


Algoritmi e strutture dati

ASD2024 - II Appello - Programmazione

	Stefano Alverino s312200
Iniziato	26 febbraio 2024, 10:31
Stato	Completato
Terminato	26 febbraio 2024, 12:11
Tempo impiegato	1 ora 40 min.
Valutazione	Non ancora valutato

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 6 sono dedicate all'esame semplificato (traccia da 12pt).

In particolare:

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

Le domande dalla 7 in poi sono dedicate all'esame completo (traccia da 18pt).

Un apposito separatore fa da intervallo tra le domande del compito da 12pt e le domande del compito 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 GIOVEDÌ 29/02/2024, alle ore 23:59, mediante caricamento su Portale
- assicurarsi di caricare l'elaborato nella Sezione Elaborati relativa all'a.a. 2023/24.
- 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

Non valutata

Indicare quale tipologia di esame si intende svolgere.

Scegli un'alternativa:

Domanda 1

a.

12 Punti - Semplificato

b.

18 Punti - Completo

Risposta corretta.

Ignorare il feedback di questa domanda.

La risposta corretta è: 18 Punti - Completo

Domanda 2

Risposta non data

Punteggio max.:
2,00

Sia data una matrice di caratteri definita come `char testo[NR][NC]`; contenente un testo, e un elenco di parole definito come `char **elenco`, vettore di `np` puntatori a carattere.

La matrice è già stata caricata in input: nel testo si possono considerare come "parole" le sotto-stringhe contenenti solo caratteri alfabetici, preceduti e seguiti da caratteri non alfabetici.

Si scriva una funzione `void paroleTrovate(...)` che generi un vettore dinamico contenente le parole dell'elenco trovate nel testo e per ognuna la lista delle posizioni (indice di riga e colonna) in cui la parola inizia nel testo.

Testo della risposta Domanda 2

Domanda 3

Risposta non data

Punteggio max.:
4,00

Sia dato un HEAP, compatibile con quanto visto a lezione (ADT di prima classe per l'HEAP, con priorità inclusa nell'ITEM). Si scriva una funzione `HEAPtoBT` che generi un albero binario (quindi realizzato mediante struct ricorsive allocate dinamicamente) equivalente (isomorfo e con contenuto identico) all'HEAP. Il prototipo della funzione sia

`BT HEAPtoBT(HEAP h).`

Si scriva poi la funzione avente prototipo

`Item BExtractLast(BT bt);`

La funzione cancella dall'albero binario l'ultima foglia (quella che nell'heap era in posizione `heapsize-1`) e ne ritorna il contenuto.

E' richiesta la definizione dei tipi HEAP e BT (come ADT di prima classe, con wrapper contenente i campi `root`, puntatore alla radice, e `size`, dimensione dell'albero), oltre che del nodo dell'albero binario.

Non è ammesso l'uso di funzioni di libreria.

Testo della risposta Domanda 3

Domanda 4

Risposta non data

Punteggio max.:
6,00

Si considerino le basi numeriche da 2 a 9. Indicando con B una base, un numero nella base B può essere rappresentato da una stringa contenente cifre comprese tra 0 (inclusa) e B (esclusa). Si scriva una funzione avente il prototipo: `void generaNumeri(int B, int NC, int minS)`.

La funzione deve generare e stampare tutti i numeri nella base B rappresentati su NC cifre, rispettando i seguenti vincoli:

- la cifra più significativa non può essere 0;
- al massimo una delle cifre presenti può apparire più di una volta nel numero;
- la somma delle cifre presenti nel numero deve essere maggiore o uguale a $minS$

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

Testo della risposta Domanda 4

Domanda 5

Risposta non data

Non valutata

Si giustifichi la scelta del modello combinatorio adottato.

Testo della risposta Domanda 5

Domanda 6

Risposta non data

Non valutata

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

Testo della risposta Domanda 6

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.

Descrizione del problema

Sono date N attività, ognuna caratterizzata da quattro elementi: nome dell'attività, tempo di inizio, durata, profitto o valore associato (≥ 0).

Il nome (stringa alfanumerica di lunghezza fino a 20 caratteri) è unico, cioè non sono possibili attività diverse con lo stesso nome. Ogni attività può poi avere un requisito di precedenza: una o due (non di più) altre attività devono essere terminate prima di iniziare l'attività stessa.

Occorre determinare il sottoinsieme di attività con profitto (somma dei profitti delle attività) massimo, compatibili tra loro, cioè tali che non si sovrappongano due attività nel sottoinsieme e siano rispettati gli eventuali vincoli di precedenza.

