ESAME 13/02/2024 **PARTE PROGRAMMAZIONE (12 PUNTI)**

Domanda 2

Punteggio max.: 2,00

Contrassegna domanda

Sia dato un tipo ADT di prima classe SLIST (lista ordinata) con item corrispondente alla typedef qui riportata

typedef struct {char name[16]; int val;} Item;

L'item consiste quindi in un nome (contenente solo caratteri alfabetici) a cui è associato un valore intero. Gli item in lista sono ordinati in modo crescente in base al campo name.

Scrivere una funzione SLISTmerge, avente il prototipo seguente

SLIST SLISTmerge(SLIST a, SLIST b);

I due parametri a e b sono liste ordinate. Si noti che è possibile che una stessa stringa compaia come name in più di un item sia in a che in b (o in entrambe).

La funzione genera una terza lista, i cui item contengono (nei campi name) tutte e sole le stringhe presenti in a e in b. Ma non sono possibili ripetizioni: quando una stringa compare più volte in a e/o in b, la si riporta una volta sola nel risultato, associando come campo val la somma dei valori associati alla stringa in a e/o b.

Si fornisca la definizione del tipo SLIST, del nodo in lista, e si scriva la funzione. Qualora si usi una funzione newNode, non è necessario scriverla.

Esempio:

a) ("roma", 7), ("torino", 4), ("zagabria", 5)

b) ("roma", 3), ("torino", 3), ("torino", 2), ("venezia", 10)

Risultato: ("roma", 10), ("torino", 9), ("venezia", 10), ("zagabria", 5)

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
SLIST SLISTmerge(SLIST a, SLIST b) {

}
```

Domanda 3

Punteggio max.: 4,00

Contrassegna domanda

Sia dato un BST di interi (ADT di prima classe). Si scriva la funzione che genera un vettore (da allocare dinamicamente, senza riallocazione) di puntatori ai nodi del BST. Il vettore di puntatori deve essere ordinato secondo questo criterio: profondità crescente e, a pari profondità, valori (dell'intero contenuto nei nodi) crescenti.

La funzione deve essere richiamabile come

pnodes = BSTlevelizedNodes (b, &n);

Dove b è il BST, pnodes il (vettore) risultato e n il numero di nodi

E' richiesta la definizione dei tipo BST e del nodo.

Non è ammesso l'uso di funzioni di libreria.

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui le definizioni dei tipi

//scrivere qui la funzione

... BSTlevelizedNodes (...) {

}
```

Domanda 4

Punteggio max.: 6,00

Contrassegna domanda

E' dato un elenco di parole senza duplicati (vettore di stringhe).

Le stringhe in elenco posso essere utilizzate per generare stringhe più lunghe mediante concatenazione. Una concatenazione è una stringa generata dalla sequenza ordinata di tutte o parte delle stringhe in elenco prese al più una volta.

Si scriva una funzione bestConcat, che, dato l'elenco, generi la stringa più lunga ottenibile mediante concatenazione, rispettando un vincolo: date due parole consecutive nella sequenza, se la prima termina con vocale, la seconda non puo' iniziare con vocale. Se la prima termina con consonante, la seconda non puo' iniziare con consonante. Ai fini della bontà del risultato la lunghezza si misura in numero di caratteri (quindi non di parole).

Nota bene: le due domande a seguire **NON** sono facoltative. Assicurarsi di rispondere a tutte le richieste.

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

```
//scrivere qui il codice...

char *bestConcat (char **parole, int nparole) {
}
```

Domanda 5

Non valutata

Contrassegna domanda

Si giustifichi la scelta del modello combinatorio adottato.

Domanda 6

Non valutata

Contrassegna domanda

Si descrivano i criteri di pruning adottati o il motivo della loro assenza.

PARTE PROGRAMMAZIONE (18 PUNTI)

Informazione

Contrassegna domanda

Descrizione del problema

E' dato un grafo non orientato. NV indica il numero di vertici e NE indica il numero di archi. I nodi sono numerati da 0 a NV-1. A ogni nodo è associato un nome (una stringa di al più 20 caratteri alfabetici, terminatore compreso) e un valore (un numero intero). Il nome non è univoco, cioè più nodi possono essere associati allo stesso nome.

Un cammino nel grafo è quindi associato a:

- Una stringa determinata dalla concatenazione dei nomi associati ai nodi nel cammino

- Un valore complessivo, dato dalla somma dei valori dei nodi nel cammino.

