Esercitazione di laboratorio n. 9

Esercizio n.1: Sequenza di attività
Competenze: esplorazione dello spazio delle soluzioni con i modelli del Calcolo Combinatorio (Ricorsione e problem-solving: 3.2.4, 3.3.6), problemi di ottimizzazione (Ricorsione e problem-solving: 3.5)
Traccia da 12 punti, esercizio da 6 punti, appello del 29/01/2018

Un'attività i è caratterizzata da un intervallo aperto (s_i, f_i), dove s_i è il tempo di inizio, f_i è un tempo di fine e $d_i=f_i-s_i$ è la durata dell'attività. Tempo di inizio, di fine e durata sono interi. Una collezione di attività S è memorizzata in un vettore v di strutture `att` aventi come campi tempo di inizio e tempo di fine. Si suppongano corretti i dati contenuti nel vettore ($\forall i\ s_i \leq f_i$). Due attività i e j sono incompatibili se e solo se si intersecano o sovrappongono:

$$(s_i < f_j) \&\& (s_j < f_i) \Leftrightarrow i \cap j \neq \emptyset$$

Si scrivano la funzione wrapper e una funzione ricorsiva in C in grado di determinare e visualizzare un sottoinsieme di attività compatibili che massimizza la somma delle durate:

$$max \sum_{k \in S} d_k \ \&\& \ \forall (k_1, k_2) \in S, k_1 \cap k_2 = \emptyset$$

Il prototipo della funzione wrapper sia:

```
void attSel(int N, att *v);
```

Esempio:
se S =*((1,2), (2,4), (2,5), (3,5), (5,7), (6,8))*, uno dei sottoinsiemi di S di attività compatibili che massimizza la somma delle durate è *(1,2), (2,5), (6,8)* per una somma delle durate pari a 6.

Esercizio n.2: **Tessere e scacchiere**
Competenze: esplorazione dello spazio delle soluzioni con i modelli del Calcolo Combinatorio (Ricorsione e problem-solving: 3.2.4, 3.3.6), problemi di ottimizzazione (Ricorsione e problem-solving: 3.5)
Traccia da 18 punti, appello del 21/06/2018

Un gioco si svolge su una scacchiera rettangolare di `RxC` caselle. In ogni casella deve essere posizionata una tessera su cui sono disegnati due segmenti di tubo, uno in orizzontale e uno in verticale. Le diagonali non sono contemplate. Ogni segmento è caratterizzato da un colore e da un punteggio (valore intero positivo).

Per ottenere i punti associati ai vari segmenti, è necessario allineare lungo un'intera riga (colonna) tubi in orizzontale (verticale) dello stesso colore. Quindi solo righe (colonne) complete di tubi orizzontali (verticali) contribuiscono al punteggio. Le tessere possono essere ruotate di 90°. Le tessere sono disponibili in copia singola. Si assuma esistano abbastanza tessere per completare la scacchiera.

Lo scopo del gioco è ottenere il massimo punteggio possibile posizionando una tessera in ogni cella della scacchiera.

Su un primo file di testo (`tiles.txt`) è riportato l'elenco delle tessere disponibili nel seguente formato:

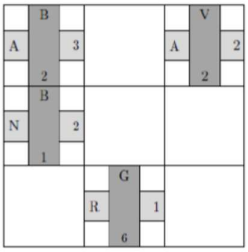
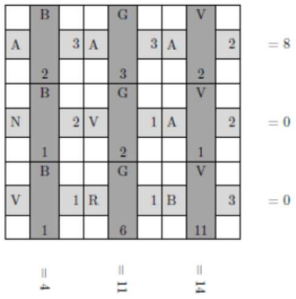
- sulla prima riga è presente il numero `T` di tessere
- seguono `T` quadruple nella forma `<coloreT1><valoreT1><coloreT2><valoreT2>` a descrivere la coppia di tubi presenti sulla tessera in termini di rispettivo colore e valore.

Si assuma che a ogni tessera sia associato un identificativo numerico nell'intervallo 0..`T`-1
Su un secondo file di testo (`board.txt`) è riportata una configurazione iniziale per la scacchiera di gioco. Il file ha il seguente formato:

- sulla prima riga è presente una coppia di interi `R C` a rappresentare il numero di righe e colonne della superficie di gioco
- seguono `R` righe riportanti `C` elementi ciascuna a definire la configurazione di ogni cella
- ogni cella è descritta da una coppia t_i/r dove t_i è l'indice di una tessera tra quelle presenti in `tiles.txt` e r rappresenta l'eventuale sua rotazione (es: 7/0 oppure 3/1 per rappresentare rispettivamente una tessera non ruotata e una ruotata). Una cella vuota è rappresentata dalla coppia -1/-1.

Si scriva un programma in C che, una volta acquisiti in opportune strutture dati l'elenco delle tessere disponibili e la configurazione iniziale della scacchiera generi la soluzione a punteggio massimo possibili a partire dalla configurazione iniziale letta da file.

Esempio:

tiles.txt	board.txt	scacchiera iniziale	scacchiera finale
9 A 3 B 2 A 2 V 1 A 2 V 2 B 1 N 2 A 3 G 3 V 1 G 2 R 1 G 6 V 1 B 1 V 11 B 3	3 3 0/0 -1/-1 2/0 3/1 -1/-1 -1/-1 -1/-1 6/0 -1/-1		

Esercizio n.3: **Gioco di ruolo (multifile)**
Competenze: Vettori di `struct`, strutture dati dinamiche, vettori dinamici, riallocazione dinamica (Puntatori e strutture dati dinamiche 2.5.5, 3.3.3, 3.2.3). Programmazione multifile(Puntatori e strutture dati dinamiche 5.5)

Si consideri lo scenario introdotto nell'esercizio n. 3 del laboratorio 7 (Gioco di ruolo). Si organizzi il codice scritto precedentemente suddividendolo in più moduli, quali:

- un modulo client contenente il main e l'interfaccia utente/menu
- un modulo per la gestione dei personaggi
- un modulo per la gestione dell'inventario.

Il modulo per i personaggi deve fornire le funzionalità di:

- acquisizione da file delle informazioni dei personaggi, mantenendo la medesima struttura a lista richiesta nel laboratorio precedente
- inserimento/cancellazione di un personaggio
- ricerca per codice di un personaggio
- stampa dei dettagli di un personaggio e del relativo equipaggiamento, se presente
- modifica dell'equipaggiamento di un personaggio
    - aggiunta/rimozione di un oggetto.

Il modulo dell'inventario deve fornire le funzionalità di:

- acquisizione da file delle informazioni relative agli oggetti disponibili, mantenendo la medesima struttura a vettore richiesta nel laboratorio precedente
- ricerca di un oggetto per nome
- stampa dei dettagli di un oggetto.

NB: il modulo personaggi è client del modulo inventario, in quanto ogni personaggio ha una collezione di (riferimenti a) dati contenuti nell'inventario.

Valutazione: gli esercizi 2 e 3 saranno oggetto di valutazione
Scadenza: caricamento di quanto valutato: entro le 23:59 del 18/12/2020.