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

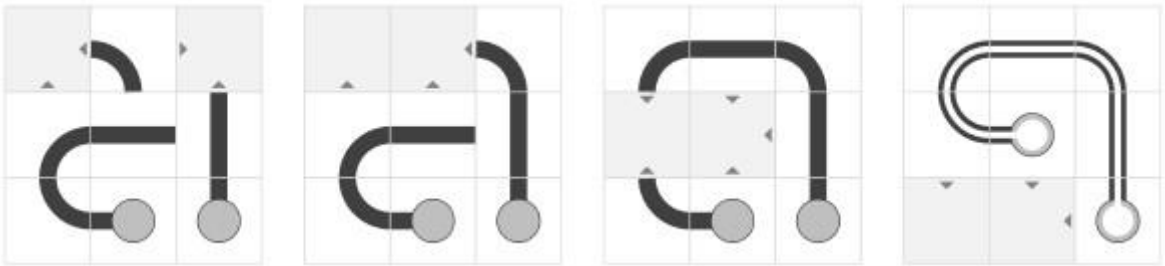
Appello del 30/06/2023 - Prova di programmazione (18 punti)

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.