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/ottimizzazione. La struttura dati principale deve essere rappresentata come ADT di prima classe, compatibile con la definizione

```
typedef struct activities *ACT;
```

Si scriva la funzione avente prototipo: `ACT activityRead(FILE *f);`

che acquisisce i dati da un file testo, avente il formato

NA NP

<Elenco attività (quaterne: nome inizio durata valore)>

<Elenco precedenze (coppie o terne di nomi, il primo è l'attività vincolata, il secondo e/o il terzo sono le attività che vanno eseguite prima)>

L'ordine con cui sono riportate le attività e le precedenze è arbitrario. Segue un esempio:

4 2

Act1 1 1 50

Act2 3 2 20

Act4 2 98 200

Act3 6 13 100

Act4 Act1

Act3 Act2 Act 1

Problema di verifica

Si scriva una funzione avente prototipo `int checkSelection(ACT a, char **selected, int`

nsel); che, date le attività e un sottoinsieme selezionato, rappresentato da un elenco di nomi (e dal relativo numero), verifichi se la selezione è compatibile con i vincoli del problema. La selezione non deve necessariamente essere ottima.

Problema di ottimizzazione

Si scriva la funzione `bestSelection`, che, date le attività e i vincoli, determini un sottoinsieme ottimo di attività tale da rispettare i vincoli. La funzione deve tornare l'elenco delle attività selezionate (un vettore di nomi e il numero di attività), ordinate per tempo di inizio, nonché il profitto complessivo ottenuto.

Si noti che nell'esempio proposto il massimo profitto è 250, ottenuto selezionando Act1 e Act4, nonostante esista una sequenza compatibile più lunga/numerosa (Act1, Act2, Act3) ma con profitto inferiore (170).

Domanda 7

Completo

Punteggio max.:
4,00

Strutture dati e acquisizione

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/ottimizzazione. La struttura dati principale deve essere rappresentata come ADT di prima classe, compatibile con la definizione

```
typedef struct activities *ACT;
```

Si scriva la funzione avente prototipo: `ACT activityRead(FILE *f);`

che acquisisce i dati da un file testo, avente il formato

NA NP

<Elenco attività (quaterne: nome inizio durata valore)>

<Elenco precedenze (coppie o terne di nomi, il primo è l'attività vincolata, il secondo e/o il terzo sono le attività che vanno eseguite prima)>

L'ordine con cui sono riportate le attività e le precedenze è arbitrario. Segue un esempio:

4 2

Act1 1 1 50

Act2 3 2 20

Act4 2 98 200

Act3 6 13 100

Act4 Act1

Act3 Act2 Act 1

Testo della risposta Domanda 7

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.

```
...
```

```
//ACT.h
```

```
typedef struct activities *ACT;
```

```
ACT activityRead(FILE *f);
```

```
typedef struct attivita;
```

```
struct attivita  
{
```

```
    char nome[20];
```

```
    int inizio;
```

```
    int durata;
```

```
    int valore;
```

```
    struct attivita vincoli[2];
```

```
    int nVin;
```

```
};
```

```
int ACTfindByName(char *tmp, ACT elencoAct);
```

```
//ACT.c
```

```
struct activities{
```

```
    attivita *vettAttivita;
```

```
    int nAtt;
```

```
}
```

```
static ACT ACTinit(int nAtt);
```

```
ACT activityRead(FILE *f){
```

```
    int nA, nP,i,j,k, index, stop;
```

```
    chat tmp[20];
```

```

ACT elencoAct;

fscanf(f,"%d %d", &nA, &nP);

elencoAct = ACTinit(nA);

for(i=0;i<nA;i++){

    fscanf(f,"%s %d %d %d", elencoAct->vettAtt[i].nome,
&(elencoAct->vettAtt[i].inizio), &(elencoAct->vettAtt[i].durata),
&(elencoAct->vettAtt[i].valore));

}

for(i=0;i<nP;i++){

    fscanf(f,"%s", tmp);

    index = ACTfindByName(tmp, elencoAct);

    stop = 0;

    j=0;

    while(!stop){

        if(j==2){

            stop = 1;

        }

        fscanf(f,"%c", &carattere);

        if(carattere == '\n'){

            stop = 1;

            elencoAct->vettAtt[index].vincoli[j++] = tmp;

            elencoAct->vettAtt[index].nVin = j;

        }else{

            tmp[k++] = carattere;

        }

    }

}

return elencoAct;

}

static ACT ACTinit(int nAtt){

```

```

    ACT elencoAct;

    elencoAct = (ACT)malloc(sizeof(*elencoAct));

    elencoAct->vettAttivita = (*attivita)malloc(nAtt*sizeof(attivita));

    for(int i=0;i<elencoAct->nAtt;i++){

        elencoAct->vettAtt[i].nVin = 0;

    }

    elencoAct->nAtt = nAtt;

    return elencoAct;

}

int ACTfindByName(char *tmp, ACT elencoAct){

    for(i=0;i<elencoAct->nAtt;i++){

        if(strcmp(elencoAct->vettAtt[i].nome, tmp) == 0{

            return i;

        }

    }

    return -1;

}

```

Domanda 8

Completo

Punteggio max.:
6,00

Problema di verifica

Si scriva una funzione avente prototipo `int checkSelection(ACT a, char **selected, int nsel);` che, date le attività e un sottoinsieme selezionato, rappresentato da un elenco di nomi (e dal relativo numero), verifichi se la selezione è compatibile con i vincoli del problema. La selezione non deve necessariamente essere ottima.

