Esercitazione di laboratorio n. 10

Esercizio n. 1: Collane e pietre preziose (versione 3)

Si risolva l'esercizio n. 1 del laboratorio 7 mediante il paradigma della memoization. Si richiede soltanto di calcolare la lunghezza massima della collana compatibile con le gemme a disposizione e con le regole di composizione.

Suggerimento: affrontare il problema con la tecnica del divide et impera, osservando che una collana di lunghezza P può essere definita ricorsivamente come:

- la collana vuota, formata da P=0 gemme
- una gemma seguita da una collana di P-1 gemme.

Poiché vi sono 4 tipi di gemme Z, R, T, S, si scrivano 4 funzioni `fZ`, `fR`, `fT`, `fS` che calcolino la lunghezza della collana più lunga iniziante rispettivamente con uno zaffiro, un rubino, un topazio e uno smeraldo avendo a disposizione `z` zaffiri, `r` rubini, `t` topazi e `s` smeraldi. Note le regole di composizione delle collane, è possibile esprimere ricorsivamente il valore di una certa funzione `fX` sulla base dei valori delle altre funzioni.
Si presti attenzione a definire adeguatamente i casi terminali di tali funzioni, onde evitare di ricorre in porzioni non ammissibili dello spazio degli stati.
Si ricorda che il paradigma della memoization prevede di memorizzare le soluzioni dei sottoproblemi già risolti in opportune strutture dati da progettare e dimensionare e di riusare dette soluzioni qualora si incontrino di nuovo gli stessi sottoproblemi, limitando l'uso della ricorsione alla soluzione dei sottoproblemi non ancora risolti.

Esercizio n. 2: Corpo libero

Il corpo libero è una specialità della ginnastica artistica/acrobatica in cui l'atleta deve eseguire una sequenza di elementi senza l'ausilio di attrezzi, ad eccezione della pedana di gara.
In un programma di corpo libero il ginnasta è tenuto ad eseguire una serie di passaggi acrobatici, detti diagonali.
Ogni diagonale è composta da uno o più elementi.
Ogni elemento è descritto da una serie di parametri:

- nome dell'elemento (una stringa di massimo 100 caratteri senza spazi)
- tipologia: l'elemento può essere un elemento acrobatico avanti [2], acrobatico indietro [1] o di transizione [0]
- direzione di ingresso: il ginnasta può entrare in un elemento frontalmente [1] o di spalle [0]
- direzione di uscita: il ginnasta può uscire da un elemento frontalmente [1] o di spalle [0]
- requisito di precedenza: l'elemento può essere eseguito come primo di una sequenza [0] o deve essere preceduto da almeno un altro elemento [1]
- finale: l'elemento non può essere seguito da altri elementi [1] o meno [0]
- valore: il punteggio ottenuto dall'atleta per la corretta esecuzione di un elemento (reale)
- difficoltà: la difficoltà di esecuzione dell'elemento (intero).

Gli elementi sono memorizzati in un file di testo (`elementi.txt`) in ragione di uno per riga. Il numero di elementi è riportato sulla prima riga del file.

Perché due elementi possano essere eseguiti in sequenza, la direzione di uscita del primo elemento deve coincidere con la direzione di ingresso del secondo elemento. Un ginnasta inizia una diagonale sempre frontalmente. La difficoltà di una diagonale è definita come la somma delle difficoltà degli elementi che la compongono. La difficoltà del programma di gara è data dalla somma delle difficoltà delle diagonali che lo compongono.

Ai fini dell'esercizio, si considerino le seguenti regole:

- il ginnasta deve presentare 3 diagonali
- il ginnasta deve includere almeno un elemento acrobatico in ogni diagonale
- il ginnasta deve includere almeno un elemento acrobatico avanti e almeno un elemento acrobatico indietro nel corso del suo programma, ma non necessariamente nella stessa diagonale
- il ginnasta deve presentare almeno una diagonale in cui compaiono almeno due elementi acrobatici in sequenza
- se il ginnasta include un elemento finale di difficoltà 8 o superiore nell'ultima diagonale presentata in gara, il punteggio complessivo della diagonale viene moltiplicato per 1.5
- ogni diagonale può contenere al massimo 5 elementi
- ogni diagonale non può avere difficoltà superiore a un dato valore DD
- il programma complessivamente non può avere difficoltà superiore a un dato valore DP.

Si scriva un programma in C in grado di identificare la sequenza di diagonali che permettono al ginnasta di ottenere il punteggio più alto possibile, dato un set di elementi disponibili e il valore dei due parametri DD e DP.

