POLITECNICO DI TORINO

Algoritmi e strutture dati

ASD2023_II_Programmazione [AULA]

	MARCO PERNO 294424
Iniziato	martedì, 21 febbraio 2023, 10:00
Terminato	martedì, 21 febbraio 2023, 11:40
Tempo impiegato	1 ora 40 min.

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' 24/02/2023, alle ore 14:00, mediante caricamento su Portale
- **SOLO LAUREANDI che avessero bisogno di una correzione anticipata** consegna delle relazioni, per entrambe le tipologie di prova di programmazione: entro MERCOLEDI' 22/02/2023, alle ore 14:00, 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 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.

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.

Domanda 1

Completo

Non valutata

Indicare quale tipologia di esame si intende svolgere.

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

Domanda 2

Risposta non data

Punteggio max.: 2,00

Sia data una matrice M di dimensione r x c contenente singoli caratteri minuscoli.
Scrivere una funzione che generi una matrice M' di dimensioni opportune derivata da M mantenendo solo le righe/colonne dove non sia presente alcuna vocale {a, e, i, o, u}.
La matrice M' sia allocata dentro alla funzione.
Completare opportunamente il prototipo in modo che la nuova matrice e le rispettive dimensioni siano disponibili al chiamante.

void f(char **M, int r, int c, ...);

Esempio

$$M = \begin{Bmatrix} a & c & f & e & g \\ z & y & t & t & p \\ q & w & j & e & t \\ p & l & l & n & m \end{Bmatrix} \quad \rightarrow \quad M' = \begin{Bmatrix} y & t & p \\ l & l & m \end{Bmatrix}$$

Nota

Potenzialmente la funzione potrebbe dover ritornare una matrice nulla, di dimensione 0 x 0.

Domanda 3

Risposta non data

Punteggio max.: 4,00

Definire una struttura dati adeguata a rappresentare una lista doppio linkata di interi come ADT I classe, e il relativo nodo.
Il tipo lista si chiami LIST.
La lista definita non deve fare uso di sentinelle.
Indicare esplicitamente in quale modulo/file appare la definizione dei tipi proposti.

Usando i tipi precedentemente definiti si scriva una funzione void f(LIST l, int a, int b) che elimini dalla lista tutti i nodi il cui valore sia compreso tra a e b, estremi inclusi.

Non è ammesso l'uso di funzioni di libreria.

Domanda 4

Risposta non data

Punteggio max.: 6,00

Una macchina industriale può produrre pezzi di diversi tipi, ognuno caratterizzato da un valore e da un tempo di produzione, valori interi e positivi.
Si assuma per comodità che i vari tipi di pezzi siano univocamente identificati da un singolo intero nell'intervallo 0 .. P-1.
Ricevuto in input un tempo T e un valore obiettivo V, entrambi interi e positivi, identificare quanti pezzi di ogni tipo sia possibile produrre entro il tempo T per minimizzare la differenza, in valore assoluto, tra V e la somma dei valori del pezzi prodotti, ossia per avvicinarsi il più possibile all'obiettivo.

Domanda 5

Risposta non data

Non valutata

Si giustifichi la scelta del modello combinatorio adottato.

Domanda 6

Risposta non data

Non valutata

Si descrivano i 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

Descrizione del problema

Un'azienda deve pianificare l'assegnazione del proprio personale a una serie di incarichi. L'azienda ha a disposizione P persone da suddividere tra T incarichi. Ogni persona deve essere assegnata a un singolo incarico. Per comodità si assuma che le persone siano identificate da un indice intero nel range 0 .. P-1 e gli incarichi da un indice nel range 0 .. T-1.

L'azienda valuta la difficoltà e il valore di ogni incarico con un singolo valore intero d_i. Tali valori sono memorizzati in un vettore D. Ogni dipendente dell'azienda ha un proprio livello di esperienza personale e_i. Tali valori sono memorizzati in un vettore E.
Tra il personale dell'azienda si instaurano anche delle sinergie, mappate in una matrice quadrata simmetrica S di dimensioni P x P. Ogni cella S[i][j] della matrice rappresenta il contributo dato da avere il dipendente i e il dipendente j assegnati al medesimo incarico.