Richieste del problema

A seguire una sintesi delle richieste del problema. Per ogni richiesta si troverà una domanda dedicata nelle sezioni a seguire con una descrizione più dettagliata per le richieste.

Strutture dati e letture

Definire opportune strutture dati per rappresentare i dati del problema e tutte le strutture dati ausiliarie ritenute opportune per la risoluzione dei problemi di verifica e di ricerca. Definire inoltre la funzione di lettura secondo il formato del file descritto nelle domande a seguire. Si chiede di utilizzare l'ADT GRAPH proposto nel corso, adattandolo al problema.

Problema di verifica

Si scriva una funzione checkString che, dato il grafo e una stringa, verifichi se la stringa può rappresentare la concatenazione dei nomi associati ai nodi su un cammino nel grafo

Problema di ottimizzazione

Si scriva la funzione bestPath, che, dato il grafo e un numero M, determini il cammino (eventualmente ciclico, ma con nomi ripetuti nel cammino al più M volte) di valore massimo, dove il valore è la somma dei valori associati a ogni vertice nel cammino. Esempio: se M valesse 2, significa che ogni nome può comparire al più 2 volte in un cammino: se il nome caratterizzasse due vertici, i vertici potranno comparire in un cammino entrambi, una volta sola, oppure solo uno dei due, fino a due volte.
Si tenga conto di un ulteriore vincolo: date due parole consecutive nella sequenza, se la prima parola termina con vocale, la seconda non può iniziare con vocale, se la prima termina con consonante, la seconda non può iniziare con consonante. E' sufficiente che il cammino venga stampato.

Domanda 7

Punteggio max.: 4,00

Contrassegna domanda

Strutture dati e acquisizione

Scrivere qui la definizione e implementazione delle strutture dati reputate necessarie a modellare le informazioni del problema, quali il grafo ed eventuali dati aggiuntivi, e la funzioni di acquisizione dei dati stessi. Si chiede di utilizzare l'ADT GRAPH proposto nel corso, con opportune modifiche per adattarlo al problema. Non è necessario realizzare completamente la GRAPHload, ma è sufficiente indicare le modifiche e scrivere in modo esplicito le eventuali aggiunte.

Il grafo è acquisito da un file testo nel formato

NV NE

<Elenco nodi (terne numero nome valore)>

<Elenco archi (coppie di numeri)>

Si puo' assumere che i nodi siano ordinati con identificatore (numero) crescente.

Esempio

4 6

0 sale 4

1 pepe 7

2 origano 6

3 cannella 8

4 peperoncino 5

5 origano 8

0 1

1 3

2 3

2 4

4 5

0 5

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

. . .

Domanda 8

Punteggio max.: 7,00

Contrassegna domanda

Problema di verifica

Si scriva una funzione checkString che, dato il grafo e una stringa, verifichi se la stringa può rappresentare la concatenazione dei nomi associati ai nodi su un cammino nel grafo.

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

. . .

Domanda 9

Punteggio max.: 9,00

Contrassegna domanda

Problema di ricerca e ottimizzazione

Si scriva la funzione bestPath, che, dato il grafo e un numero M, determini il cammino (eventualmente ciclico, ma con nomi ripetuti nel cammino al più M volte) di valore massimo, dove il valore è la somma dei valori associati a ogni vertice nel cammino. Esempio: se M valesse 2, significa che ogni nome può comparire al più 2 volte in un cammino: se il nome caratterizzasse due vertici, i vertici potranno comparire in un cammino entrambi, una volta sola, oppure solo uno dei due, fino a due volte.
Si tenga conto di un ulteriore vincolo: date due parole consecutive nella sequenza, se la prima parola termina con vocale, la seconda non può iniziare con vocale, se la prima termina con consonante, la seconda non può iniziare con consonante. E' sufficiente che il cammino venga stampato.

Attenzione! Si consiglia l'uso degli spazi al posto delle tabulazioni per l'indentazione del codice, dal momento che il carattere TAB viene utilizzato per la navigazione della pagina da parte della piattaforma.

. . .

PARTE TEORIA (12 PUNTI)

Domanda 1

Punteggio max.: 2,00

Contrassegna domanda

Sia data una coda a priorità di dati inizialmente vuota implementata mediante uno heap. Sia data la sequenza di interi e carattere *:

20 32 19 * 41 * 28 * 54 39 71 * * 7 * 1

dove ad ogni intero corrisponde un inserimento nella coda a priorità e al carattere * un'estrazione con cancellazione del **massimo**. Si eseguano **in sequenza** le operazioni di cui sopra.

