03AAX Algoritmi e Strutture Dati

Appello del 07/02/2023 - Prova di programmazione (18 punti)

1. (**18 punti**)

In un noto videogioco, il giocatore deve superare delle prove di *hacking* inserendo delle sequenze di token esadecimali di due caratteri in un buffer di lunghezza fissa per ottenere vari bonus. Si consideri la seguente descrizione del mini-gioco.

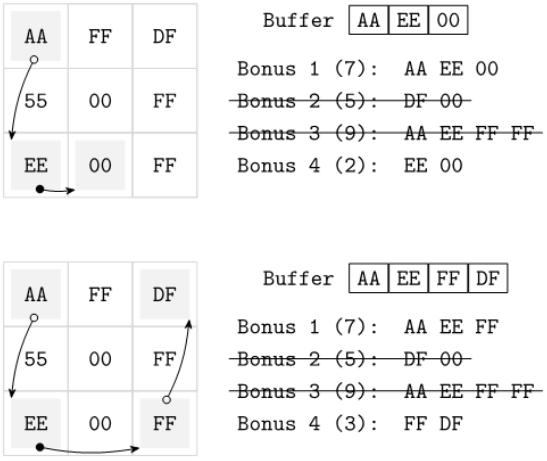
I token vengono scelti da una matrice quadrata di dimensione `N x N`. Ogni token della matrice può essere usato una singola volta, ma nella matrice possono apparire dei duplicati. La scelta dei token avviene alternando una serie di movimenti per riga e per colonna, partendo sempre con una scelta iniziale sulla prima riga. I movimenti possono interessare un qualsiasi token lungo una certa riga/colonna, indipendentemente dalla distanza dalla posizione corrente. Non sono mai ammessi movimenti in diagonale. Lo schema di movimento/scelta è quindi simile alla sequenza a seguire:

- Il primo token (scelta di indice zero) viene sempre scelto sulla prima riga
- Il secondo token (scelta di indice uno) viene scelto nella colonna in cui si sia scelto il token precedente
- Il terzo token (scelta di indice due) viene scelto nella riga in cui si sia scelto il token precedente
- E così via, alternando colonna/riga nelle mosse a seguire...

I bonus ottenuti dal giocatore sono associati a delle stringhe obiettivo, composte anch'esse da token esadecimali. Per semplicità, si assuma che il bonus sia un valore intero positivo. Il giocatore ottiene tutti i bonus la cui stringa associata appaia come sottostringa nel buffer riempito dal giocatore.

Il programma riceve in input la lunghezza `L` del buffer, un file contenente la matrice di gioco `grid.txt` e un file contente la lista di bonus disponibili `bonus.txt`. Il file `grid.txt` riporta sulla prima riga la dimensione `N` della matrice e a seguire `N` righe di `N` token esadecimali separati da un singolo spazio. Il file `bonus.txt` riporta sulla prima riga il numero di `B` bonus disponibili e a seguire `B` righe nel formato: `<numero_token> <valore_bonus> <sequenza_di_token>`

Esempi:



Si faccia riferimento all'immagine proposta, assumendo che per il primo esempio si abbia `L=3` e per il secondo esempio si abbia `L=4`. **(Es. 1, sopra)** il primo e quarto bonus sono entrambi ottenibili introducendo la sequenza `AA EE 00` (che corrisponde alla selezione dei token di posizione `[0,0],[2,0],[2,1]`), che rappresenta anche la sequenza a valore massimo che si possa introdurre. Il secondo bonus non è ottenibile poiché non c'è nessuna sequenza di movimenti, indipendentemente dalla lunghezza del buffer, che permetta di introdurre `DF 00` nel buffer stesso. Il terzo bonus non è ottenibile poiché il buffer ha lunghezza `L=3`. Buffer più lunghi avrebbero reso possibile ottenere quel bonus poiché la sequenza corrispondente esiste come sequenza di mosse valide nella griglia (`[0,0],[2,0],[2,2],[1,2]`). **(Es. 2, sotto)** il buffer di lunghezza quattro permetterebbe di completare tutte le sequenze, individualmente o meno, sempre con l'eccezione della seconda che non è realizzabile data la griglia di input. Nell'immagine è proposta una scelta sub-ottima, che copre il primo e il quarto bonus. L'ottimo sarebbe stato raggiunto preferendo la scelta dei quattro token `AA EE FF FF` (terminando in [1,2] anziché [0,2]) che coprirebbe contemporaneamente sia il Bonus 1 sia il Bonus 3, anziché Bonus 1 e Bonus 4 come invece si è scelto nella configurazione rappresentata.
Nelle immagini si sono distinte mosse obbligate per riga (colonna) con il pallino pieno (vuoto) come marker di inizio della freccia.

Richieste:

- **(Strutture Dati)** Definire e implementare opportune 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.
- **(Verifica)** Data una sequenza di scelte sulla griglia, valutare che questa rispetti i vincoli di movimento esposti e determinare la composizione del buffer. Se valida, valutare il valore della somma dei bonus ottenibili. Il formato dei dati in input per questa richiesta è a discrezione del candidato, che è tenuto a fornirne anche una breve descrizione.
- **(Ottimizzazione)** Identificare la scelta di token con cui riempire il buffer che massimizzi la somma dei bonus ottenibili.

