Esercitazione di laboratorio n. 12

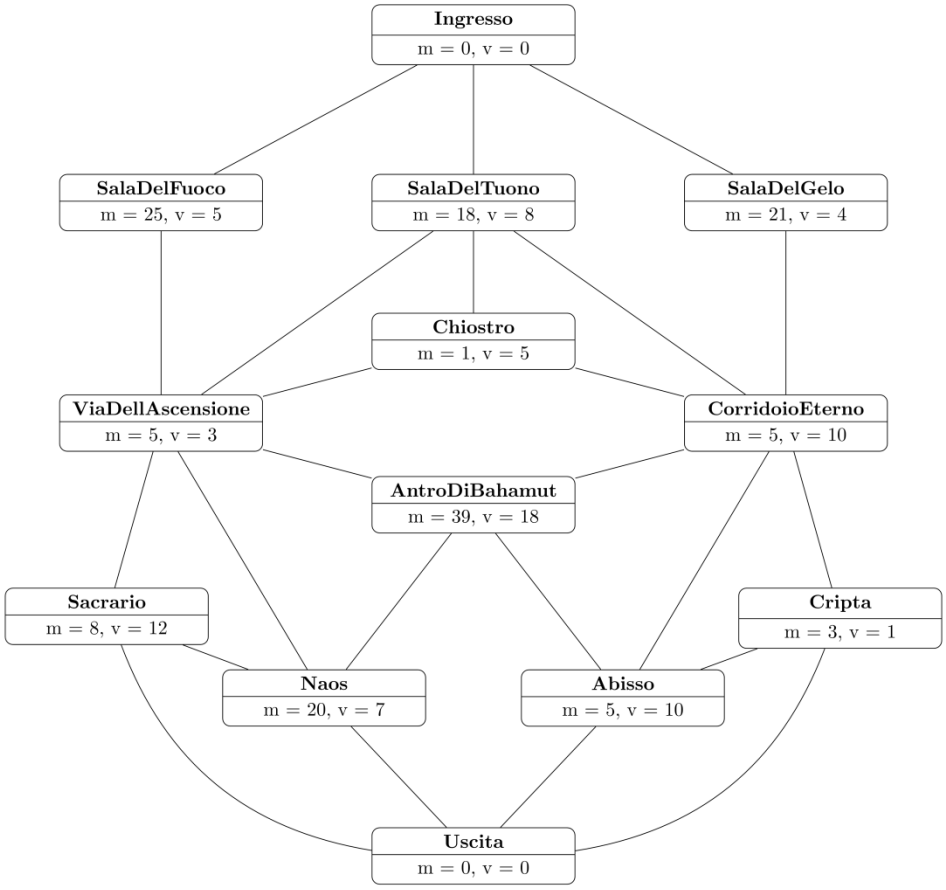
(Caricamento sul portale entro le 23.59 del 30/01/2017 dell'esercizio 1 e di almeno uno tra gli esercizi 2 e 3)

Esercizio n.1: Dungeon crawler

L'avventuriero del *lab10* è pronto per la sua missione: l'esplorazione di un antico tempio. Il tempio è costituito da diverse sale, ognuna delle quali è caratterizzata dalla presenza di un tesoro e di una creatura a difenderlo. La descrizione del tempio è contenuta in un file `tempio.txt` organizzato come segue:

- sulla prima riga appare il numero `S` di sale
- seguono `S` terne, in ragione di una per riga, nella forma `<nome_sala> <monete_oro> <ferite>`
- per convenzione l'ingresso e l'uscita siano sempre le sale i cui nomi sono fatti corrispondere dalla tabella di simboli agli indici `0` e `S-1`, rispettivamente
- seguono, in numero indefinito, coppie di sale `<nome_sala_1> <nome_sala_2>` a rappresentare le connessioni tra le stesse.

La figura seguente riporta la struttura del tempio associata al file di ingresso d'esempio allegato al testo del laboratorio.



Le regole di esplorazione sono le seguenti:

- esiste un unico punto di ingresso nel tempio

- esiste un unico punto di uscita dal tempio (diverso dall'ingresso)
- l'avventuriero inizia la visita con `V` punti ferita
- l'avventuriero, entrato in una sala `s`, usa v_s dei suoi punti ferita residui per sconfiggere la creatura e appropriarsi delle m_s monete d'oro contenute nella stanza
- non è possibile evitare lo scontro in una sala una volta entrati
- non è possibile rientrare in una sala più di una volta
- l'avventuriero deve raggiungere l'uscita vivo (ossia con almeno 1 punto ferita rimanente). Giunto all'uscita l'esplorazione termina immediatamente.

Si scriva un programma in C che, una volta letta la struttura del tempio da file, individui la sequenza di sale da attraversare perché la resa dell'esplorazione, in termini di monete d'oro raccolte, sia massima.

Esercizio n.2: Amici in viaggio
Un gruppo di amici deve organizzarsi per suddividere gli `N` partecipanti a un viaggio tra le `K` auto disponibili. Si assuma che tutte le auto siano uguali, e che ognuna possa accogliere al più `M` occupanti. Si garantisce che ci siano abbastanza posti per non dover lasciare a piedi nessuno.
Per ogni coppia di amici, inoltre, è noto un indice di reciproco gradimento, espresso mediante un intero nell'intervallo `[-5..5]`.
La sommatoria del gradimento reciproco per tutte le coppie di amici in un'auto definisce il livello di umore per l'auto stessa.
L'obiettivo finale è assegnare un posto a ogni amico tale per cui l'umore globale della comitiva, definito come la media dell'umore di tutte le auto, sia maggiore di un intero `U`.

Si scriva un programma in linguaggio C che, una volta acquisite le informazioni necessarie dal file `amici.txt`, produca la soluzione cercata.

Il file `amici.txt` è organizzato come segue:

- sulla prima riga è riportata la quaterna di interi `N K M U`
- seguono `N` righe composte da `N` interi ciascuna, a rappresentare l'indice di reciproco gradimento tra due ospiti. La matrice risultante è ovviamente simmetrica.

Esercizio n.3: Amici (ancora) in viaggio
Si consideri la medesima situazione introdotta dall'esercizio n.2, in cui però cambia l'obiettivo.
L'obiettivo finale, in questa variante, è assegnare un posto a ogni amico massimizzando l'umore globale della comitiva, definito come la media dell'umore di tutte le auto.
Per questa variante si richiede di prestare attenzione a **NON** generare soluzioni simmetriche.