Formato della risposta:

- contenuto del vettore che implementa lo heap dopo l'inserzione di 28
- contenuto del vettore che implementa lo heap dopo l'inserzione di 71
- contenuto del vettore che implementa lo heap dopo l'inserzione di 1

Domanda 2

Punteggio max.: 2,50

Contrassegna domanda

Data la catena di matrici (A_1, A_2, A_3, A_4) di dimensioni (3x6), (6x5), (5x4) e (4x2) rispettivamente, si determini mediante un algoritmo di programmazione dinamica la parentesizzazione ottima del prodotto di matrici che minimizza il numero di moltiplicazioni.

Domanda:

cosa contengono:

m[2][3]

s[2][3]

m[1][4]

s[1][4]

Formato della risposta:

m[2][3] = valore intero

s[2][3] = valore intero

m[1][4] = valore intero

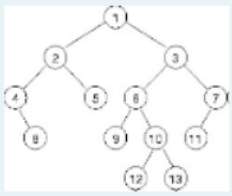
s[1][4] = valore intero

Domanda 3

Punteggio max.: 1,50

Contrassegna domanda

Si visiti in pre-order, in-order e post-order il seguente albero binario:



Formato della risposta:

pre-order:

elenco chiavi

el1, el2, el3, ... eln

in-order:

elenco chiavi

el1, el2, el3, ... eln

post-order:

elenco chiavi

el1, el2, el3, ... eln

Domanda 4

Punteggio max.: 2,00

Contrassegna domanda

Si inseriscano in sequenza le N=9 chiavi intere

138 48 103 60 213 28 98 70 215

in una tabella di hash (inizialmente vuota) gestita con open addressing con double hashing.

Si determini la dimensione minima M della tabella di per avere un fattore di carico alpha < 0.5.

Formato della risposta:

dimensione minima della tabella

M =

Quale tra le seguenti funzioni di hash è opportuno usare?

$h_1(k) = k \% 17, h_2(k) = 1+k \% 17$

$h_1(k) = k \% 17, h_2(k) = 1+k \% 97$

$h_1(k) = k \% 19, h_2(k) = 1+k \% 19$

$h_1(k) = k \% 19, h_2(k) = 1+k \% 97$

Si motivi brevemente la risposta.

Collisioni chiave 98:

cella1 dove vi è collisione:

...

cellan dove vi è collisione:

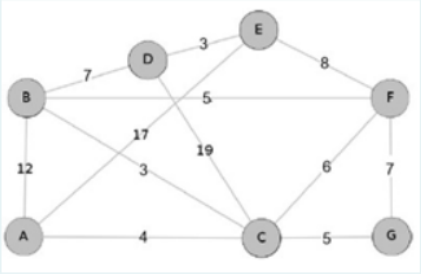
Al termine dell'inserzione, in quale cella si trova la chiave 215?

Domanda 5

Punteggio max.: 2,00

Contrassegna domanda

Si determini mediante l'algoritmo di Kruskal un albero ricoprente minimo per il grafo non orientato, pesato e connesso in figura.



Formato della risposta:

Indicare se gli archi seguenti appartengono all'MST o meno:

- (B,C): sì/no
- (D,E): sì/no
- (B,F): sì/no
- (C,G): sì/no
- (B,D): sì/no
- (F,G): sì/no

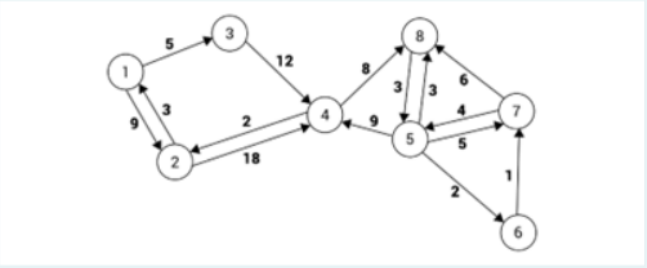
Peso minimo:

Domanda 6

Punteggio max.: 2,00

Contrassegna domanda

Si determinino per il seguente grafo orientato pesato mediante l'algoritmo di Dijkstra i valori di tutti i cammini minimi che collegano il vertice **5** con ogni altro vertice. Qualora necessario, si trattino gli archi secondo l'ordine numerico.



Formato: contenuto della coda a priorità vertice/distanza minima ad ogni passo (inf per ∞)

Passo 0

PQ = ={ }

Passo 1

PQ = ={ }

....

Passo n

PQ = ={ }

Il risultato finale è ottimo? Sì/No

Breve motivazione: