LABORATORIO 8
NAVIGATORE SATELLITARE

Algoritmi e strutture dati

NAVIGATORE SATELLITARE

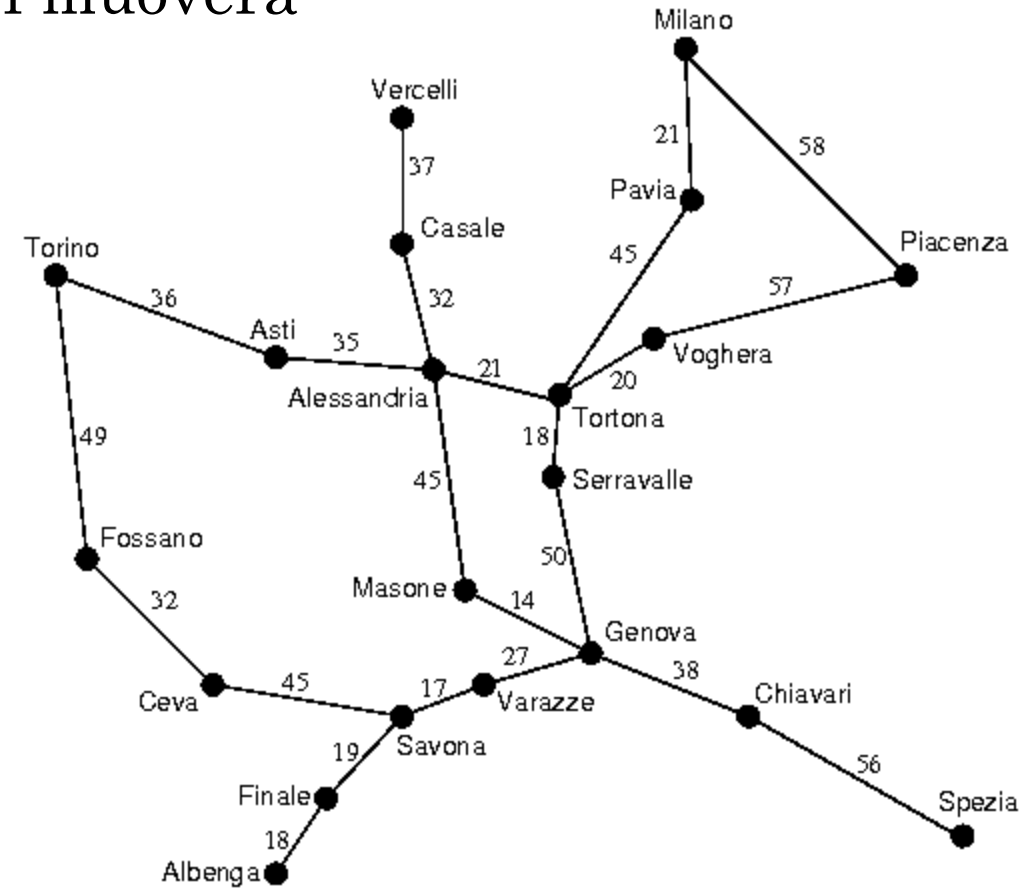
- Consideriamo un navigatore satellitare

RICERCA DI UN PERCORSO (CAMMINO)

- 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 localita venga visitata al piu una volta

MAPPE

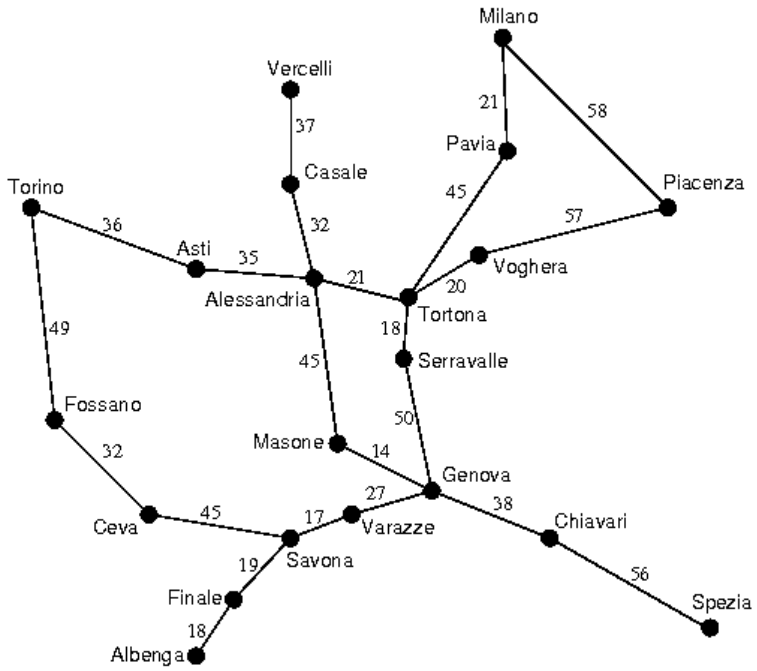
- Il navigatore **deve poter caricare, da file o da standard input, le mappe delle regioni** in cui l'utente si muoverà



FORMATO MAPPA

- Tale mappa è rappresentata in formato testo come una lista che elenca i segmenti stradali fornendo per ciascuno le due città e la lunghezza in km

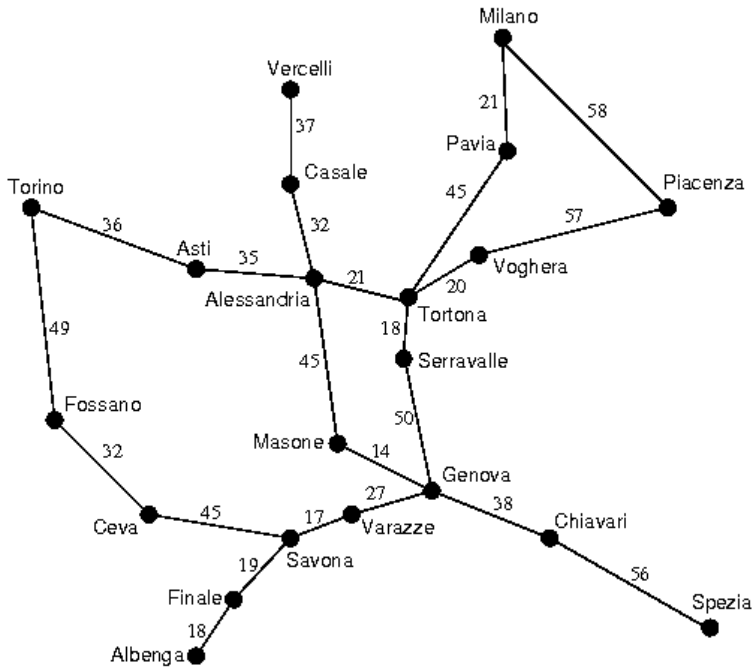
...
Tortona Alessandria 21
Genova Chiavari 38
Chiavari Spezia 56
Tortona Voghera 20
Piacenza Voghera 57
Piacenza Milano 58
Tortona Pavia 45
Pavia Milano 21
0



FORMATO MAPPA

- Tale mappa è rappresentata in formato testo come una lista che elenca i segmenti stradali fornendo per ciascuno le due città e la lunghezza in km

...
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



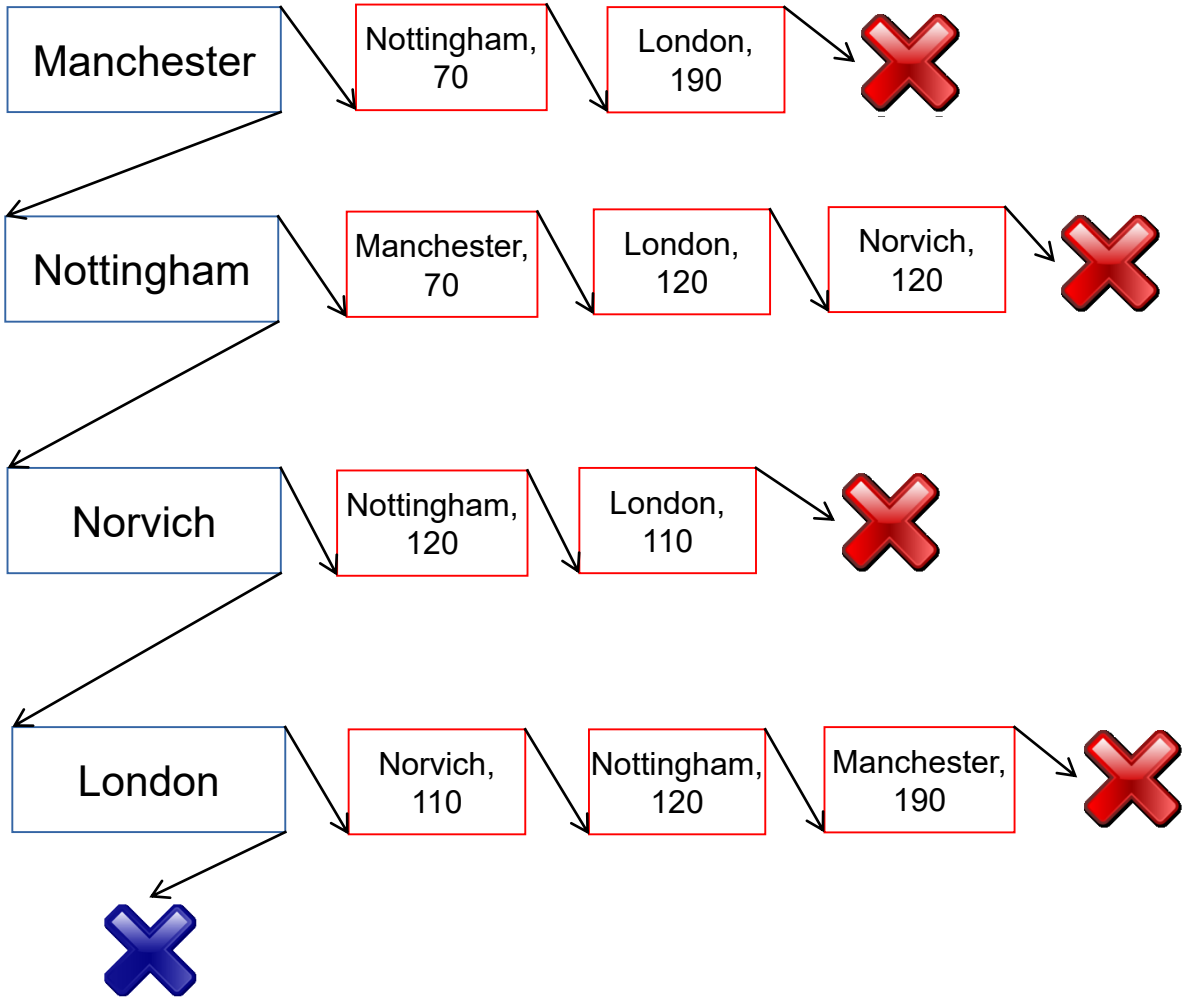
Si noti lo "0" in fondo, utilizzato per terminare la sequenza di lettura

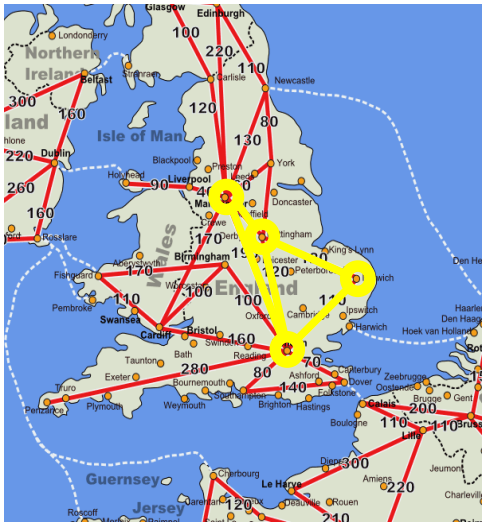
LABORATORIO 8

- Scopo di questo laboratorio è implementare le strutture dati e gli algoritmi necessari per risolvere il problema descritto sopra
- L'idea di fondo è che una **mappa stradale si può rappresentare come grafo**
  - città sono i vertici e le strade sono gli archi
- I vertici risultano etichettati con i nomi delle citta. Gli archi, per semplicità, sono non orientati
  - Cioè le strade non hanno sensi unici
- Ciascun arco riceve un peso uguale alla lunghezza in chilometri della relativa tratta stradale

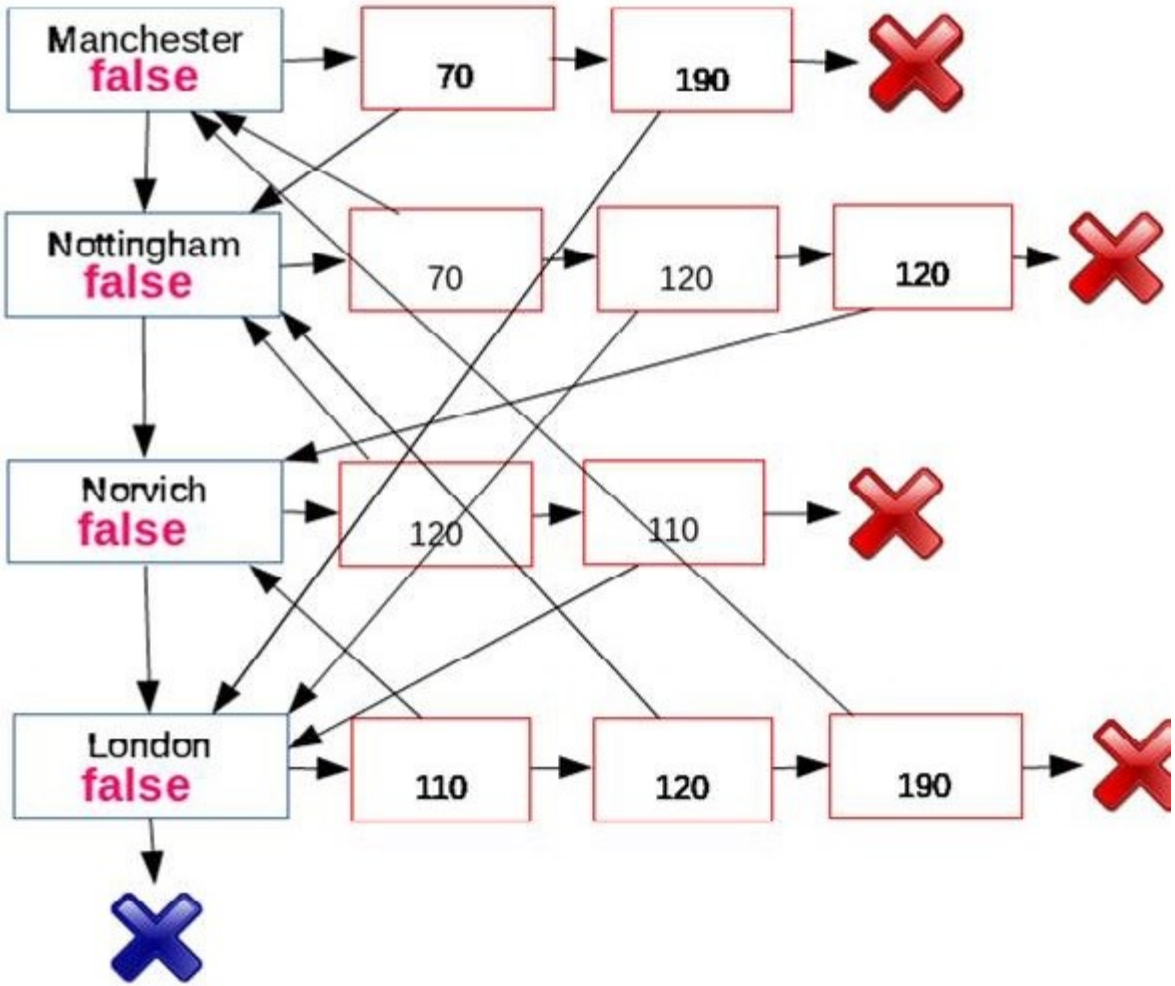
IMPLEMENTAZIONE DEL GRAFO

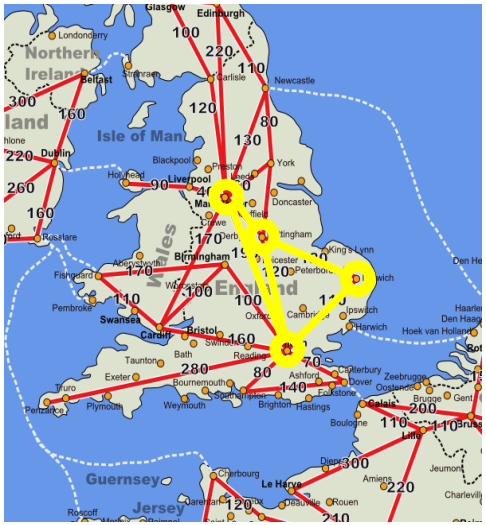
- L'implementazione deve sfruttare l'approccio a **liste di adiacenza** visto a teoria





OPPURE LA VARIANTE CON PUNTATORE





FUNZIONALITÀ OFFERTE

```
------------------------------------------------------------------------
                                MENU
------------------------------------------------------------------------
1. Inserimento della mappa (grafo) da tastiera
2. Inserimento della mappa (grafo) da file
3. Visualizzazione della mappa (grafo)
4. Inserimento di una citta' (vertice) nella mappa (grafo)
5. Inserimento di una nuova strada (arco) nella mappa (grafo)
6. Determinazione del numero di citta' presenti nella mappa
7. Determinazione del numero di strade nella mappa
8. Determinazione del grado di un citta'
9. Verifica dell'adiacenza tra due citta'
10. Stampa le citta' adiacenti (ad una data citta')
11. Calcolo di un cammino tra due citta'
0. Uscita

Fornisci la tua scelta --->
```

Zip-file contenente 5 file

OPERATIVAMENTE (1)

- **list.h**, **list.cpp**
  - Header e implementazione del tipo di dato **lista**
- **graph.h**, **graph.cpp**
  - Header e implementazione del tipo di dato **grafo**
- **main.cpp**
  - Definisce il main che contiene un *semplice menu* il quale richiama le funzioni implementate in graph.cpp

Zip-file contenente 5 file

OPERATIVAMENTE (1)

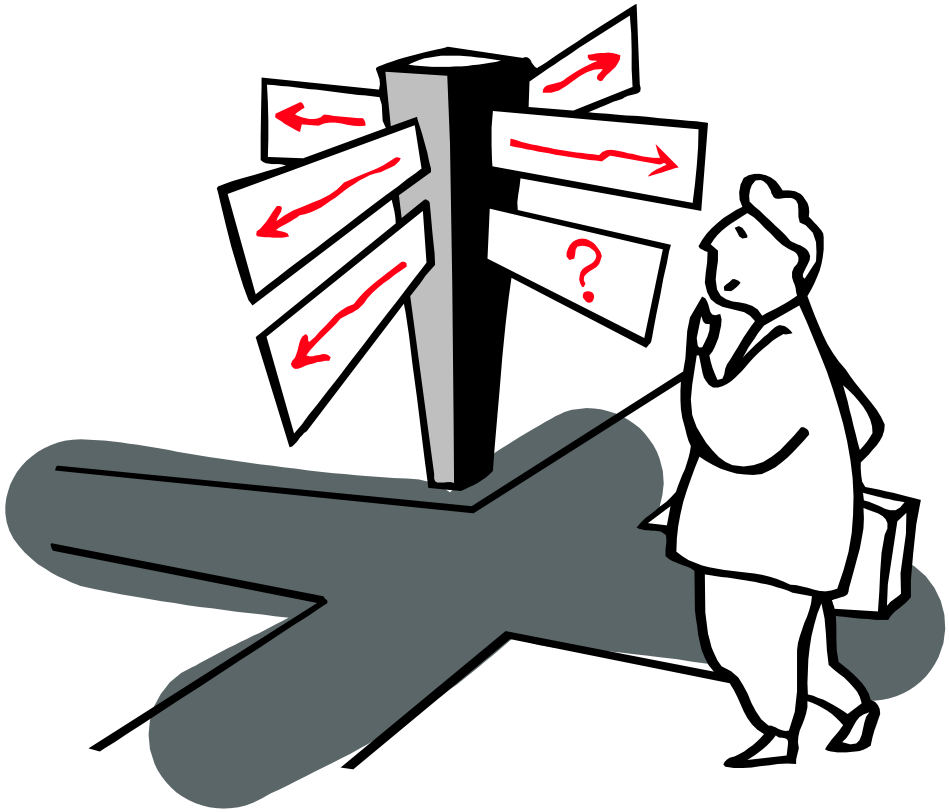
- **list.h**, **list.cpp**
  - Header e implementazione del tipo di dato **lista**
- **graph.h**, **graph.cpp**
  - Header e implementazione del tipo di dato **grafo**
- **main.cpp**
  - Definisce il main che contiene un *semplice menu* il quale richiama le funzioni implementate in graph.cpp

NON modificare!

OPERATIVAMENTE (2)

- Creare le struct relative a vertici ed archi
  - Come visto a lezione
- Implementare le operazioni del grafo *creatyEmptyGraph()*, *addVertex()* e *addEdge()* in modo tale da poter leggere la mappa da file e standard-input
- Implementare la *printGraph()* in modo da poter testare se l'inserimento delle mappe funziona
- Implementare le operazioni semplici come: *numVertices()*, *numEdges*(), *areAdjacent()*, *nodeDegree*()
- Implementare le liste e l'operazione del grafo *adjacentList()* (che ritorna la lista di adiacenza di un nodo)
- Implementare la *findPath()*

THE END ...



Domande?