03AAX Algoritmi e Strutture Dati

Appello del 07/02/2023 - Prova di programmazione (12 punti)

1. (2 punti)

Sia data una matrice `M` di dimensione `r x c` contenente elementi interi. Scrivere una funzione che generi una matrice `M'` di dimensione `r x c` derivata da `M` sostituendo il valore di ogni cella con la media di tutti i suoi vicini (diagonali incluse) e la cella stessa. Per gli angoli la media sarà quindi tra 4 valori, per le celle lungo i bordi (angoli esclusi) tra 6 elementi e per tutte le altre celle tra 9 elementi. La matrice `M'` sia allocata dentro alla funzione. Completare opportunamente il prototipo in modo che la nuova matrice sia disponibile al chiamante.

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

Esempio:

$$M = \begin{Bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{Bmatrix} \qquad M' = \begin{Bmatrix} 1.25 & 1.33 & 1.25 \\ 1.33 & 1.33 & 1.33 \\ 1.25 & 1.33 & 1.25 \end{Bmatrix}$$

2. (4 punti)

Si scriva una funzione wrapper `int f(T t)` (e la relativa funzione ricorsiva) che ricevuto in input un albero n-ario di interi `t` di tipo `T` conti la lunghezza del cammino più lungo contenente solo valori positivi o nulli. Fornire la definizione del tipo `T` e del tipo nodo al suo intero, come ADT di prima classe e come quasi ADT rispettivamente, indicando esplicitamente la divisione in moduli adottata. Per rappresentare l'albero richiesto, ogni nodo tiene traccia dei figli mediante un vettore di puntatori a nodo e il numero di figli.

Non è ammesso l'utilizzo di funzioni di libreria.

3. (6 punti)

Una matrice `M` quadrata di dimensione `N x N` rappresenta le distanze dirette tra `N` città. Ogni cella contenente un valore positivo `d` indica che la coppia di città `(i,j)` dista `d`. Un valore pari a zero indica l'assenza di una connessione diretta tra le due città. Scrivere una funzione ricorsiva in grado di suddividere le città nel minor numero di gruppi possibili facendo in modo che tutte le città in un gruppo siano mutualmente raggiungibili (direttamente o transitivamente) attraversando al massimo una città intermedia. È ammesso che ci siano gruppi composti da una singola città.

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

- ***indicare nell'elaborato e nella relazione nome, cognome e numero di matricola.***
- ***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 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 il 10/02/2023, 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 in aula. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

03AAX Algoritmi e Strutture Dati

Appello del 21/02/2023 - Prova di programmazione (12 punti)

1. (2 punti)

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:

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

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

2. (4 punti)

Definire una struttura dati adeguata a rappresentare una lista doppio linkata di interi come ADT di 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 avente il seguente prototipo:

```
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.

3. (6 punti)

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 dei pezzi prodotti, ossia per avvicinarsi il più possibile all'obiettivo.

03AAX Algoritmi e Strutture Dati

Appello del 21/02/2023 - Prova di programmazione (18 punti)

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

Strutture dati e letture
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
```

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**.

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.

03AAX Algoritmi e Strutture Dati

Appello del 11/05/2023 - Prova di programmazione (18 punti)

1. (**18 punti**)

Dato un grafo non orientato non pesato $G = (V, E)$ si definisce *triangle packing* una collezione $V_1, V_2, \dots, V_k$ di sottoinsiemi disgiunti di vertici, ognuno contenente esattamente tre vertici, tali per cui per ogni $\mathrm{V_i} = \{\mathrm{u_i}, \mathrm{v_i}, \mathrm{w_i}\}$ con $1 \leq i \leq k$ tutti e tre gli archi (u_i, v_i), (u_i, w_i) e (v_i, w_i) esistono e appartengono a E. Un triangle packing può non coprire tutti i vertici del grafo. Per gli scopi del problema si è interessati all'individuazione di un triangle packing a cardinalità massima.

Strutture dati e acquisizione

Fornire la definizione e implementazione delle strutture dati reputate necessarie a modellare le informazioni del problema, quali il grafo e un triangle packing, 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 il grafo in input sia riportato in un file di nome `grafo.txt`, organizzato come segue:

- Sulla prima riga appare il numero V di vertici
- Seguno un numero indefinito di righe riportanti coppie (u, v) di interi, con $0 \leq u, v < V$ a rappresentare gli archi del grafo

Problema di verifica

Data una soluzione proposta, verificare che questa rappresenti effettivamente un triangle packing, tenendo in considerazione la definizione teorica proposta in precedenza. Si trascuri la richiesta di cardinalità massima in questo contesto. La soluzione proposta deve essere letta da file il cui nome e formato è a discrezione del candidato, che è tenuto a fornire anche una breve spiegazione dei contenuti del file stesso.

Problema di ricerca e ottimizzazione

Identificare, se possibile, un triangle packing a cardinalità massima per il grafo dato in input.

03AAX Algoritmi e Strutture Dati

Appello del 11/05/2023 - Prova di programmazione (12 punti)

1. (**2 punti**)

Sia data una matrice `M` di dimensione `r x c` contenente elementi interi. Scrivere una funzione che generi una matrice `M'` di dimensione `r x c` derivata da `M` in cui ogni elemento `[i][j]` assume il valore della somma cumulata di tutti gli elementi di indice inferiore, lungo la riga `i` e la colonna `j`, incluso l'elemento `[i][j]` originale il cui contributo è contato una singola volta. La matrice `M'` sia allocata dentro alla funzione. Completare opportunamente il prototipo in modo che la nuova matrice sia disponibile al chiamante.

