Programmazione Funzionale – Febbraio 2020

Nota: è indispensabile specificare il tipo e dare una descrizione dichiarativa di ogni funzione ausiliaria utilizzata (anche locale), altrimenti non verrà presa in considerazione (ad eccezione delle funzioni il cui tipo e specifica sono già dati nel testo).

Una rete metropolitana è costituita da stazioni e linee di collegamento. Due stazioni adiacenti possono essere collegate da più di una linea, le linee possono anche essere circolari e una stazione può essere collegata a più di una stazione mediante una stessa linea.

Se si rappresentano le stazioni mediante interi e le linee mediante stringhe (che si assumono diverse dalla stringa vuota), una rete metropolitana si può rappresentare mediante un grafo non orientato etichettato negli archi, utilizzando il tipo `metro` così definito:

`type metro = (int × int × string) list`

dove, in ogni tripla della lista, i primi due elementi rappresentano le stazioni collegate direttamente dalla linea rappresentata dal terzo elemento della tripla (sono cioè nodi adiacenti nel grafo, collegati da un arco etichettato dal nome della linea).

1. Definire una funzione `line: metro → string → int list`, tale che `line m ln` riporti (in qualsiasi ordine, ma senza ripetizioni) una lista con tutte le stazioni per le quali passa la linea `ln`.

2. Definire una funzione `vicini: int → metro → (int × string) list`, tale che `vicini stazione m` riporti la lista delle coppie `(s,ln)` tali che `s` è una stazione direttamente collegata (adiacente) a `stazione` mediante la linea `ln` nella rete `m`;

3. Definire una funzione

   `raggiungi: metro → int → int → int → int list`

   tale che `raggiungi m maxc start goal` riporti una lista di stazioni rappresentante un percorso, nella rete `m`, dalla stazione `start` alla stazione `goal` nel quale si cambi treno (linea) al massimo `maxc` volte. La funzione solleverà un'eccezione se un tale percorso non esiste.

   Ad esempio, se la rete è rappresentata dalla lista `m = [(1,2,"A"); (2,3,"A"); (3,1,"A"); (2,4,"B"); (4,5,"B"); (4,6,"C"); (6,3,"C"); (5,7,"D"); (6,7,"D")]`, `raggiungi m 2 6 1` può fornire come soluzione il percorso [6; 3; 2; 1] (con un cambio) o il percorso [6; 4; 2; 1] (con 2 cambi). Mentre `raggiungi m 1 1 7` solleverà un'eccezione (non si può arrivare da 1 a 7 con meno di 2 cambi).

   Suggerimento: adattare l'algoritmo di ricerca di un cammino in un grafo, aggiungendo due parametri alle funzioni ausiliarie: un intero `cambi` che rappresenti il numero di cambi già effettuati, e una stringa `linea` che rappresenti la linea con la quale si è arrivati alla stazione corrente `s`. Quando si esamina una stazione `s'` collegata dalla linea `ln` alla precedente stazione `s`, il numero di cambi viene incrementato se `ln` è diverso da `linea`. Partendo da `start`, il parametro `linea` sarà inizializzato con la stringa vuota e `cambi` con `-1` (infatti, passando da `start` a una stazione adiacente `s`, la stringa vuota sarà diversa dalla linea che collega `start` con `s` e il parametro `cambi` diventerà 0).