Testo della risposta Domanda 8

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.

```

...

checkSelection(ACT a, char **selected, int nsel){

    int i,j, index1,index2,cnt;

```



```
for(i=1;i<nSel;i++){
```

```
    index1 = ACTfindByName(selected[i],a);
```

```
    for(j=i-1; j>=0; j--){
```

```
        index2 = ACTfinByName(selected[j],a);
```

```
        //Per verificare la condizione di non intersezione suppongo che le  
        attività non siano ordinate, se fossero ordinate basterebbe controllare la  
        precedente
```

```
        if(a->vettAtt[index1].inizio < a->vettAtt[index2].inizio &&  
a->vettAtt[index1].inizio + a->vettAtt[index1].durata >  
a->vetAtt[index2].inizio || a->vettAtt[index2].inizio <  
a->vettAtt[index1].inizio && a->vettAtt[index2].inizio +  
a->vettAtt[index2].durata > a->vettAtt[index1].inizio){
```

```
            reutrnr 0;
```

```
        }
```

```
    }
```

```
for(i=0;i<nSel; i++){
```

```
    index1 = ACTfindByName(selected[i],a);
```

```
    if(a->vetAtt[index1].nVin != 0){
```

```
        cnt = 0;
```

```
        for(j=0;j<a->vetAtt[index1].nVin; j++){
```

```
            for(k=0;k<nSel && cnt < a->vettAtt[index1].nVin;k++){
```

```
                if(strcmp(selected[k], a->vettAtt[index1].vincoli[j])==0){
```

```
                    cnt++;
```

```
                }
```

```
            }
```

```
        }
```

```
        if(cnt != a->vettAtt[index1].nVin){
```

```
            return 0;
```

```
        }
```

```
    }
```

```
}
```

```
}
```

```
return 1;
```

```
}
```

Domanda 9

Completo

Punteggio max.:
8,00**Problema di ricerca e ottimizzazione**

Si scriva la funzione `bestSelection`, che, date le attività e i vincoli, determini un sottoinsieme ottimo di attività tale da rispettare i vincoli. La funzione deve tornare l'elenco delle attività selezionate (un vettore di nomi e il numero di attività), ordinate per tempo di inizio, nonché il profitto complessivo ottenuto.

Si noti che nell'esempio proposto il massimo profitto è 250, ottenuto selezionando Act1 e Act4, nonostante esista una sequenza compatibile più lunga/numerosa (Act1, Act2, Act3) ma con profitto inferiore (170).

Testo della risposta Domanda 9

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.□

```
...
```

```
//utilizzo il modello del powerset con combinazioni semplici
```

```
void bestSelectionR(ACT a, char **bestNomi, int pNsel, char **sol, int pos, int k, int *bestProf);
```

```
int calcolaProfitto(char **sol, ACT a, int k);
```

```
void sortSol(ACT a, char **bestNomi, int nSel);
```

```
void QuickSortR(char **bestNomi, int l, int r);
```

```
void bestSelection(ACT a, char **bestNomi, int *pNsel){
```

```
    char **sol;
```

```
    int i,j, bestProf = 0;
```

```
    for(i=0;i<a->nAtt;i++){
```

```
        sol = (char **)malloc(i*sizeof(char*));
```

```
        for(j=0;j<i;j++){
```

```
            sol[i] = (char*)malloc(20*sizeof(char));
```

```
        }
```

```
        bestSelectionR(a,bestNomi,pNsel, sol, 0, i,&bestProf);
```

```
    for(j=0;j<j;j++){
```

```
free(sol[i]);
```

```
}
```

```
free(sol);
```

```
}
```

```
sortSol(a, bestNomi, *pNsel);
```

```
}
```

```
void bestSelectionR(ACT a, char **bestNomi, int* pNsel, char **sol, int pos,  
int k, int *bestProf){
```

```
int prof,i;
```

```
if(pos>=k){
```

```
prof = calcolaProfitto(sol);
```

```
if(prof >= *bestProf){
```

```
for(i=0;i<k;i++){
```

```
strcpy(bestNomi[i], sol[i]);
```

```
}
```

```
*pNsel = k;
```

```
}
```

```
return;
```

```
}
```

```
for(i=0;i<a->nAtt;i++){
```

```
sol[pos] = a->vettAtt[i]->nome;
```

```
if(checkSelection(a,sol,pos){
```

```
bestSelectionR(a, bestNomi, pNsel, sol, pos+1, k, bestProf);
```

```
}
```

```
}
```

```
}
```

```
int calcolaProfitto(char **sol, ACT a, int k){
```

```
int index,prof=0;
```

```
for(int i=0;i<k;i++){
```

```
index = ACTfindByName(sol[i],a);
```

```
prof+=a->vettAtt[index].valore;
```

```
}
```

```
return prof;
```

```
}
```

```
void sortSol(ACT a, char **bestNomi, int nSel){
```

```
    QuickSortR(bestNomi, 0, nSel);
```

```
}
```

```
//Implementare variante del quickSort che lavori sui tempi di inizio
```