Perché un incarico possa essere svolto con successo occorre assegnare personale tale per cui la somma dell'esperienza individuale e i contributi dovuti alle sinergie raggiungano almeno il 75% del rispettivo valore d_i. Se non è possibile raggiungere tale soglia per un certo incarico, il contributo di tutta la forza lavoro assegnata è sostanzialmente perduto. È possibile assegnare più personale del dovuto a un certo incarico, ma tutto il contributo oltre il valore d_i dell'incarico stesso è irrilevante.
La resa di ogni incarico svolto con successo è quindi data dal minimo tra il valore complessivo della forza lavoro assegnata (individuale e sinergica) e il valore d_i dell'incarico stesso. Data una assegnazione di persone ai vari incarichi, il valore complessivo dell'assegnazione corrisponde alla somma delle rese dei singoli incarichi svolti con successo.

Esempio

D = {10,6,8} // Vettore degli incarichi
E = {3,2,5,3} // Vettore dell'esperienza delle persone
S = // Matrice di sinergia

```
{
  { 0, 2, 1, 4 },
  { 2, 0, 4, 1 },
  { 1, 4, 0, 3 },
  { 4, 1, 3, 0 }
}
```

Assegnando le persone di indice 0 e 3 all'incarico di indice 0 si riuscirebbe a soddisfare completamente il valore d_0 con resa pari a 10 (3+3 individuali e +4 di sinergia).
Assegnando le persone di indice 0 e 1 all'incarico di indice 0 si arriverebbe a un contributo lavorativo pari a 7 (3+2 individuali e +2 di sinergia) che non permette di raggiungere il 75% richiesto, per cui la resa è completamente nulla.
Assegnando le persone di indice 0 e 2 all'incarico di indice 0 si arriverebbe a un contributo lavorativo pari a 9 (3+5 individuali e +1 di sinergia) che permette di svolgere l'incarico con resa pari a 9.
Assegnando le persone di indice 0 e 3 all'incarico di indice 1 si riuscirebbe a soddisfare (in esubero) completamente il valore d_1, con resa pari a 6.

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
Definire opportune strutture dati per rappresentare le informazioni presentate in precedenza e quanto sia ritenuto necessario alla soluzione dei problemi di verifica e di ricerca. Acquisire i dati nelle strutture definite.

Problema di verifica
Data una proposta di assegnazione completa persone-incarichi, valutare che questa rispetti i dati del problema e in caso affermativo valuti la resa complessiva dell'assegnazione proposta, secondo le regole illustrate in precedenza.

Problema di ottimizzazione
Identificare una assegnazione tale da massimizzare la resa complessiva.

Domanda 7

Completo

Non valutata

Strutture dati e acquisizione

Definire opportune strutture dati per rappresentare le informazioni che caratterizzano il contesto e quanto sia ritenuto necessario alla soluzione dei problemi di verifica e di ricerca.

Scrivere qui la definizione e implementazione delle strutture dati reputate necessarie a modellare le informazioni del problema secondo quanto richiesto e le funzioni di acquisizione dei dati stessi. In caso di organizzazione delle strutture dati su più file, indicare esplicitamente il modulo di riferimento.

Si assuma che le informazioni relative al problema siamo riportate in un unico file input.txt con il seguente formato:

- Sulla prima riga è riportata il numero T di incarichi.
- La seconda riga contiene T valori interi separati da un singolo spazio a rappresentare il valore d di ogni incarico
- Sulla terza riga è riportata il numero P di persone.
- La quarta riga contiene P valori interi separati da un singolo spazio a rappresentare il valore e di ogni persona
- Seguono P righe composte da P interi separati da un singolo spazio, ciascuna, a rappresentare i valori della matrice S

Rispetto all'esempio presentato in precedenza, il contenuto del file corrispondente sarebbe il seguente (*i commenti appaiono solo come ulteriore chiarimento. non fanno parte del file!*):

```
3 // T
10 6 8 // Valori associati ai T = 3 incarichi
4 // P
3 2 5 3 // Esperienza delle P = 4 persone
0 2 1 4 // Prima riga della matrice S (4x4)
2 0 4 1
1 4 0 3
4 1 3 0 // Ultima riga della matrice S
```

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.

```
#person.h
typedef struct person_wrapper_s * Person_Wrapper;
Person_Wrapper PERSONLoad(FILE * fin);
int ** PERSONBuildMat(int P);
void PERSONFree(Person_wrapper p_wrapper);



#person.c
typedef struct {
    int id; // corrispondennza id pos per sicurezza
    int exp;
    int available;
} Person;

struct person_wrapper_s {
    int N;
```

```
    Person * vett;
};

Person_Wrapper PERSONLoad(File * fin) {
    int N;
    fscanf(fin, "%d", &N);
    Person_Wrapper p_wrapper = malloc(sizeof(struct person_wrapper_s));
    p_wrapper->vett = malloc(sizeof(Person)*N);
    p_wrapper->N = N;
    for(int i = 0; i < N; i++) {
        p_wrapper[i].id = i;
        fscanf(fin, "%d", &(p_wrapper->vett[i].exp));
        p_wrapper->vett[i].available = 1;
    }
    return p_wrapper;
}

void PERSONFree(Person_Wrapper p_wrapper) {
    free(p_wrapper->vett)
    free(p_wrapper);
}

int ** PERSONBuildMat(int P) {
    int mat[P][P];
    for(int i = 0; i < P; i++) {
        for(int j = 0; j < P; j++) {
            fscanf(fin, "%d", &(mat[i][j]);
        }
    }
    return mat;
}

#activity.h
typedef struct activity_wrapper_s * Activity_Wrapper;
Activity_Wrapper ACTIVITYLoad(FILE * fin);
void ACTIVITYFree(Activity_Wrapper a_wrapper);

#activity.c
typedef struct {
    int id; //corrispondenza id pos per sicurezza
    int diff;
} Activity;

struct activity_wrapper_s {
    int N;
    Activity * vett;
}
```

```
Activity_Wrapper ACTIVITYLoad(FILE * fin) {
    int N;
    fscanf(fin, "%d", &(N));
    Activity_Wrapper a_wrapper = malloc(sizeof(struct activity_wrapper_s));
    a_wrapper->N = N;
    a_wrapper->vett = malloc(sizeof(Activiyt)* N);
    for(int i = 0; i < N; i++) {
        a_wrapper->vett[i].id = i;
        fscanf(fin, "%d", &(a_wrapper->vett[i].diff);
    }
    return a_wrapper;
}

void ACTIVITYFree(Activiti_wrapper a_wrapper) {
    free(a_wrapper->vett);
    free(a_wrapper);
}
```

Domanda 8

Completo

Non valutata

Problema di verifica

Data una proposta di assegnazione, valutare che questa rispetti i dati del problema e in tal caso determinare il valore della resa complessiva corrispondente.

Il formato dei dati in input per questa richiesta è a discrezione del candidato, **che è tenuto a fornirne anche una breve descrizione**.