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

Esempio:

$$M = \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{Bmatrix} \Rightarrow M' = \begin{Bmatrix} 1 & 3 & 6 \\ 5 & 11 & 18 \\ 12 & 22 & 33 \end{Bmatrix}$$

2. (**4 punti**)

Si fornisca la definizione delle strutture dati `LIST` e `NODE`, come ADT di I categoria e quasi ADT rispettivamente, per rappresentare una lista singolo linkata di interi, senza sentinelle. Suddividere il codice in modo opportuno tra file `.h` e `.c`

Si scriva una funzione `void f(LIST l)` che ricevuta in input una lista di interi (rappresentata facendo riferimento ai tipi definiti in precedenza), già ordinata in ordine crescente, compatti i contenuti della lista cancellando tutti i nodi uguali consecutivi, ad eccezione del primo di ogni gruppo.

Non è ammesso l'utilizzo di funzioni di libreria.

Esempio:

$$1 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 5 \rightarrow 7 \rightarrow 7$$

diventa

$$1 \rightarrow 2 \rightarrow 5 \rightarrow 7$$

3. **(6 punti**)

Una matrice binaria `M` di dimensione `O x S` rappresenta l'appartenenza di una serie di oggetti a una serie di insiemi. Un oggetto `o_i` appartiene all'insieme `s_j` se nella matrice appare un 1 nella cella `[i][j]`. L'unico altro valore ammesso nella matrice è il valore 0, per indicare non appartenenza. Ogni colonna della matrice rappresenta quindi un sottoinsieme dell'insieme universo di oggetti. Scrivere una funzione ricorsiva in grado di identificare, se possibile, l'insieme a cardinalità minima di sottoinsiemi disgiunti in grado di coprire l'insieme universo. Si giustifichi la scelta del modello combinatorio adottato. Si descrivano i criteri di pruning adottati o il motivo della loro assenza.

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

- ***indicare nell'elaborato e nella relazione nome, cognome e numero di matricola.***
- ***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 il 14/05/2023, 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 in aula. Non verranno corretti i compiti di cui non sarà stata inviata la relazione nei tempi stabiliti.***

03AAX Algoritmi e Strutture Dati

Appello del 30/06/2023 - Prova di programmazione (12 punti)

1. (2 punti)

Sia data una matrice `M` di dimensione `r x c` contenente elementi interi.
Scrivere una funzione che generi una matrice `M'` di dimensione `r x c` derivata da `M` in cui ogni elemento `[i][j]` assume il valore della somma cumulata di tutti gli elementi sulla medesima diagonale e antidiagonale, considerando solo elementi il cui indice di colonna sia maggiore o uguale all'elemento preso in considerazione. Il contributo dell'elemento `[i][j]` originale è contato una singola volta.
La matrice `M'` sia allocata dentro alla funzione.
Completare opportunamente il prototipo in modo che la nuova matrice sia disponibile al chiamante.

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

Esempio:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Longrightarrow M' = \begin{bmatrix} 15 & 8 & 3 \\ 14 & 17 & 6 \\ 15 & 14 & 9 \end{bmatrix}$$

2. (4 punti)

Si fornisca la definizione delle strutture dati `LIST` e `NODE`, come ADT di I categoria e quasi ADT rispettivamente, per rappresentare una lista singolo linkata di interi, senza sentinelle. Suddividere il codice in modo opportuno tra file .h e .c
Si scriva una funzione `void f(LIST l)` che ricevuta in input una lista di interi (rappresentata facendo riferimento ai tipi definiti in precedenza), compatti i contenuti della lista cancellando tutti i nodi il cui indice, ossia la cui posizione originale, sia divisibile per 3. La posizione di ogni nodo NON è salvata nel nodo stesso. Si assuma che la testa sia il nodo di posizione/indice 0.
Non è ammesso l'uso di funzioni di libreria.

Esempio:

$$1 \rightarrow -2 \rightarrow 3 \rightarrow 8 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 9$$

diventa

$$-2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 9$$

3. (6 punti)

Dato un vettore `V` di interi positivi, di dimensione nota `d`, identificare se possibile un modo di suddividerne il contenuto in `x` sottoinsiemi, con `x` parametro del problema, tale per cui la somma di tutti gli elementi di ognuno dei sottoinsiemi sia la stessa. Ogni elemento del vettore deve essere assegnato a un singolo sottoinsieme.
La ricerca, in caso di successo, deve terminare alla prima soluzione valida trovata.

03AAX Algoritmi e Strutture Dati

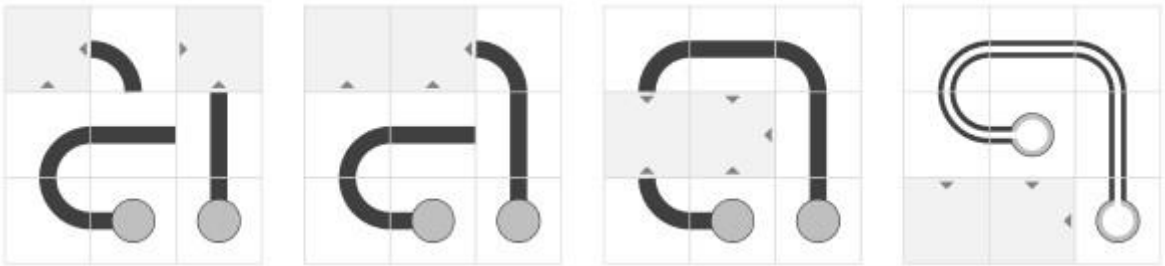
Appello del 30/06/2023 - Prova di programmazione (18 punti)