Esercizio n. 3: Corpo libero (vers. greedy)
A partire dallo scenario introdotto nell'esercizio precedente, si risolva il problema proponendo uno o più algoritmi greedy, definendo opportune funzioni obiettivo.

Esercizio n. 4: Rete di elaboratori
Un grafo non orientato e pesato rappresenta una rete di elaboratori appartenenti ciascuno ad una sottorete. Il peso associato ad ogni arco rappresenta il flusso di dati tra due elaboratori della stessa sottorete o di sottoreti diverse, come nell'esempio seguente (cfr. figura successiva).

Il grafo è contenuto in un file, il cui nome è passato come argomento sulla linea di comando. Il file è composto da un numero indefinito di righe ciascuna delle quali contiene una quaterna di stringhe alfanumeriche, di al massimo 30 caratteri, e un intero:

```
<id_elab1> <id_rete1> <id_elab2> <id_rete2> <flusso>
```

Si facciano anche le seguenti assunzioni:

- i nomi dei singoli nodi sono univoci all'interno del grafo
- non sono ammessi cappi
- tra due nodi c'è al massimo un arco (non è un multigrafo)
- le sotto-reti sono sotto-grafi non necessariamente connessi.

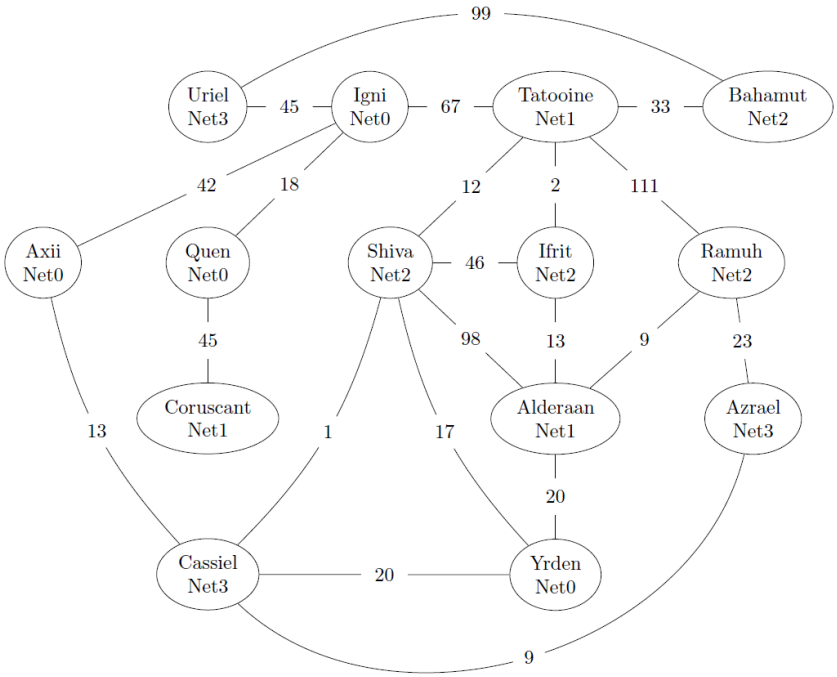
Si scriva un programma in C in grado di caricare in memoria il grafo, leggendone i contenuti da file e di potervi effettuare alcune semplici operazioni.
La rappresentazione della struttura dati in memoria deve essere fatta tenendo conto dei seguenti vincoli:

- il grafo sia implementato come ADT di I classe, predisposto in modo tale da poter contenere sia la matrice sia le liste di adiacenza. Nella fase di caricamento dei dati da file si generi solamente la matrice di adiacenza, su comando esplicito va generata anche la lista di adiacenza
- si utilizzi una tabella di simboli tale da fornire corrispondenze "*da nome a indice*" e "*da indice a nome*".

Sul grafo, una volta acquisito da file, sia possibile:

- elencare in ordine alfabetico i vertici e per ogni vertice gli archi che su di esso insistono, sempre in ordine alfabetico
- dati 3 vertici i cui nomi sono letti da tastiera, verificare se essi sono adiacenti a coppie, cioè se formano un sottografo completo. Tale funzione sia implementata sia per la rappresentazione con matrice delle adiacenze, sia per la rappresentazione con lista delle adiacenze
- generare la rappresentazione a lista di adiacenza, **SENZA** leggere nuovamente il file, a partire da quella a matrice di adiacenza.

In allegato al testo è presente il grafo d'esempio (`grafo.txt`) rappresentato a seguire:



Valutazione: gli esercizi 1, 3 e 4 saranno oggetto di valutazione
Scadenza: caricamento di quanto valutato: entro le 23:59 del 24/01/2020