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.

```
// formato file: prima riga numero incarichi previsti, seconda riga num_person_primo incarico e id_attivi
tà come id/pos nel vettore del wrapper, terza riga elenco id persone per prima attività come id/pos e co
si via per gli latri incarichi

int ** generateComb(int * sol, int pos, int * val, int N; int k, int *** vett_sol, int cont_vett_sol, int * N_vett
_sol) {
```

```
    if(pos == k) {
        // Terminal case
        if(*N_vett_sol == *cont_vett_sol) {
            *N_vetT_sol *= 2;
            *vett_sol = realloc(*vett_sol, *N_vett_sol * sizeof(int *));
        }
        *vett_sol[cont_vett_sol] = malloc(sizeof(int) * k);
        for(int i = 0; i < N; i++) {
            *vett_sol[cont_vett_sol][i] = sol[i];
        }

    }
    else {
        for(int i = pos; i < N; i++) {
            sol[pos] = val[i];
            generateComb(sol, pos + 1, val, N, k, vett_sol, cont_vett_sol N_vett_sol);
        }
    }
}

int searchSinergie(int * person, int N_person, int **mat) {
    if(N_person == 1)
        return 0;
    int sinergie = 0;
    int N_comb_person = 1;
    int ** comb_person = malloc(sizeof(int *) * 1);
    comb_person[1] = malloc(sizeof(int) * 2);
    int cont_comb_person;
    int sol[2];
    generateComb(sol, pos, person, N_person, 2, &comb_person, &cont_comb_person, &N_comb_per
som);
    for(int i = 0; i < cont_comb_person; i++) {
        sinergie+=mat[comb_person[i][0]] + mat[com_person[i][1]];
    for(int  i = 0; i < N_com_person; i++) {
        free(comb_person[i]);
    }
    free(comb);
    return sinergie;
}

int verify(int * person, int N_person, int id_activity, Person_Wrapper p_wrapper, Activity_Wrapper
a_wrapper, int ** mat) {
    int punt = searchSinergie(person, N_person, mat);
    int valid = 1;
    for(int i = 0; i < N_person && valid == 1; i++) {
        if(person[i].avaialble == 0)
            valid = 0
```

```
        else {
            punt += person[i].exp;
        }
      }
    if(valid == 1 && punt > a_wrapper->vett[id_activity] / 100 *75)
        return punt;
    retrun -1;
}


void ReadFileSol(FILE * fin, Person_Wrapper p_wrapper, Activity_Wrapper a_wrapper, int ** mat) {
    int N_activities;
    fscanf(fin, "%d", &(N_activities));
    int valid = 1;
    for(int i = 0; i < N_activities && valid == 1; i++) {
        int N_person, id_activity;
        fscanf(fin, "%d", &N_person, &id_activity);
        int p_sol[N_person]
        for(int i = 0; i < N; i++) {
            fscanf(fin, %d", sol[i]);
        }
        if(verifySol() >= 0) {
            for(int i = 0; i < N_person; i++) {
                p_wrapper->vett[p_sol[i]].avalaible = 0;
            }
        }
        else
            validate = 0;
    }
    if(validate == 1)
        printf("ok");
    else
        printf("errata");
}
```

Domanda 9

Completo

Non valutata

Problema di ricerca e ottimizzazione

Identificare una assegnazione tale da massimizzare la resa complessiva secondo le regole illustrate in precedenza, ovvero:

Perché un incarico possa essere svolto con successo occorre assegnare personale tale per cui la

somma dell'esperienza individuale e i contributi dovuti alle sinergie raggiungano almeno il 75% del valore rispettivo valore d_i. Se non è possibile raggiungere tale soglia per un certo incarico, il contributo di tutta la forza lavoro assegnata è sostanzialmente perduto. È possibile assegnare più personale del dovuto a un certo incarico, ma tutto il contributo oltre il valore d_i dell'incarico stesso è irrilevante.
La resa di ogni incarico svolto con successo è quindi data dal minimo tra il valore complessivo della forza lavoro assegnata (individuale e sinergica) e il valore d_i dell'incarico stesso. Data una assegnazione di persone ai vari incarichi, il valore complessivo dell'assegnazione corrisponde alla somma delle rese dei singoli incarichi svolti con successo.

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.

```
void ER(int * val, int pos, int m, int k, Activity_wrapper a_wrapper, Perosn_wrapper p_wrapper, int ** mat, int * exit) {
    if(pos == m && m == k) {
        int ** sol;
        int * dim_sol;
        //Ricostruisco sol
        int valid = 1;
        for(int i = 0; i < k; i++)
            if(verify(sol[i],....) <  0)
                valid = 0;
        if(valid == 1)
            *exit = 1
    }
    else {
        for(int i = 0; i < m && * exit == 0; i++) {
            sol[pos] = i;
            ER(..pos+1..);
        }
        sol[pos] = ;
        ER(.. pos+1, m+1..)
    }
}

void generateBEstSol(Person_Wrapper p_wrapper, Activity_Wrapper a_wrapper, int ** mat) {
    int N_best_sol;
```

```
    int ** best_sol = malloc(...)
    int id_activities[a_wrapper->N]

        int vett_id_person[p_wrapper->N]
        for(int i = 0; i < p_wrapper->N; i++) {
            vett_id_person[i] = i;
        }
    for(int i = a_wrapper->N; i != 0; i--) {
        if(i == a_wrapper->N)
            ER(vett_id_person,..)
        else {
            int a_wrappert_tmp = malloc(..)
            int N;
            int ** tmp = disp_sempl(a_wrapper->vett, &N)
            for(int i = 0; i < tmp; i++) {
                a_wrapper-tmp ->vett = tmp[i];
                a_wrapper_tmp->N= i;
                ER(vett_id_person, ... a_wrapper_tmp, ..)
        }
    }
}
```