POLITECNICO DI TORINO

Algoritmi e programmazione

AP2021_III_Programmazione

	ALBERTO HUGONIN 249803
Iniziato	martedì, 15 giugno 2021, 18:23
Terminato	martedì, 15 giugno 2021, 20:03
Tempo impiegato	1 ora 40 min.

Informazione

Attenzione!

Indicare quale tipologia di esame si intenda svolgere rispondendo alla domanda a scelta multipla seguente (domanda 1). Si noti che è possibile leggere tutte o parte delle domande prima di effettuare la scelta e rispondere alla domanda 1.
Le domande dalla 2 alla 7 sono dedicate all'esame semplificato (traccia da 12pt).

In particolare:

- la domanda 2 fa parte dell'esercizio da 2 punti.
- le domande 3,4 fanno parte dell'esercizio da 4 punti.
- le domande 5,6,7 fanno parte dell'esercizio da 6 punti.

Le domande dalla 8 in poi sono dedicate all'esame completo (traccia da 18pt).

Informazione

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

- 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 riferiti dal codice dovranno essere inclusi 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 VENERDI' 18/06/2021, alle ore 23:59, 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 durante l'appello. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.

Domanda 1

Completo

Punteggio max.: 1,00

Indicare quale tipologia di esame si intende svolgere.

- ○ (a) 12 Punti - Semplificato
- ◉ (b) 18 Punti - Completo

Domanda 2

Risposta non data

Punteggio max.: 100,00

Esercizio da 2 punti

Sia data una matrice di interi NxN. Ogni cella della matrice può contenere solo il valore 0 o il valore 1. La matrice rappresenta delle relazioni di amicizia tra persone. Se in posizione i,j è contenuto il valore 1, le persone i e j sono amiche.

Scrivere una funzione che individui e stampi tutte le coppie i,j aventi almeno k amici in comune

```
void f(int **m, int n, int k);
```

Esempio: N=6 K=2

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Risultato: (0, 4) (1, 2) (1, 3) (1, 4) (1, 5) (3, 4) (3, 5) (4, 5)

Domanda 3

Risposta non data

Non valutata

Definire una struttura dati adeguata a rappresentare una lista singolo linkata di stringhe come ADT I classe, dove il tipo lista si chiami LIST.

Indicare esplicitamente in quale modulo/file appare la definizione dei tipi proposti.

La struttura così definita è quella da usare per completare la seconda domanda di questo esercizio.

Non è ammesso l'uso di funzioni di libreria.

Domanda 4

Risposta non data

Punteggio max.: 100,00

Implementare una funzione caratterizzata dal seguente prototipo:

```
void split(LIST l, char *str, char sep)
```

la funzione split suddivide la stringa str in una lista di (sotto)stringhe spezzandostr ogni volta che viene incontrato il carattere sep. Il carattere sep deve essere incluso come ultimo carattere della (sotto)stringa corrente

Non è ammesso l'uso di funzioni di libreria. Attenersi al prototipo dato.

Esempio #1

Input

str = "abracadabra", sep = 'r'

Output

$abr \rightarrow acadabr \rightarrow a$

Esempio #2

Input

str = "ramarro", sep = 'r'

Output

$r \rightarrow amar \rightarrow r \rightarrow o$

Domanda 5

Risposta non data

Punteggio max.: 100,00

Data una funzione wrapper con il seguente prototipo

```
void solve(char *start, char *end, sub *S, int nSubs);
```

Dove start e end sono due generiche stringhe lunghe uguali.

sub è una struttura definita come segue:

```
typedef struct {
 char *s;
 int pos;
 int costo;
} sub;
```

usata per rappresentare una sostituzione.

Una sostituzione è caratterizzata da una sottostringa s, con cui sovrascrivere una porzione di una stringa destinazione a partire dalla posizione pos. Ogni sostituzione ha un certo costo associato.

Es. di sostituzione:

start = "amico"

s1 = {s = "BA", pos = 1, cost = 4}
applicando la sostituzione s1 alla stringa start, si ottiene "aBAco"

s2 = {s = "TE", pos = 3, cost = 1}
applicando una ulteriore sostituzione s2, si ottiene "aBATE"

La trasformazione ha complessivamente costo 4 + 1 = 5

S è un vettore di sostituzioni ammissibili, di lunghezza nSubs.

