Corso di Algoritmi e Strutture Dati
Esercitazione 7: cammini su grafi

Si consideri un navigatore satellitare, usato da un commesso viaggiatore per pianificare viaggi tra le città in cui opera i propri commerci. Tra le tante funzioni che il navigatore deve offrire all'utente c'è anche quella di ricercare e suggerire un percorso che, da una qualunque città di partenza, conduca ad un'altra città di arrivo. Normalmente il percorso da cercare sarebbe quello di lunghezza minima, ma per semplicità ci limitiamo ad un percorso qualsiasi purchè sia *aciclico*, ossia tale per cui una stessa località venga visitata al più una volta.

Il navigatore deve poter caricare, da file o da standard input, le mappe delle regioni in cui l'utente si muoverà. A titolo di esempio si consideri la mappa della Romania rappresentata in Figura 1.

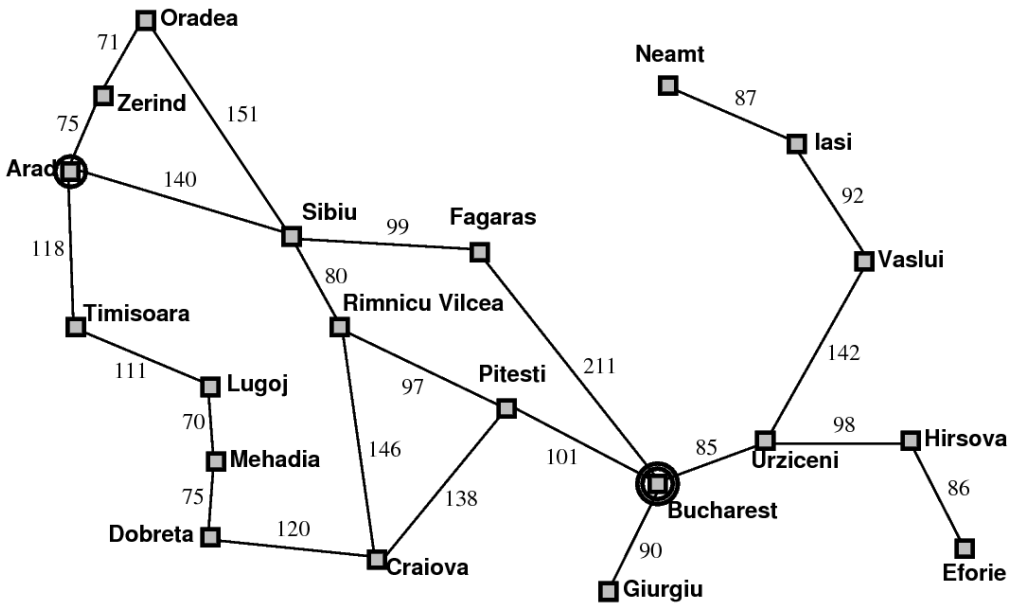

Figure 1: Principali città della Romania e collegamenti tra esse.

Tale mappa può essere rappresentata in formato testo nel seguente modo:

```
arad zerind 75
arad timisoara 118
sibiu arad 140
zerind oradea 71
oradea sibiu 151
timisoara lugoj 111
lugoj mehadia 70
mehadia dobreta 75
```

```
dobreta craiova 120
craiova rimnicuvilcea 146
craiova pitesti 138
rimnicuvilcea pitesti 97
rimnicuvilcea sibiu 80
sibiu fagaras 99
fagaras bucharest 211
pitesti bucharest 101
bucharest giurgiu 90
bucharest urziceni 85
urziceni hirsova 98
hirsova eforie 86
urziceni vaslui 142
vaslui iasi 92
iasi neamt 87
0
```

Si noti lo "0" in fondo, utilizzato per terminare la sequenza di lettura.

Scopo di questa esercitazione è implementare le strutture dati e gli algoritmi necessari al navigatore satellitare per risolvere adeguatamente il problema descritto sopra. L'idea di fondo è che una mappa stradale si può rappresentare come grafo, supponendo che le città siano i vertici e le strade siano gli archi. I vertici risultano etichettati con i nomi delle città. Gli archi, per semplicità, sono non orientati (in pratica si suppone che le strade non abbiano sensi unici). Ciascun arco riceve un peso uguale alla lunghezza in chilometri della relativa tratta stradale.

Si implementi dunque, come struct C++ e rispettando il principio di incapsulamento, il tipo di dato `Grafo non orientato con vertici etichettati e archi pesati`. L'implementazione deve sfruttare l'approccio a **liste di adiacenza**. Le operazioni, da realizzare come metodi della struct, sono almeno (ma non solo) le seguenti:

- Creazione di un *grafo vuoto*.
- *Inserimento di un nuovo vertice etichettato.*
  L'etichetta da attribuire al nuovo vertice è una stringa passata come parametro al metodo (sarebbe il nome di una città). Il metodo restituisce `false` se l'etichetta è già presente nel grafo, e in tal caso nessun nuovo vertice deve essere inserito.
- *Inserimento di un nuovo arco pesato* tra due vertici esistenti e distinti.
  I due vertici sono identificati dalle loro etichette, passate come parametri al metodo assieme al peso del nuovo arco (un numero, che nel caso specifico rappresenta la distanza in chilometri tra le due città). Il metodo restituisce `false` se i due vertici sono già collegati da un arco, oppure se uno dei vertici non è presente nel grafo, oppure se i due vertici coincidono; in tali casi nessun nuovo arco deve essere inserito. È ammesso che due archi distinti possano portare uguale peso, per cui nessun controllo di unicità del peso dell'arco è richiesto.
- *Cammino aciclico tra due vertici*:
  questo metodo prende come parametri le due etichette di due vertici A e B del grafo, e crea una lista che rappresenta un possibile cammino aciclico che conduca da A a B. Se non esiste alcun cammino tra A e B, allora il metodo crea una lista vuota (rappresenta un cammino vuoto).
  Il metodo, oltre a restituire la lista creata, restituisce anche la lunghezza complessiva del cammino.

È permesso utilizzare la libreria standard del C++, ad eccezione della STL (Standard Template Library). Non è permesso ricorrere ad altre librerie.

La struct C++ implementata deve essere impiegate in un semplice programma C++. Tale programma deve svolgere, nell'ordine, le azioni seguenti:

1. Creazione del grafo acquisendo da standard input le informazioni sui vertici e sugli archi, espresse nel formato seguente:

   ```
   origine1 destinazione1 dist1
   origine2 destinazione2 dist2
   ....
   origineN destinazioneN distN
   0
   ```

2. Visualizzazione testuale (non grafica!) del grafo, che mostri l'elenco dei vertici con le rispettive etichette e, per ciascun vertice, la relativa lista di adiacenza con i pesi degli archi (utile a capire se il grafo è stato creato in maniera corretta).

3. Ricerca di un cammino aciclico tra `origine` e `destinazione` e visualizzazione su standard output del cammino trovato, nel formato `origine vertice1 vertice2 .... verticeN destinazione` e della lunghezza complessiva.