Descrizione del problema

Una matrice quadrata di dimensione `NxN` rappresenta una griglia di gioco per uno "sliding puzzle".
La griglia contiene una serie di tessere semoventi e alcuni buchi.
La regola di movimento è la seguente: il giocatore ad ogni passo può scegliere una direzione (su/giù/destra/sinistra) e tutte le tessere libere di muoversi in quella direzione (c'è un buco nella cella destinazione) si muovono di conseguenza in contemporanea di UNA posizione. A ogni passo deve muoversi almeno una tessera, altrimenti la mossa non è valida (non cambierebbe lo stato del problema). Le tessere non possono muoversi uscendo dai bordi della griglia.
Tutte le tessere (ad eccezione di due, descritte a seguire) sono caratterizzate da avere due lati collegati da un canale.
Due tessere speciali, sorgente e destinazione, hanno un singolo lato di connessione.
Scopo del gioco è muovere le tessere in modo date che esista un canale contiguo che connetta sorgente e destinazione.
Nota bene: sorgente e destinazione sono formalmente interscambiabili tra loro come ruolo. Non c'è una vera direzione di flusso tra le due.
Non è necessario coinvolgere tutte le tessere nella creazione del canale complessivo, l'importante è che sorgente e destinazione siano connesse.
Il giocatore ha a disposizione al massimo `M` mosse, parametro del problema.

Esempio

Nell'esempio proposto è rappresentata una griglia `3x3` con due buchi (inizialmente, angolo alto a sinistra e alto a destra), cinque tessere standard (due lati comunicanti) e le due tessere terminali (sorgente/destinazione tonda grigia e singolo lato aperto).
Le frecce grigio scuro ai bordi di tessere confinanti con un buco evidenziano direzioni di movimento ammesse per ogni tessera, e di conseguenza tutte le tessere coinvolte da un movimento in quella direzione.
Nell'esempio proposto, la sequenza di movimenti DESTRA, SU, SU porta alla soluzione del puzzle.
Si ricorda che la singola mossa muove tutte le tessere libere di spostarsi nella direzione specificata.



Richieste del problema

Strutture dati e letture
Scrivere la definizione e implementazione delle strutture dati reputate necessarie a modellare le informazioni del problema, 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 lo schema di gioco in input sia riportato in un file di nome `grid.txt`, organizzato come segue:

- Sulla prima riga appare una coppia di interi `N` e `T` dove `N` è la dimensione del lato della griglia e `T` il numero di tessere presenti
- Seguono `NxN` righe ognuna caratterizzata da una quaterna di valori 0/1 a indicare i lati connessi dal canale nell'ordine nord-sud-ovest-est, a rappresentare la griglia di gioco in rappresentazione row-major
- Righe dove appaiono due zeri sono tessere normali, in cui gli 1 rappresentano i lati comunicanti
- Righe dove appare un singolo 1 sono tessere sorgente/destinazione
- Righe in cui appare una quaterna di zeri sono buchi.

Con riferimento all'esempio presentato in precedenza, la configurazione iniziale della griglia corrisponde a un file i cui contenuti sono i seguenti.
Nota bene: i commenti sono a mero scopo descrittivo, e non fanno parte del file.

```
3 7
0 0 0 0 // Buco
0 1 1 0 // Tessera con connessione SUD-OVEST
0 0 0 0 // Buco
0 1 0 1 // Tessera con connessione SUD-EST
0 0 1 1 // Tessera con connessione OVEST-EST
1 1 0 0 // Tessera con connessione NORD-SUD
1 0 0 1 // Tessera con connessione NORD-EST
0 0 1 0 // Sorgente/Destinazione con connessione a OVEST
1 0 0 0 // Sorgente/Destinazione con connessione a NORD
```

Problema di verifica
Data una soluzione proposta rappresentata da una sequenza di movimenti, verificare che questa rappresenti una sequenza valida tenendo conto delle regole di movimento espresse in precedenza. Valutare inoltre se questa sequenza sia in grado di risolvere o meno il puzzle.
La soluzione proposta deve essere letta da file il cui nome e formato è a discrezione del candidato, che è tenuto a fornire anche una breve spiegazione dei contenuti del file stesso.

Problema di ottimizzazione
Identificare, se possibile, una sequenza di movimenti in grado di risolvere il puzzle, tenendo conto delle regole di movimento espresse in precedenza. Si ricorda che non è necessario fare uso di tutto le tessere: l'obiettivo è solo quello di creare un canale continuo tra sorgente/destinazione.

03AAX Algoritmi e Strutture Dati

Appello del 20/09/2023 - Prova di programmazione (12 punti)

1. (2 punti)

Sia data una matrice `M` di dimensione `r x c` contenente elementi interi.
Scrivere una funzione che generi una matrice `M'` di dimensione `r x c` derivata da `M` in cui ogni elemento `[i][j]` assume il valore della somma cumulata dei primi `k` elementi adiacenti lungo la riga `i` e la colonna `j`, in entrambe le direzioni, escluso l'elemento `[i][j]` originale. Nel caso non ci fossero elementi sufficienti lungo una certa direzione, ci si limiti a usare quelli disponibili.
La matrice `M'` sia allocata dentro alla funzione.
Completare opportunamente il prototipo in modo che la nuova matrice sia disponibile al chiamante.

