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

Appello del 26/01/2016 - Prova di teoria (12 punti)

1. (2 punti)
Sia data la sequenza di interi, supposta memorizzata in un vettore:

4 53 92 52 32 34 13 12 91 93 22

si eseguano i primi 2 passi dell'algoritmo di quicksort per ottenere un ordinamento ascendente, indicando ogni volta il pivot scelto. NB: i passi sono da intendersi, impropriamente, come in ampiezza sull'albero della ricorsione, non in profondità. Si chiede, pertanto, che siano ritornate le 2 partizioni del vettore originale e le due partizioni delle partizioni trovate al punto precedente.

2. (2.5 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 massima. 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 101 e si disegni la configurazione risultante della coda a priorità.

3. (2 punti)
Sia data la sequenza di chiavi "alpha", "beta", "delta", "epsilon", "zeta", "eta", "theta", "iota", "kappa". Si supponga che h("alpha")=2, h("beta")=14, h("delta")=18, h("epsilon")=17, h("zeta")=17, h("eta")=16, h("theta")=18, h("iota")=2, h("kappa")=14. Si riporti la struttura di una tabella di hash di dimensione 19, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza indicata. Si supponga di utilizzare l'open addressing con linear probing.

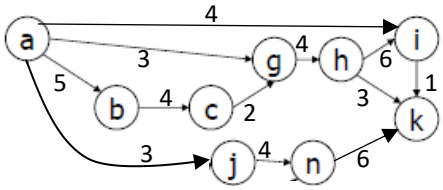
4. (2 punti)
Si determini mediante un algoritmo greedy un codice di Huffman ottimo per i seguenti simboli con le frequenze specificate:

a: 46 c: 20 g: 14 k: 3 i: 35 n: 13 o: 24 r: 19 t: 12 u: 17

5. (1 punto)
Si inseriscano in sequenza nella radice di un albero di ricerca binario supposto inizialmente vuoto le seguenti chiavi:

12 34 7 89 23

6. (2.5 punti)
Sia dato il seguente DAG pesato: considerando **a** come vertice di partenza, si determinino i cammini **massimi** tra **a** 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.



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

Appello del 26/01/2016 - Prova di programmazione (18 punti)

N città sorgono lungo una strada rettilinea. Ogni città è caratterizzata da un nome, dal numero di abitanti e dalla sua distanza rispetto all'inizio della strada. La distanza tra due città è data dalla differenza, in valore assoluto, tra le relative distanze dall'inizio della strada. Le informazioni sulle città sono lette da un file, il cui nome è passato come parametro sulla riga di comando. Il file ha il seguente formato:

- la prima riga contiene N (numero intero)
- in ognuna delle N righe successive compaiono una stringa di al massimo 20 caratteri per il nome della città, un intero per la popolazione (in migliaia) e un intero per la distanza rispetto all'inizio della strada.

Esempio: contenuto del file

```
11
Piacenza 102 0
Fidenza 27 35
Parma 190 57
ReggioEmilia 171 83
Modena 185 115
Bologna 386 144
Imola 69 177
Faenza 58 193
Forlì 118 207
Cesena 97 225
Rimini 147 253
```

Si vuole pianificare in quali città localizzare K<N Autorità di Ambito Territoriale Ottimale (ATO).. Sia K passato come parametro sulla riga di comando. Il sottoinsieme di K città va scelto in modo tale da minimizzare la distanza media, per ogni abitante, dalla sede di Autorità ATO più vicina. Questo risultato si può ottenere minimizzando la sommatoria SD delle distanze di ogni abitante dall'Autorità ATO più vicina:

$$\mathrm{SD} = \sum_{i=0}^{num.città-1} popolazione_i * distMinDaATO_i$$

Nella formula precedente *popolazione*$_i$ indica la popolazione dell'i-esima città, mentre *distMinDaAto*$_i$ ne indica la distanza dalla sede di Autorità ATO più vicina.