Si assuma per semplicità che tutte le sostituzioni presenti in S siano totalmente applicabili alla stringa start ricevute in input, ossia che la posizione di applicazione e la lunghezza delle sostituzioni non vada a superare la lunghezza della stringa destinazione.

Scrivere una funzione ricorsiva che determini se è possibile individuare una sequenza di sostituzioni, tra quelle disponibili nel vettore S, in grado di trasformare la stringa start nella stringa end, minimizzando il costo totale di trasformazione. A parità di costo prediligere sequenze di sostituzioni più corte.

Es. completo:

```
start = "passato"
end = "persona"
nSubs = 8
S = {{"er", 1, 4},
{"ers", 1, 5},
{"sa", 3, 1},
{"so", 3, 2},
{"ato", 0, 1},
{"on", 4, 2},
{"ona", 4, 3},
{"a", 6, 1}}
passato
   + er in posizione 1 a costo 4
persato
   + ona in posizione 4 a costo 3
persona
COSTO: 7
```

Domanda 6

Risposta non data

Non valutata

Giustificare la scelta del modello combinatorio adottato.

Domanda 7

Risposta non data

Non valutata

Giustificare la scelta del/dei criteri di pruning adottati, o il motivo della loro assenza.

Informazione

PAGINA DI INTERMEZZO

La prova da 12 punti termina prima di questa pagina di intermezzo.

La prossima domanda rappresenta l'inizio della prova da 18 punti.

Informazione

Inizio prova da 18 punti

Descrizione

Un magazzino di stoccaggio è costituito da C corridoi. Ogni corridoio ospita C_s scaffali e ogni scaffale ha a disposizione un numero complessivo fisso K di slot, numerati da 0 a $K-1$.

Ogni corridoio è caratterizzato da un identificativo numerico unico, da 0 a $C-1$, e dal numero di scaffali che ospita. Ogni corridoio può ospitare un numero diverso di scaffali.

Ogni scaffale è caratterizzato da un codice identificativo alfanumerico unico all'interno del magazzino.
Si assuma che tutti gli scaffali del magazzino abbiano il medesimo numero di slot.

Il magazzino è utilizzato per conservare dei pacchi.
Ogni pacco è caratterizzato da un codice alfanumerico unico.

Per ogni pacco si è interessati a memorizzare la posizione dello stesso nel magazzino, in termini della terna <corridoio, scaffale, slot>.
Si assuma che tutti i pacchi siano di egual dimensione, e occupino un singolo slot ciascuno.

Definire delle opportune strutture dati così da poter rappresentare lo scenario precedentemente descritto, rispettando i seguenti vincoli:

- tipo "Pacco": implementazione a discrezione del candidato
- tipo "Scaffale": implementazione a discrezione del candidato
- tipo "Corridoio": collezione di "Scaffale" (**ADT I Cat**)
- tipo "Magazzino": collezione di "Corridoio" (**ADT I Cat**)

La base dati deve supportare le seguenti operazioni:

- Ricerca della posizione di un pacco, dato il codice,*con complessità al più logaritmica nel numero dei pacchi totali del magazzino*. Per questo punto, in caso si voglia usare un BST, si assuma che siano rispettate automaticamente le proprietà di bilanciamento. In caso si faccia uso di strutture dati di libreria, indicare esplicitamente la tipologia di "Item" contenuto e l'eventuale definizione del nodo contenitore;
- Inserimento di un pacco in una data posizione, con eventuale indicazione di errore a fronte di posizione già occupata/non esistente;
- Estrazione di un pacco da una data posizione, con eventuale indicazione di errore a fronte di posizione vuota/non esistente;
- Spostamento di un pacco da una posizione a un'altra, eventualmente in un corridoio/scaffale diverso, con gestione degli errori di cui sopra;
- Compattazione dei contenuti di due scaffali in uno solo dei due, se possibile

Organizzazione delle domande/risposte

Nelle pagine a seguire sono presenti cinque domande separate. Il primo blocco di risposta è dedicato a eventuale codice che non rientri in nessuno dei quattro moduli sopra richiesti (ad esempio, eventuali funzioni del client principale e/o del main se ritenute opportune, le chiamate di top-level per le funzionalità richieste, e così via...). A seguire, è presente un blocco di risposta per ognuno dei tipi di dato richiesti. Ogni blocco è in una pagina distinta.