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

Esempio:

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & -2 & 5 \\ 1 & 2 & 3 & 4 \\ -2 & -3 & -1 & 0 \\ -5 & 1 & -2 & 9 \end{pmatrix} \xrightarrow{k=2} M' = \begin{pmatrix} 8 & 13 & 8 & 14 \\ 1 & 6 & 15 & 9 \\ 1 & 11 & 5 & 23 \\ -6 & 3 & -6 & 14 \\ -2 & 1 & 7 & 3 \end{pmatrix}$$

2. (4 punti)

Si fornisca la definizione delle strutture dati `LIST` e `NODE`, come ADT di I categoria e quasi ADT rispettivamente, per rappresentare una lista doppio linkata di caratteri, senza sentinelle. Suddividere il codice in modo opportuno tra file .h e .c
Si scriva una funzione `void f(LIST l, int k)` che ricevuta in input una lista rappresentata facendo riferimento ai tipi definiti in precedenza, già ordinata in ordine alfabetico crescente, compatti i contenuti della lista cancellando tutti i nodi uguali consecutivi, ad eccezione del primo di ogni gruppo, solo se la sotto-lista di nodi uguali consecutivi è composta `k` da almeno elementi.

Esempio:

$$a \leftrightarrow a \leftrightarrow b \leftrightarrow b \leftrightarrow b \leftrightarrow c \leftrightarrow d \leftrightarrow d$$

diventa per `k=3`

$$a \leftrightarrow a \leftrightarrow b \leftrightarrow c \leftrightarrow d \leftrightarrow d$$

3. (6 punti)

Dato un multi-insieme di interi $S = \{a_1, \dots, a_n\}$ di dimensione nota `n` e un intero $0 < s \leq \sum_{i=1}^{n} a_i$ identificare, se possibile, due multi-insiemi disgiunti $S_1 \subset S$ e $S_2 \subset S$ tali per cui valga la relazione:

$$\sum_{j=1}^{|S_1|} a_j = s + \sum_{k=1}^{|S_2|} a_k$$

A fronte di più soluzioni ammissibili, preferire quella che usa più elementi possibili di S, ossia per cui si abbia $max\ \{|S_1| + |S_2|\}$.

03AAX Algoritmi e Strutture Dati

Appello del 20/09/2023 - Prova di programmazione (18 punti)

Descrizione del problema

Dato un grafo non orientato e pesato $G = (V, E)$, si definisce *k-capacitated tree partition* una collezione $E_1, \dots, E_n$ di sottoinsiemi di E *disgiunti per vertici*, tali per cui per ogni E_i il grafo indotto è un albero di almeno `k` vertici. Tutti i vertici del grafo originale devono appartenere a uno dei grafi indotti. Per gli scopi del problema si è interessati all'individuazione di un partizionamento tale per cui valga $min \sum_i \sum_{e \in E_i} w(e)$, dove $w(e)$ rappresenta il peso del generico arco $e \in E_i$. Sia `k` un parametro del problema.

Richieste del problema

Strutture dati e letture

Scrivere la definizione e implementazione delle strutture dati reputate necessarie a modellare le informazioni del problema, quali il grafo e il relativo tree partitioning, 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 il grafo n input sia riportato in un file di nome `grafo.txt`, organizzato come segue:

- Sulla prima riga appare il numero `V` di vertici;
- Seguono un numero indefinito di righe riportante coppie (u, v) di interi, con $u \geq 0$ e $v < V$ a rappresentare gli archi del grafo, e il relativo peso $w_{(u,v)}$.

Problema di verifica

Data una soluzione proposta, verificare che questa rappresenti effettivamente un k-capacitated tree partitioning, tenendo in considerazione la definizione teorica proposta in precedenza. Si trascuri la richiesta di minimalità della somma complessiva dei pesi per gli archi scelti.

La soluzione proposta deve essere letta da file il cui nome e formato è a discrezione del candidato, che è tenuto a fornire anche una breve spiegazione dei contenuti del file stesso.

Problema di ricerca e ottimizzazione

Identificare, se possibile, un k-capacitated tree partitioning che rispetti anche la condizione di minimalità della somma dei pesi per gli archi scelti, per il grafo dato in input e il relativo parametro `k` passato come argomento.

03AAX Algoritmi e Strutture Dati

Appello del 13/02/2024 - Prova di programmazione (12 punti)

1. (2 punti)

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 sia in `a` che 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 in `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)}

2. (4 punti)

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 segue:

```
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.

3. (6 punti)

È 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 può iniziare con vocale; se la prima termina con consonante, la seconda non può iniziare con consonante. Ai fini della bontà del risultato la lunghezza si misura in numero di caratteri (quindi non di parole).

03AAX Algoritmi e Strutture Dati

Appello del 13/02/2024 - Prova di programmazione (18 punti)

Descrizione del problema

È 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

Strutture dati e letture

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 seguente formato:

```
NV NE
<Elenco nodi (terne numero nome valore)>
<Elenco archi (coppie di numeri)>
```

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

Esempio:

```
6 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
```

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 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.

03AAX Algoritmi e Strutture Dati

Appello del 26/02/2024 - Prova di programmazione (12 punti)

1. (2 punti)

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 sottostringhe 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.

2. (4 punti)

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:

```
BTHEAPtoBT(HEAP h);
```

Si scriva poi la funzione avente prototipo:

```
Item BTextractLast(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 ATD di prima classe, con wrapper contenente i campi root, puntatore alla radice e size, cioè dimensione dell'albero), oltre che del nodo dell'albero binario.
Non è ammesso l'uso di funzioni di libreria.

3. (6 punti)

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.

