02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 22/02/2016 - 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:

1-7, 2-1, 0-3, 4-0, 1-6, 3-4, 0-8, 1-3, 6-7, 10-3

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. (2.5 punti)

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

$T(n) = T(n-1) + n^2 \quad n>1$

$T(1) = 1 \quad n=1$

ricordando che $\sum_{i=0}^{n-1} i^2 = \frac{n(n+1)(2n+1)}{6}$

3. (1 punto)

Si ordini in maniera ascendente mediante counting-sort il seguente vettore di interi:

3 0 5 0 6 1 4 0 8 1 2 1 8 0 2

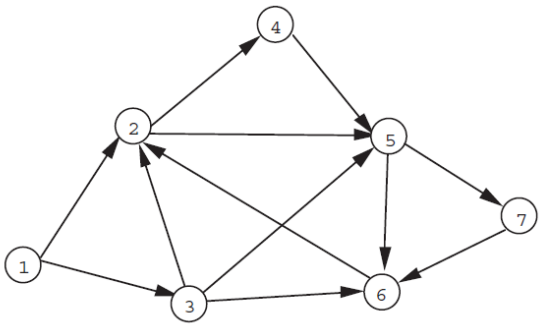
Si indichino le strutture dati usate nei passi intermedi.

4. (2 punti)

Sia data la sequenza di stringhe **I**n **B**ob **U**ntil **X**eno **E**dge **G**raph **Z**eta **H**otel **Y**ou, dove ogni stringa ha come chiave il suo primo carattere, individuato dal suo ordine progressivo nell'alfabeto (A=1, ..., Z=26). Si riporti la struttura 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.

5. (1.5 + 1.5 punti)

Sia dato il seguente grafo orientato:

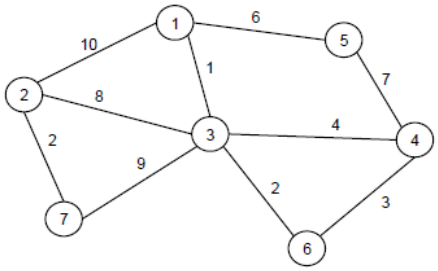


- se ne effettui una visita in ampiezza, considerando **1** come vertice di partenza (**1.5 punti**)
- lo si ridisegni, etichettando ogni suo arco come T (tree), B (back), F (forward), C (cross), considerando **1** come vertice di partenza (**1.5 punti**).

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

6. (1.5 + 0.5 + 0.5 punti)

Si determini mediante l'algoritmo di Kruskal l'albero ricoprente minimo per il grafo in figura illustrando i passaggi intermedi del procedimento adottato (**1.5 punti**).



Dopo aver determinato la soluzione, si consideri l'aggiunta di un nuovo arco (1-4) di costo 5. Questo arco è utile per migliorare il costo della soluzione? Perché (**0.5 punti**)? Si consideri anche l'arco (1-7) di costo 5. Questo arco è utile per migliorare il costo della soluzione? Perché (**0.5 punti**)?

02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 22/02/2016 - Prova di programmazione (18 punti)

Un grafo non orientato, pesato e **colorato** è memorizzato su un file mediante l'elenco dei suoi archi, con il seguente formato:

$$idV_1\ colV_1\ val\ idV_2\ colV_2$$

che indica che il vertice con identificatore idV_1 e colore $colV_1$ è connesso con un arco non orientato di peso val al vertice con identificatore idV_2 e colore $colV_2$. Gli identificatori alfanumerici sono di lunghezza massima 20 caratteri. I colori sono 2 e sono identificati dalle stringhe ROSSO e NERO. I pesi sono valori interi e positivi.

Una volta acquisito il grafo in una opportuna struttura dati, implementare le seguenti funzionalità che risolvono 2 problemi distinti:

- verifica di congruenza della colorazione dei vertici: non deve succedere che lo stesso vertice compaia con colori diversi in archi diversi
- calcolo del cammino semplice a peso massimo all'interno del grafo che soddisfi il seguente vincolo:

  nel cammino un vertice NERO può essere seguito da un vertice sia NERO che ROSSO, mentre un vertice ROSSO può solo essere seguito da un vertice NERO

  Una volta individuato tale cammino si provveda a stamparlo a video elencando i nomi dei nodi attraversati dalla sorgente alla destinazione
- identificazione del sottografo che soddisfa i seguenti vincoli :
  - è connesso
  - la somma dei pesi dei suoi archi è massima
  - il numero di vertici neri e rossi che lo compongono differisce al più di 2.

Poiché il numero di vertici del grafo non è noto a priori, si suggerisce di calcolarlo tramite lettura preliminare del file e/o caricamento in una tabella di simboli. Si può assumere che non esistano archi duplicati. Non è lecito assumere nessuna forma di ordinamento degli archi. E' richiesta la scrittura della funzione di verifica di connettività, nella quale è ammesso richiamare una funzione standard di libreria di visita in profondità.

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

- ***indicare nell'elaborato e nella relazione (oltre a nome, cognome e numero di matricola) anche il nome del corso per cui si sta sostenendo l'esame (AP, APA I+II).***
- ***Se non indicato diversamente, è consentito utilizzare chiamate a funzioni <u>standard</u>, 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):***
  - ***LAUREANDI: entro martedì 24/02/2016 ore 24:00***
  - ***NON LAUREANDI: entro giovedì 25/02/2016, alle ore 24:00***

  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 discussione o esame orale, <u>sulla base dell'elaborato svolto in aula</u>. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.

NB: per i LAUREANDI:

- ***indicare di essere laureandi sul compito scritto consegnato***
- ***gli orali si svolgeranno lunedì 29 in orario e aula che verranno pubblicati successivamente sul Portale.***

Queste modalità, concordate con la Segreteria, consentono a chi supera l'esame di fare domanda di laurea entro i tempi stabiliti.

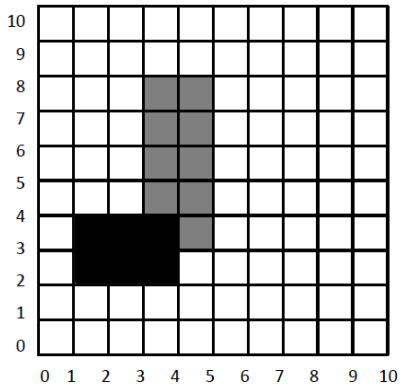
02MNO Algoritmi e Programmazione 01JKE APA I / 01JKF APA II

Appello del 22/02/2016 - Prova di programmazione (12 punti)

1. (**2 punti**)

Nel primo quadrante del piano cartesiano è individuata una regione quadrata i cui vertici in basso a sinistra e in alto a destra hanno coordinate (0,0) e (100,100). In un file è descritta una sequenza di rettangoli con i lati paralleli agli assi cartesiani. Ogni rettangolo è individuato dalle coordinate dei vertici in basso a sinistra e in alto a destra. Le coordinate sono interi compresi tra 0 e 100, estremi inclusi. Scrivere la funzione `int areaTot(FILE *fp);` che riceve come parametro un puntatore al file (già aperto) e restituisce il valore totale dell'area coperta dai rettangoli. Nel caso di intersezione di rettangoli l'area è contata una sola volta.

Esempio (semplificato con coordinate x e y nell'intervallo (0..10): se il file contiene sulla prima riga 1 2 4 4 e sulla seconda 3 3 5 8, l'area coperta vale 15.



2. (**4 punti**)

Sia data una lista non ordinata di interi e un valore intero inteso come soglia. Si scriva una funzione C che divida la lista in 2 : nella prima lista compaiono gli elementi della lista originaria minori della soglia, nella seconda quelli maggiori o uguali alla soglia. Sia mantenuto l'ordine relativo della lista originaria nelle 2 liste così create. **Si noti che NON si può far riferimento a funzioni di libreria sulle liste.** Si usi un ADT di I categoria per le liste con una `struct` wrapper di tipo `lista_t`. La funzione C sia compatibile con la chiamata effettuata dal seguente frammento del `main`:

```
lista_t *L0, *L1, *L2;
…
// acquisizione della lista L0
…
L1 = split_list(n, L0, &L2);
```

La funzione riceve come parametro la lista originale L0, ritorna la prima lista L1 e, by reference, la seconda L2.

Esempio: se la lista L0 in ingresso contiene i valori 7, 8, 25, 2, 9, -5, 10, 37 e la soglia è n=18, la prima lista L1 dovrà contenere 7, 8, 2, 9, -5, 10 e la seconda L2 25,37.

3. (**6 punti**)

Sia dato un insieme di n interruttori e un insieme di m lampadine, inizialmente tutte spente. Ogni interruttore comanda un sottoinsieme delle lampadine: se lo si preme, ogni lampadina da esso controllata commuta di stato, cioè se è accesa si spegne, se è spenta si accende. L'informazione su quale sottoinsieme di lampadine agisce ogni singolo interruttore è memorizzata in una matrice di interi di dimensioni n x m. L'elemento [i,j] della matrice vale 1 se l'interruttore i controlla la lampadina j, 0 altrimenti. Si scriva una funzione ricorsiva in C in grado di determinare l'insieme minimo di interruttori per accendere tutte le lampadine. Si noti che una lampadina è accesa se e solo se è dispari il numero di interruttori premuti che la controllano.

Esempio: se per n=4 e m=5 la matrice contiene

1	1	0	0	1
1	0	1	0	0
0	1	1	1	0
1	0	0	1	0

il minimo numero di interruttori da premere è 3 e questi interruttori sono gli interruttori 0, 1 e 3.