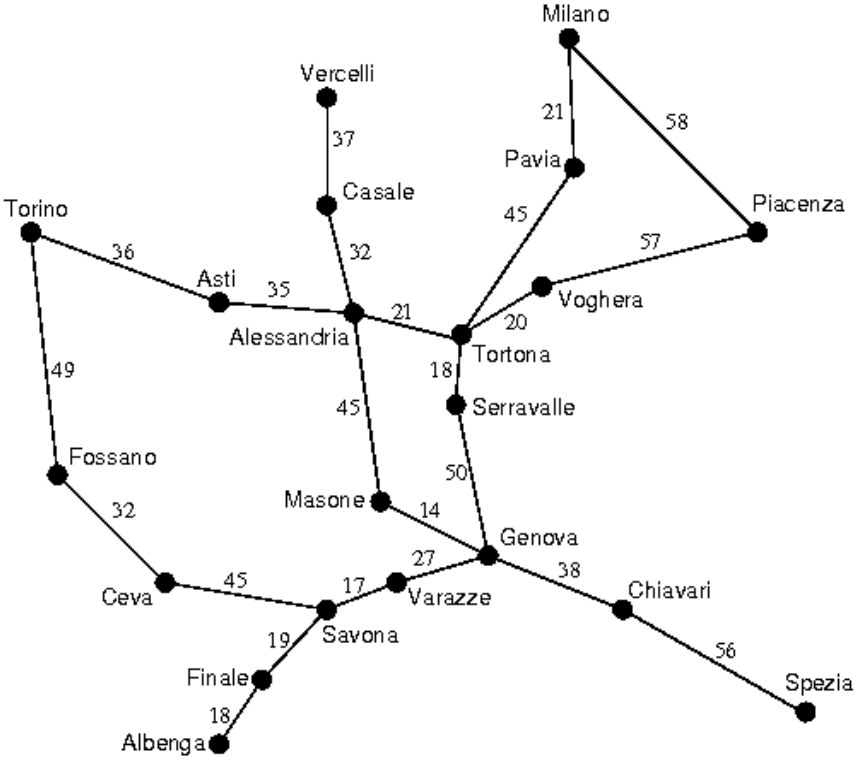
Corso di Algoritmi e Strutture Dati
Laboratorio 8: Grafo non orientato

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 rappresentata sotto:



Tale mappa può essere rappresentata in formato testo come una lista che elenca i segmenti stradali fornendo per ciascuno le due città estreme e la lunghezza in km. Per l'esempio in figura abbiamo:

```
Torino Asti 36
Asti Alessandria 35
Torino Fossano 49
Fossano Ceva 32
Ceva Savona 45
Albenga Finale 18
Savona Finale 19
Varazze Savona 17
```

```
Genova Varazze 27
Casale Vercelli 37
Casale Alessandria 32
Alessandria Masone 45
Masone Genova 14
Genova Serravalle 50
Serravalle Tortona 18
Tortona Alessandria 21
Genova Chiavari 38
Chiavari Spezia 56
Tortona Voghera 20
Piacenza Voghera 57
Piacenza Milano 58
Tortona Pavia 45
Pavia Milano 21
0
```

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

Scopo di questo laboratorio 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.

In questo laboratorio, si richiede di implementare il tipo di dato `Grafo non orientato con vertici etichettati e archi pesati`. L'implementazione deve sfruttare l'approccio a **liste di adiacenza**. I prototipi delle funzioni da implementare sono presenti nel file *graph.h*.

Il grafo implementato al punto precedente sarà utilizzato in un semplice programma C++ che deve permettere di svolgere, a scelta dell'utente, 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. Creazione del grafo acquisendo da file le informazioni sui vertici e sugli archi (il formato è lo stesso come per l'acquisizione da standard input)

3. 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).

4. Inserimento di un nuovo vertice nel grafo, la cui etichetta viene fornita da input.

5. Inserimento di un nuovo arco nel grafo, assumendo che le etichette dei suoi vertici vengano fornite in input.
6. Determinazione del numero dei vertici.
7. Determinazione del numero degli archi.
8. Determinazione del grado di un vertice, la cui etichetta viene fornita da input.
9. Verifica adiacenza tra due vertici, le cui etichette vengono fornite in input.
10. Visualizzazione della lista di adiacenza associata ad un vertice, la cui etichetta viene fornita in input.
11. 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.
12. Facoltativo: scrivere la funzione che svuota il grafo; l'implementazione già fornita ha un *memory leak*, perché la memoria allocata non viene svuotata quando il grafo viene reinizializzato.

I file di test non verificano che i percorsi restituiti siano corretti, quindi sta a voi testare la cosa.

Per fare le verifiche iniziali, è fornito un esempio di mappa più piccola:

```
Asti Alessandria 35
Varazze Savona 17
Genova Varazze 27
Alessandria Masone 45
Masone Genova 14
Genova Serravalle 50
Serravalle Tortona 18
Tortona Alessandria 21
Genova Chiavari 38
Tortona Voghera 20
0
```