03AAX Algoritmi e Strutture Dati

Appello del 26/02/2024 - Prova di programmazione (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 (numero positivo).
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

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 seguente formato:

```
NA NP
<Elenco attività (quaterne: nome inizio durata valore)>
<Elenco precedenze (coppie o terne di nomi, ove 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 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).

03AAX Algoritmi e Strutture Dati

Appello del 09/05/2024 - Prova di programmazione (12 punti)

1. (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 terne di persone in cui ogni persona è amica delle altre due. Il prototipo della funzione è:

```
void f(int **m, int N);
```

Si noti che una persona puo' appartenere a più terne. Ogni terna va stampata con gli indici delle tre persone.

2. (4 punti)

Si scriva una funzione (wrapper) `char *decode(H h, char *str)` che, ricevuto in input un albero binario `h`, rappresentante la codifica di Huffman associata a un certo set di caratteri, e una stringa di caratteri `str` che contiene una sequenza di 0/1, la decodifichi, sulla base della codifica memorizzata nell'albero `h`, ritornando come risultato una stringa decodificata.
Fornire inoltre la definizione del tipo `H` (come ADT di prima classe) e del tipo nodo al suo interno (come quasi ADT).

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

3. (6 punti)

Si scriva una funzione che date due stringhe, ne trovi la più lunga sottosequenza comune. La funzione ha prototipo:

```
char *maxSubs(char *s1, char *s2);
```

Si ricorda che una stringa è una sequenza di caratteri (terminata da `'\0'`), e che una sottosequenza è un sottoinsieme (ordinato) di elementi di una sequenza, non necessariamente contigui.
Ad esempio, la più lunga sottosequenza comune a "una stringa" e "narrativa" è "natia".
La stringa risultato va allocata dinamicamente.
Il problema sarebbe risolvibile mediante un approccio basato su programmazione dinamica, ma si richiede espressamente che sia risolto sfruttando uno dei modelli combinatori visti a lezione, di cui si chiede di fornire anche una breve giustificazione della scelta fatta.

03AAX Algoritmi e Strutture Dati

Appello del 09/05/2024 - Prova di programmazione (18 punti)

Descrizione del problema

E' dato un grafo orientato pesato. `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 unico (una stringa di al più 20 caratteri alfabetici, terminatore compreso).

Richieste del problema

Strutture dati e letture

Definire e implementare le 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, quindi NON è necessario realizzare la `GRAPHload`. È invece necessario definire una opportuna struttura dati (come ADT di prima classe `vertexSeq`) per la rappresentazione di un elenco di vertici, da utilizzare sia per i due cammini `P1` e `P2` che per gli insiemi `S1` e `S2`. Per questi va prevista una funzione di lettura da file, con prototipo:

```
vertexSeq readVertexSeq(FILE *fp);
```

Il file contiene un elenco di nomi di vertici, uno pe riga.

Problema di verifica

Si scriva una funzione `checkCommonPath` che, dati due cammini `P1` e `P2` nel grafo e un intero `min`, verifichi se i due cammini sono fattibili e se hanno un sotto-cammino comune di lunghezza maggiore o uguale a `min`. La funzione abbia prototipo:

```
int checkCommonPath (GRAPH g, vertexSet p1, vertexSet p2);
```

Problema di ricerca e ottimizzazione

Si scriva la funzione `bestPath`, che, dato il grafo e due insiemi di vertici `S1` e `S2`, trovi (se esiste) il cammino ottimo (dove il criterio di ottimizzazione è dato dalla somma dei pesi degli archi nel cammino) nel grafo, tale che siano rispettati i seguenti requisiti/vincoli:

- tutti i vertici nel cammino debbono appartenere ad almeno uno degli insiemi S1 o S2;
- almeno la metà dei vertici nel cammino deve appartenere sia a S1 che a S2;
- il cammino deve essere semplice.

03AAX Algoritmi e Strutture Dati

Appello del 04/07/2024 - Prova di programmazione (12 punti)

1. (2 punti)

Sia dato un vettore di stringhe `v` (puntatori a `char`). Le stringhe contengono al loro interno delle parole chiave comprese tra angolari (caratteri `'<'` e `'>'`). Si vuole realizzare una funzione `estraiParoleChiave` che rimuova da `v` le parole chiave, inserendole in un vettore dinamico di parole chiave. La funzione deve essere richiamata come segue :

```
parole = estraiParoleChiave(v);
```

Si richiede di realizzare solamente la funzione `estraiParoleChiave`.

Esempio

Supponendo che le stringhe in `v` siano:

```
"La prima <riga> comprende <due> parole chiave"
"La seconda ne comprende <una> sola"
```

il programma deve compattare le stringhe come segue:

```
"La prima <> comprende <> parole chiave"
"La seconda ne comprende <> sola"
```

e ritornare un vettore dinamico contenente le seguenti stringhe: `"riga"`, `"due"`, `"una"`.

2. (4 punti)

Sia dato un `BST`, in cui si vogliono estendere le informazioni presenti in ogni nodo mediante statistiche/annotazioni aggiuntive. Si scriva una funzione in grado di assegnare a ogni nodo del `BST` la dimensione e l'altezza del sottoalbero radicato nel nodo stesso.

La funzione ha il seguente prototipo:

```
BST extend(BST b);
```

Si richiede, oltre alla funzione, la definizione del tipo `BST` e del tipo usato per il nodo.

3. (6 punti)

Sia dato un vettore di parole (stringhe contenenti solo caratteri alfabetici minuscoli). Si definisce come "intersezione" tra due stringhe l'insieme di caratteri comuni. Attenzione: un insieme non è ordinato.

Si vuole trovare l'insieme di caratteri che costituisce intersezione per il massimo numero di coppie di stringhe.

La funzione ha il seguente prototipo:

