti la configurazione della coda a priorità dopo ogni inserzione. Al termine si cambi la priorità del dato in posizione 4 in 11 e si disegni la configurazione risultante della coda a priorità.

2. (2.5 punti)

Data la catena di matrici (A_1, A_2, A_3, A_4) di dimensioni (3x7), (7x4), (4x2) e (2x5) rispettivamente, si determini mediante un algoritmo di programmazione dinamica la parentesizzazione ottima del prodotto di matrici che minimizza il numero di moltiplicazioni. Si indichino i passaggi.

3. (3 x 0.5 punti)

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

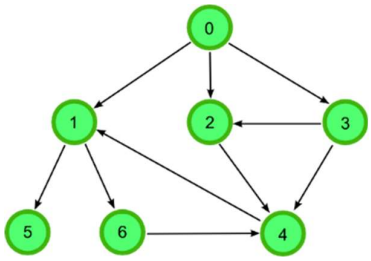


4. (2 punti)

Sia data la sequenza di chiavi intere: 31 124 167 102 98 10 75 107 130 99 17. 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. Si indichino i passaggi.

5. (2 punti)

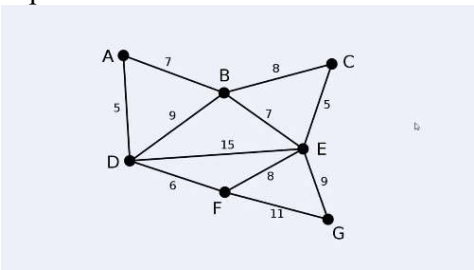
Sia dato il seguente grafo orientato:



se ne effettui una visita in profondità, considerando **0** come vertice di partenza etichettando i vertici con tempo di scoperta/tempo di fine elaborazione e gli archi con T, B, F, C. Qualora necessario, si trattino i vertici secondo l'ordine numerico.Si indichino i passaggi.

6. (2punti)

Si determini mediante l'algoritmo di Kruskal l'albero ricoprente minimo per il grafo non orientato, pesato e connesso in figura illustrando i passaggi intermedi del procedimento adottato. Si disegni l'albero e si ritorni come risultato il valore del peso minimo.



03MNO Algoritmi e Programmazione

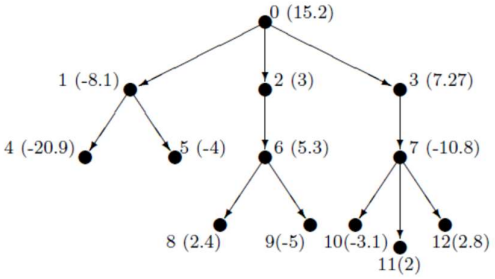
Appello del 22/02/2019 - Prova di programmazione (12 punti)

1. (**2 punti**)
Sia dato un vettore *V* di *N* interi. Si scriva una funzione C che visualizzi un qualsiasi sottovettore proprio di celle contigue la somma dei cui valori sia massima. Un sottovettore si dice proprio se esso contiene un numero di celle strettamente inferiore a quello del vettore da cui deriva.
Esempio: se V= -1, 2, 3, -6, 1, 3, 1 i 2 sottovettori a somma massima 5 sono 2,3 e 1,3,1 entrambi ammissibili come soluzione.

2. (**4 punti**)
Sia dato un albero T di grado `k`a cui si accede tramite il puntatore `root` alla radice. I nodi dell'albero constano di un identificatore intero `i` e di un peso `float` qualsiasi `wi` e dei puntatori di tipo `link` ai sottoalberi. Il peso di un sottoalbero è definito come la somma dei pesi dei suoi nodi. Si definisca opportunamente il tipo `nodo_t` e si scriva una funzione C con il seguente prototipo che ritorni l'identificatore della radice del sottoalbero di peso massimo ed il peso massimo:

```
int maxSum(link root, float *maxwt);
```

Esempio: dato l'albero T di grado `k`=3 in figura, il sottoalbero di peso massimo è radicato in 2 ed ha peso 3 + 5.3 + 2.4 − 5 = 5.7:



3. (**6 punti**)
Sia dato un vettore `V` di `N` interi. Si definisce **sottosequenza** di `V` di lunghezza `k` (`k`≤`N`) un qualsiasi n-upla `Y` di `k` elementi di `V`non necessariamente contigui.Si ricordi che nella n-upla l'ordine conta.
Esempio: se `N`=9 e `V` = [8, 9, 6, 4, 5, 7, 3, 2, 4], una sottosequenza con `k`=4 è `Y` = [9, 6, 7, 2]

La sottosequenza si dice **alternante minore-maggiore** se e solo se valgono entrambe le seguenti condizioni:

- ogni elemento `Y[i]` di indice `i`(nella sottosequenza `Y`) pari è seguito, se esiste, da un elemento `Y[i+1]` di indice dispari e di valore maggiore, cioè tale che `Y[i]<Y[i+1]`
- ogni elemento `Y[i]` di indice `i` dispari è seguito, se esiste, da un elemento `Y[i+1]` di indice pari e di valore minore, cioè tale che `Y[i] > Y[i+1]`.

Esempio: la sequenza 4, 11, 3, 8, 5, 6, 2, 10, 1 è alternante minore-maggiore.

Si scriva una funzione C che, ricevuti come parametri il vettore `V` e l'intero `N` calcoli e visualizzi una qualsisasi sottosequenza alternante minore-maggioredi lunghezza massima.