Domanda 8

Completo

Punteggio max.: 1,00

Modulo Principale

Si scrivano qui eventuali funzioni che non rientrano in nessuno dei moduli dedicati alle strutture dati 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.

```
// Eventuale codice non appartenente a moduli specifici
#include "magazzino.h"

int main() {


Magazzino magaz = MagazzinoInit();
```

Domanda 9

Completo

Punteggio max.: 1,00

Gestione della struttura dati Pacco

Si scrivano le strutture dati e le funzioni principali per la gestione del modulo relative al tipo Pacco

---

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.

```
#DEFINE K 5
// pacco.h
DEFINE ID_S_LEN 10

//struttura pacco come item di un BST con chiave dell'item calcolata attraverso la f
unzione keyget questo permette di poter trovare un pacco con complessità logaritm
ica attraverso la BST search

typedef struct Pacco_ {
char codice[K];
//il primo indica il corridoio, il secondo lo slot nello scaffale di stringa pos_s
int pos[2];
char pos_s[ID_S_LEN];

} Pacco;
```

```
// pacco.c
int KeyGet(Pacco pacco) {
int key = 0;
for (int a=0; a<10; a++) {
key +=
}
}
```

Domanda 10

Completo

Punteggio max.: 1,00

Gestione della struttura dati Scaffale

Si scrivano le strutture dati e le funzioni principali per la gestione del modulo relative al tipo Scaffale

---

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.

```
// scaffale.h
#include "pacco.h"
#DEFINE MAX_SLOT 10

typedef struct Scaffale_ {

//identificativo scaffare
char id_s[ID_S_LEN];
//slot è un vettore di pacchi


} Scaffale;
```

```
// scaffale.c
...
```

Domanda 11

Completo

Punteggio max.: 1,00

Gestione della struttura dati Corridoio

Si scrivano le strutture dati e le funzioni principali per la gestione del modulo relative al tipo Corridoio

---

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.

```
// corridoio.h

typedef struct Corridoio_ *Corridoio

Corridoio CorridoioInit(int id_c, int C_s, int slot_s);
```

```
// corridoio.c

struct Corridoio_ {

//identificatore numerico corridoio
int id_c;

//numero scaffali in corridoio
int C_s;

//vettore allocato dinamicamente di scaffali
Scaffale *scaffali;
}

Corridoio CorridoioInit(int id_c, int C_s, int slot_s) {

    Corridoio corr = malloc(sizeof(struct Corridoio_));
    corr->id_c = id_c;
    corr->C_s = C_s;
    corr->scaffali = malloc(sizeof(Scaffali));
      for (int a=0; a<C_s; a++) {
          corr->scaffali[a] = ScaffaleInit(slot_s);
      }
   return corr;
}
```

Domanda 12

Completo

Punteggio max.: 1,00

Gestione della struttura dati Magazzino

Si scrivano le strutture dati e le funzioni principali per la gestione del modulo relative al tipo Magazzino

---

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.

```
//magazzino.h

#include "corridoio.h"


typedef struct Magazzino_ *Magazzino;

//num corridoi e dimensione di ciascun corridoio in scaffali
Magazzino MagazzinoInit(int c, int dim_c[], int slot_s);
```

```
// magazzino.c


struct Magazzino_ {
//numero di corridoi
int c;
//vettore di corridoi allocato dinamicamente
Corridoio corridoi;
}

Magazzino MagazzinoInit (int c, int dim_c[], int slot_s) {

    Magazzino magaz = malloc(sizeof(struct Magazzino_));
    magaz->c = c;
    magaz->corridoio = malloc(sizeof(Corridoio));
    for (int a=0; a<c; a++) {
        magaz->corridoio[a] = CorridoioInit(a,dim_c[a], int slot_s);
    }
return magaz;
}
```