```
char *maxStringInters(char **parole, int nParole);
```

03AAX Algoritmi e Strutture Dati

Appello del 04/07/2024 - Prova di programmazione (18 punti)

Descrizione del problema

Sia dato un grafo orientato pesato, in cui:

- `NV` indica il numero di vertici;
- `NE` indica il numero di archi;
- i nodi sono numerati da `0` a `NV-1`
- a ogni nodo è associato un nome unico (una stringa di al più 20 caratteri alfabetici, terminatore compreso).

Richieste del problema

Struttura dati

Definire opportune strutture dati per rappresentare i dati del problema.
Si assuma di utilizzare un grafo standard, compatibile con quanto sviluppato a lezione. Si fornisca la definizione del tipo `GRAPH`. Non è necessaria l'implementazione della funzione `GRAPHload`.
Definire inoltre un tipo di dato `graphPath` e uno `graphPathSet` (entrambi come ADT di prima classe): i due tipi rappresentano, rispettivamente, un cammino e un insieme di cammini nel formato più idoneo a risolvere in modo efficiente i problemi proposti.
Va realizzata una funzione `readGraphPathSet` avente il seguente prototipo:

```
graphPathSet readGraphPathSet(GRAPH g, FILE *fp);
```

La funzione acquisisce un insieme di cammini da un file, in cui i cammini sono rappresentati mediante elenco di nomi di vertici separati da spazi o a-capo. Tra un cammino e il successivo c'è una riga contenente solo il carattere `'#'`.

Qui di seguito un esempio di contenuto di un file rappresentante un insieme formato da due cammini:

```
Torino Milano Venezia
#
Bologna Ancona Pescara
```

Problema di verifica

Si scriva una funzione `checkPathSet` che, dato un insieme di cammini, verifichi che nessun cammino abbia vertici interni (si escludono quindi il primo e l'ultimo vertice del cammino stesso) comuni con gli altri cammini. La funzione deve avere complessità O(NV).

Problema di ricerca e ottimizzazione

Si scriva la funzione `bestPath` che, dato il grafo `g` e l'insieme di cammini `ps`, trovi (se esiste) il cammino di valore massimo (il valore di un cammino è dato dalla somma dei pesi degli archi) nel grafo, tale che siano rispettati i seguenti requisiti/vincoli:

- il cammino deve essere ottenuto mediante concatenazione di (non necessariamente tutti) cammini appartenenti a `ps`;
  N.B: i cammini `p1` e `p2` sono concatenati se il vertice finale di `p1` coincide con quello iniziale di `p2`.
- il cammino deve essere semplice;
- a parità di valore, va selezionato il cammino contenente più vertici.

03AAX Algoritmi e Strutture Dati

Appello del 13/09/2024 - Prova di programmazione (12 punti)

1. (2 punti)

Siano dati due vettori di interi ordinati in modo crescente e privi di ripetizioni.
Si scriva una funzione che generi un vettore (allocato dinamicamente) contenente gli interi appartenenti al primo vettore e non al secondo. La funzione deve essere chiamata come segue:

```
c = diffVett(a,na,b,nb,&nc);
```

ove `a` e `b` sono i due vettori, `na` e nb il numero di dati che contengono; `c` è il vettore risultato, allocato dinamicamente nella funzione, `nc` il numero di interi nel vettore risultato.
Si richiede di realizzare solamente la funzione `diffVett` (quindi non il programma chiamante).

2. (4 punti)

Sia dato un `BST` avente come valori delle stringhe, che fungono anche da chiave di ricerca.
Si scriva una funzione che determini la foglia a profondità massima `MAXF` (in caso di uguaglianza, si selezioni la foglia con chiave maggiore) e stampi a ritroso (quindi da foglia a radice) le chiavi sul cammino che connette la foglia `MAXF` alla radice. La funzione ha il seguente prototipo:

```
void BSTprintDeepest(BST b);
```

Si richiede, oltre alla funzione, la definizione del tipo `BST` (ADT di prima classe) e del tipo usato per il nodo.

3. (6 punti)

Una superficie piana rettangolare viene suddivisa in `N` righe e `M` colonne ed è rappresentata da una matrice `NxM` di caratteri, in cui si usa in carattere `'0'` per rappresentare una casella libera (utilizzabile) e il carattere `'1'`per una casella occupata (non utilizzabile).
Si vuole realizzare una funzione che, data la matrice e le coordinate di due caselle libere (di coordinate `(r0,c0)` e `(r1,c1)`), determini il percorso di lunghezza minima per connetterle.
Un percorso è dato da una sequenza di caselle libere adiacenti (aventi riga o colonna in comune) a due a due; detto in altri termini, un percorso si ottiene mediante tratti orizzontali o verticali comprendenti solo caselle libere. La lunghezza di un percorso è data dal numero di caselle che lo compongono. Del percorso ottimo va solo calcolata e ritornata come risultato la lunghezza.
La funzione ha il seguente prototipo:

```
int minPath(char **area, int N, int M, int r0, int c0, int r1, int c1);
```

Esempio
Matrice `4x5` in cui si è rappresentato con delle `'x'` il percorso ottimo (di lunghezza `5`) che connette la casella `(1,0)` alla casella `(2,3)`:

```
0 1 0 0 0
x x x 1 0
0 1 x x 0
0 0 0 0 0
```

03AAX Algoritmi e Strutture Dati

Appello del 13/09/2024 - Prova di programmazione (18 punti)

Descrizione del problema

La Regione deve decidere la localizzazione di un servizio di Pronto Soccorso per `N` città. Tra queste occorre selezionare `M < N` città che possono essere sede di Pronto Soccorso. L'obiettivo del problema è determinare le sedi di Pronto Soccorso ed assegnare ad ognuna di esse il servizio di altre città, dati vincoli e criteri per valutare costi e benefici.

Date:

- le distanze tra tutte le coppie di città, come matrice `NxN` di interi, ove ogni città dista 0 da se stessa;
- una distanza massima tollerabile `MAXD` tra sede di un Pronto Soccorso e una città servita;
- un numero minimo `MINS` di città servibili da ogni sede di Pronto Soccorso (siccome una sede di Pronto Soccorso serve ovviamente se stessa, `MINS` tiene conto solo delle altre città non sede di Pronto Soccorso);

un insieme di `M` città sede di Pronto Soccorso è accettabile se sono rispettati i vincoli seguenti:

- per ognuna delle `N-M` città, esiste almeno una sede di Pronto Soccorso (tra le `M` scelte) a distanza minore o uguale a `MAXD`;
- ognuna delle sedi di Pronto Soccorso è in grado di servire almeno `MINS` città non sedi di Pronto Soccorso, rispettando per ognuna il vincolo precedente.

Richieste del problema

Struttura dati

Definire opportune strutture dati per rappresentare:

- L'elenco delle città (tipo `ELENCO`);
- La matrice delle distanze (tipo `DISTMATR`);
- Un insieme di sedi di Pronto Soccorso (tipo `SEDI`, vedere i punti successivi);
- Un'assegnazione di città alle sedi (tipo `SERVIZI`, la soluzione del problema di ottimizzazione).

In particolare, tutte le strutture dati precedenti devono essere ADT di prima classe.

Si scriva la funzione `caricaDati` che legge un file così composto:

- la prima riga contiene il valore `N`;
- le `N` righe successive contengono i nomi di `N` città;
- le ulteriori `N` righe successive rappresentano la matrice quadrata delle distanze, ove ogni riga contiene `N` interi separati da spazi.

La funzione ritorna un elenco di città (tipo `ELENCO`) e una matrice delle distanze (tipo `DISTMATR`).

Qui di seguito un esempio di contenuto del file, ove, per semplicità, si usano nomi di un solo carattere:

```
6
A
B
C
D
E
F
0 2 5 4 2 6
2 0 6 6 2 5
5 6 0 7 4 3
4 6 7 0 6 3
2 2 4 6 0 6
6 5 3 3 6 0
```

Problema di verifica

Si assuma di voler enumerare i possibili insiemi di `M` città sedi di Pronto Soccorso, mediante un algoritmo basato su combinazioni semplici. Si scriva la funzione `checkSedi` che, quando viene chiamata nel caso terminale, sia in grado di verificare l'accettabilità di una soluzione. Questa deve ricevere, tra gli altri parametri da decidere, la matrice delle distanze, la distanza massima `MAXD`, il numero minimo `MINS` e l'insieme ottenuto nel caso terminale (la soluzione, di tipo `SEDI`, da verificare).

N.B: si chiede solamente di implementare la funzione `checkSedi`, non l'algoritmo per generare le combinazioni.

Problema di ricerca e ottimizzazione

Dato un insieme valido di `M` sedi (tipo `SEDI`) di Pronto Soccorso, già verificato mediante la funzione `checkSedi`, si vuole determinare un partizionamento ottimo delle città servite, in modo tale che si rispettino i criteri seguenti:

- ogni città non sede di Pronto Soccorso viene assegnata a una sola sede di Pronto Soccorso a `distanza ≤ MAXD`;
- la distanza media tra le città servite e le sedi di Pronto Soccorso è minima;
- a ognuna delle sedi di Pronto Soccorso si assegnano almeno `MINS` città.

Si scriva la funzione `bestPart`, che, mediante l'uso di un algoritmo di partizionamento, trovi la soluzione ottima e la ritorni come struttura dati (tipo `SERVIZI`). La funzione deve ricevere come parametri l'elenco delle città, la matrice delle distanze, un insieme di `M` sedi di Pronto Soccorso e i valori `MAXD` e `MINS`.

Non è necessario realizzare la funzione wrapper; è sufficiente implementare la funzione `bestPart`, la quale al suo interno deve usare le seguenti funzioni (anch'esse da implementare):

- La funzione di verifica `checkPart` da chiamare nel caso terminale;
- La funzione `prunePart` da chiamare per fare pruning.

N.B: si dica esplicitamente di quali vincoli o criteri si può fare pruning e di quali invece no.

Esempio

Supponendo che l'elenco delle città e la matrice delle distanze siano quelle proposte nell'esempio precedente e ipotizzando che:

- si siano selezionate come sedi di Pronto Soccorso le città `{A,C}`;
- `MAXD` valga 4;
- `MINS` valga 2.

Si riporta qui di seguito la matrice delle distanze, nella quale sono state evidenziate in rosso le sedi di Pronto Soccorso (sulle righe) e in grassetto le distanze compatibili con `MAXD`:

	A	**B**	**C**	**D**	**E**	**F**
A	0	**2**	5	**4**	**2**	6
B	2	0	6	6	2	5
C	5	6	0	7	**4**	**3**
D	4	6	7	0	6	3
E	2	2	4	6	0	6
F	6	5	3	3	6	0

In tal caso, la soluzione ottima consiste nella seguente assegnazione:

```
sede A = {B, D}
sede C = {E, F}
```

e ha distanza media tra le città servite e le sedi pari a `(2+4+4+3)/4 = 13/4`.