Esempio: se `N`=9 e `V` = [8, 9, 6, 4, 5, 7, 3, 2, 4] una sottosequenza alternante minore-maggiore di lunghezza massima `k`=6 è `V` = [8, 9, 6, 7, 3, 4].

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 25/02/2019, alle ore 12:00, 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.***

03MNO Algoritmi e Programmazione

Appello del 22/02/2019 - Prova di programmazione (18 punti)

1. (**18 punti**)

Un'azienda impiega dipendenti che possono svolgere più di una **tipologia** di lavoro(**o**peraio, **a**mministrativo, **t**ecnico e **i**nformatico) con un certo grado di competenza. Un primo file (*dei dipendenti*) contiene le informazioni sui dipendenti:

- la prima riga riporta il numero N dei dipendenti
- le `N` righe successive contengono i dati sui dipendenti, ognuno descritto (in campi separati da spazi) da:
  - matricola (intero su 4 cifre)
  - nome e cognome (stringhe di al massimo 20 caratteri)
  - competenze (4 interi tra 0 e 100) nelle 4 tipologie (operaio, amministrativo, tecnico o informatico).

Esempio: `1234 Rossi Mario 10 60 30 80` è un dipendente molto adatto al ruolo di informatico (80 su 100), abbastanza al ruolo di amministrativo (60 su 100), poco al ruolo di tecnico (30 su 100), per nulla al ruolo di operaio (10 su 100).

L'azienda è organizzata in `D`divisioni che la Direzione Generale sta ristrutturando. Gli `N` dipendenti vanno ripartiti tra le `D`divisioni, dove svolgerà una sola delle 4 tipologie di lavoro con la relativa competenza. A tal fine, un secondo file (*delle divisioni*) contiene le richieste delle divisioni: la prima riga contiene il numero `D` di divisioni, seguono `D` gruppi di 5 righe: l'`i`-esimo gruppo, corrispondente alla divisione `i`, identificata, su una riga, dalla sigla (una stringa la massimo di 10 caratteri). Seguono 4 righe, associate ognuna a una delle 4 tipologie per le quali la divisione fa una richiesta: detta `j` la riga (la tipologia), questa contiene una terna di interi m_{ij},$cmin_{ij}$ e r_{ij}. (separati da spazi), che rappresentano, rispettivamente (per la divisione `i` e la tipologia `j`), il numero minimo di addetti, la competenza minima totale e competenza ottimale totale richieste.

Si scriva un programma C che, ricevuti sulla linea di comando i nomi di tre file:

- acquisisca in opportune strutture dati le informazioni contenute nei 2 file dei dipendenti e delle divisioni. Si richiede di utilizzare un quasi ADT (`dipendente_t`) per un dipendente e un ADT di prima classe (`divisione_t`) per una divisione. Le collezioni di dipendenti e di divisioni siano realizzate come vettori dinamici(`dipendenti` e `divisioni`) di tali strutture dati. L'ADT `divisione_t` deve contenere la collezione dei dipendenti assegnati alla divisione stessa (a scelta come collezione di item `dipendente_t` o di riferimenti a elementi del vettore `dipendenti`)e la tipologia di lavoro che ogni dipendente svolgerà. Per questo ADT si implementino esplicitamente le funzioni di *creazione, distruzione, acquisizione* della divisione*, inserimento* di un dipendente *e stampa/salvataggio* su file
- legga un terzo file (*delle associazioni*) contenente, su `N` righe, associazioni dipendente-tipologia-divisione, mediante terne `<matricola><tipologia><sigla>` che specificano a quale divisione e con quale tipologia è stato assegnato il dipendente con quella matricola, verifichi se sono soddisfatti i vincoli di numero minimo di addetti e di competenza totale minima richiesti per ogni tipologia di ogni divisione. La tipologia è identificata tramite la lettera iniziale (`o`, `a`, `t`, `i`). Si calcoli inoltre lo scostamento complessivo medio, rispetto alla competenza richiesta (somma degli scostamenti per divisione e per tipologia diviso il numero di divisioni)

$$\Delta_{avg} = \frac{\sum_{i=1}^{D}\sum_{j=1}^{4}|r_{ij} - c_{ij}|}{D}$$

  Per competenza totale c_{ij} della divisione `i` nella tipologia `j` si intende la somma delle competenze in quella tipologia dei singoli dipendenti assegnati alla divisione `i` con competenza `j`.
- calcoli un'attribuzione ottima di dipendenti alle divisioni tale da:
  - soddisfare i vincoli minimi per numero di dipendenti e competenza totale per ogni tipologia di ogni divisione
  - minimizzare lo scostamento complessivo medio Δ_{avg} (si veda definizione al punto precedente).

  Tale attribuzione sia memorizzata su un file di uscita, il cui nome è passato come parametro sulla riga di comando, nel formato una coppia matricola-sigla per ogni riga.
- dopo aver generato la soluzione ottima del punto precedente, la elabori e la memorizzi nel vettore di collezioni (`divisioni`): si ricorda che il tipo`divisione_t` prevede la possibilità di collezionare i dipendenti di una divisione. Si salvi, usando la funzione di *stampa/salvataggio* (di cui al punto 1) dell'ADT `divisione_t`, l'attribuzione ottima su un file di uscita, nello stesso formato del file delle associazioni (il terzo file, usato al punto 2).