Si scriva in C un programma che, ricevuti come argomenti al main in nome del file e K:

1. legga il file e generi una opportuna struttura dati
2. calcoli la mutua distanza tra ogni città e tutte le altre, memorizzando tali informazioni in una struttura dati appropriata. La complessità dell'algoritmo deve essere O(N^2)
3. calcoli, mediante un algoritmo ricorsivo, la soluzione ottima richiesta (un sottoinsieme di K città in cui collocare le sedi ATO, in modo da minimizzare SD)
4. date due soluzioni, lette da due file (ognuno contenente l'elenco di K città, una per riga), determini quale delle due soluzioni è migliore.

Nei punti 3 e 4, al fine di determinare SD, si realizzi una opportuna funzione SommaDistanze, la quale, a partire dal vettore delle popolazioni e da un insieme di città selezionate come sedi dell'Autorità ATO, calcoli il relativo SD. La complessità di questa funzione è oggetto di valutazione. Si noti che è possibile realizzare SommaDistanze con un algoritmo O(N).

Indicare esplicitamente sulla prima pagina del foglio il modello di Calcolo Combinatorio utilizzato motivando la scelta in al massimo 3 righe.

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

Appello del 26/01/2016 - Prova di programmazione (12 punti)

1. (**2 punti**)

Scrivere la funzione `char *charErase (char *str, int *pos);` che riceve una stringa `str` e un vettore di interi `pos` e restituisce la stringa ottenuta da `str` cancellando i caratteri nelle posizioni indicate dal vettore `pos`. La stringa ritornata va opportunamente allocata. Il vettore `pos` ha dimensione ignota, ma il suo ultimo elemento contiene il valore -1.

Ad esempio, se la stringa `str` è `T h i s I s A S t r i n g` e il vettore `pos` contiene `7 4 2 0 11 -1` occorre cancellare dalla stringa `str` i caratteri `S, I, i, T, n` e restituire la stringa contenente

`h s s A t r i g`

2. (**4 punti**)

Sia data una matrice sparsa di `float` realizzata mediante un ADT (tipo `matr_t`) di prima categoria, che internamente gestisce i dati mediante una lista di liste. Si ricordi che una matrice si dice *sparsa* se viene realizzata con una struttura dati che alloca memoria per i soli dati non nulli. La lista di liste è così composta:

- una lista di primo livello contiene un elemento per ogni riga della matrice contenente almeno un dato non nullo,
- ad ogni elemento di questa prima lista corrisponde una sotto-lista, contenente un elemento per ogni dato non nullo della riga.

Al tipo `matr_t` corrisponde una `struct` wrapper contenente le dimensioni della matrice e il puntatore al primo elemento della lista di righe.

Si scriva la funzione `MatrWrite`, di prototipo

```
void MatrWrite (matr_t *M, float d, int r, int c);
```

che scrive nella matrice il dato `d` nella casella di indici `r`, `c`. Sono possibili quattro casi:

- `d != 0.0` e la matrice sparsa non contiene dati in riga `r` e colonna `c`. Il dato va aggiunto
- `d != 0.0` e la matrice sparsa contiene già un dato in riga `r` e colonna `c`. Il vecchio dato viene sostituito da quello nuovo
- `d == 0.0` e la matrice sparsa non contiene dati in riga `r` e colonna `c`. Non si fa nulla
- `d == 0.0` e la matrice sparsa contiene un dato in riga `r` e colonna `c`. Il dato va rimosso.

Si noti che NON si può far riferimento a funzioni di libreria sulle liste. Si richiede la realizzazione della funzione, nonché la definizione dei tipi `struct` (compreso `matr_t`) utilizzati.

3. (6 punti)

Sia data una stringa `str` di al massimo 30 caratteri e un vettore `lungh` di `num` interi distinti che rappresentano la lunghezza delle sottostringhe in cui si vuole decomporre la stringa `str`. Si scriva una funzione ricorsiva in C `void decomponi(char *str, int num, int *lungh);`

che visualizzi **una** delle possibili decomposizioni di `str` usando sottostringhe di lunghezza specificata nel vettore `lungh`.

Esempi:

- se `str= "tentativo", num = 3, lungh` contiene `2, 5, 7`, una delle possibili decomposizioni è `"te" "nt" "ativo"`
- se `str= "tentativo", num = 2, lungh` contiene `2, 4,` non vi sono decomposizioni.

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): entro venerdì 29/01/2016, 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.***