LABORATORIO 8
NAVIGATORE SATELLITARE

Algoritmi e strutture dati

NAVIGATORE SATELLITARE

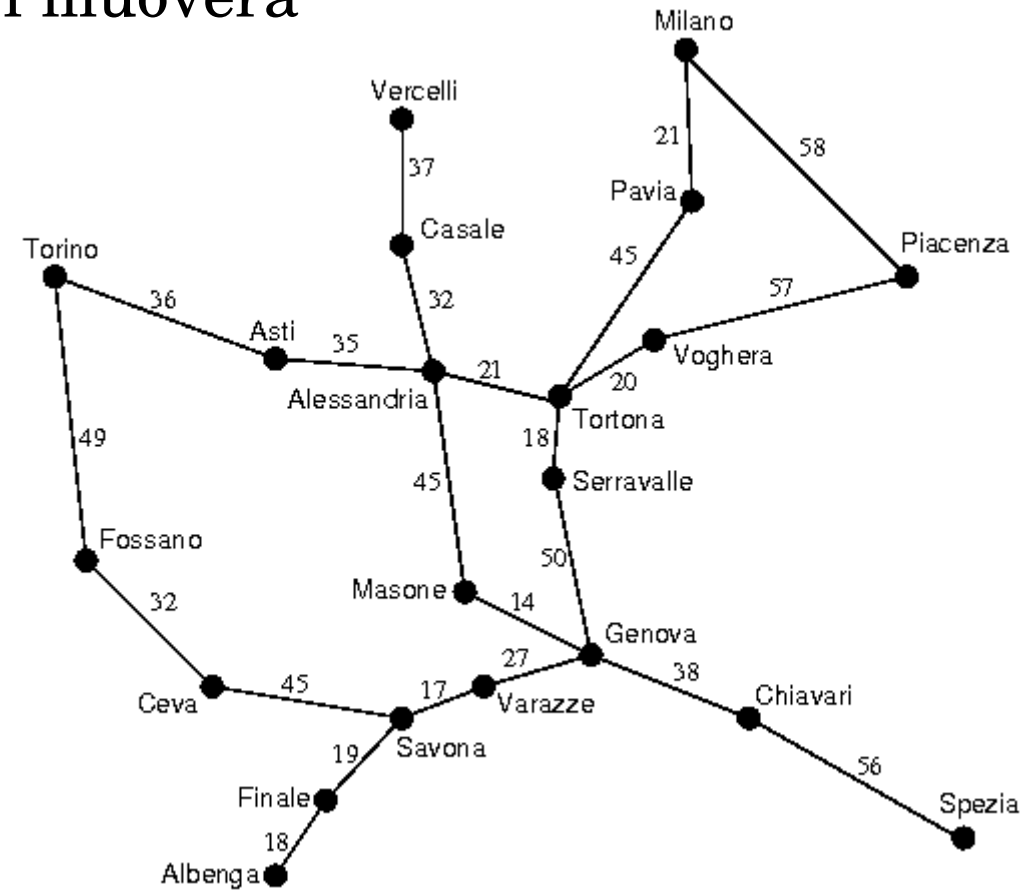
- Consideriamo un navigatore satellitare

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 località venga visitata al piu una volta

MAPPE

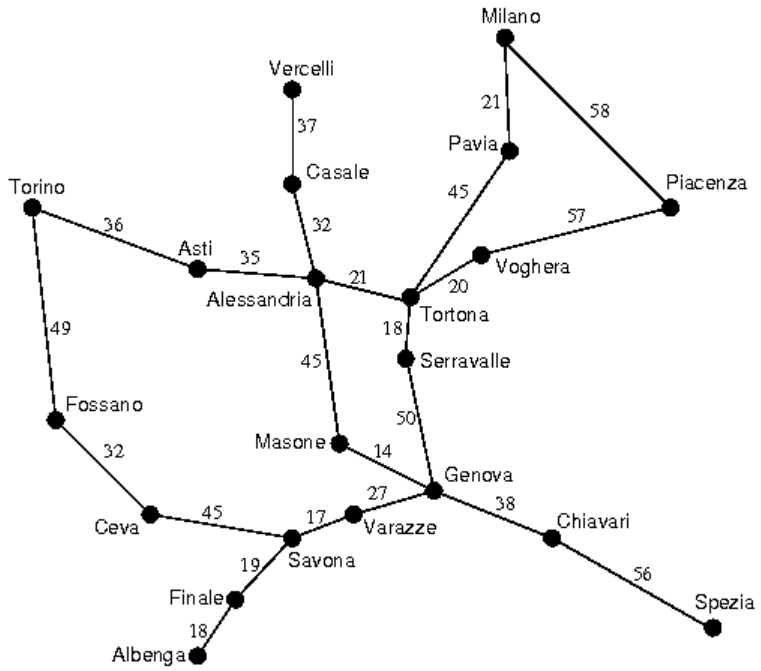
- Il navigatore **deve poter caricare, da file o da standard input, le mappe delle regioni** in cui l'utente si muoverà



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

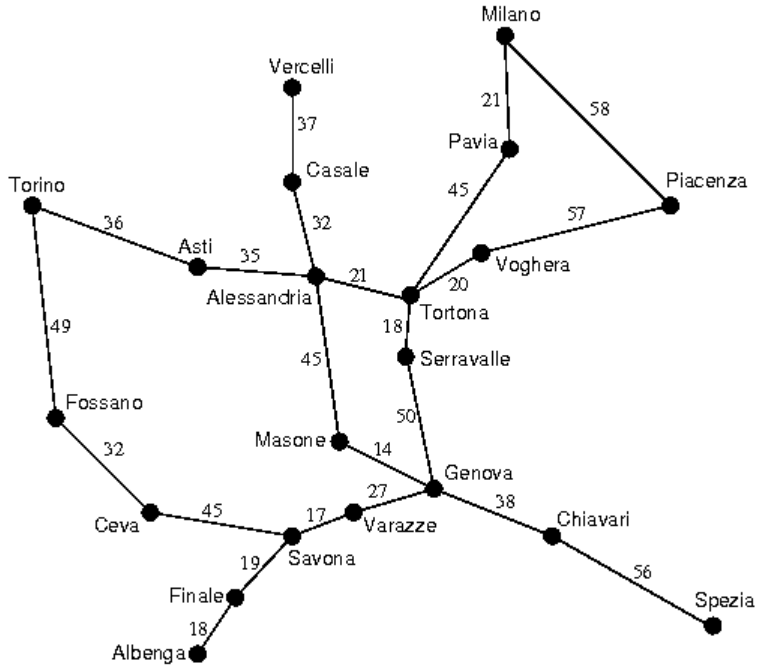
...
Tortona Alessandria 21
Genova Chiavari 38
Chiavari Spezia 56
Tortona Voghera 20
Piacenza Voghera 57
Piacenza Milano 58
Tortona Pavia 45
Pavia Milano 21
0



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



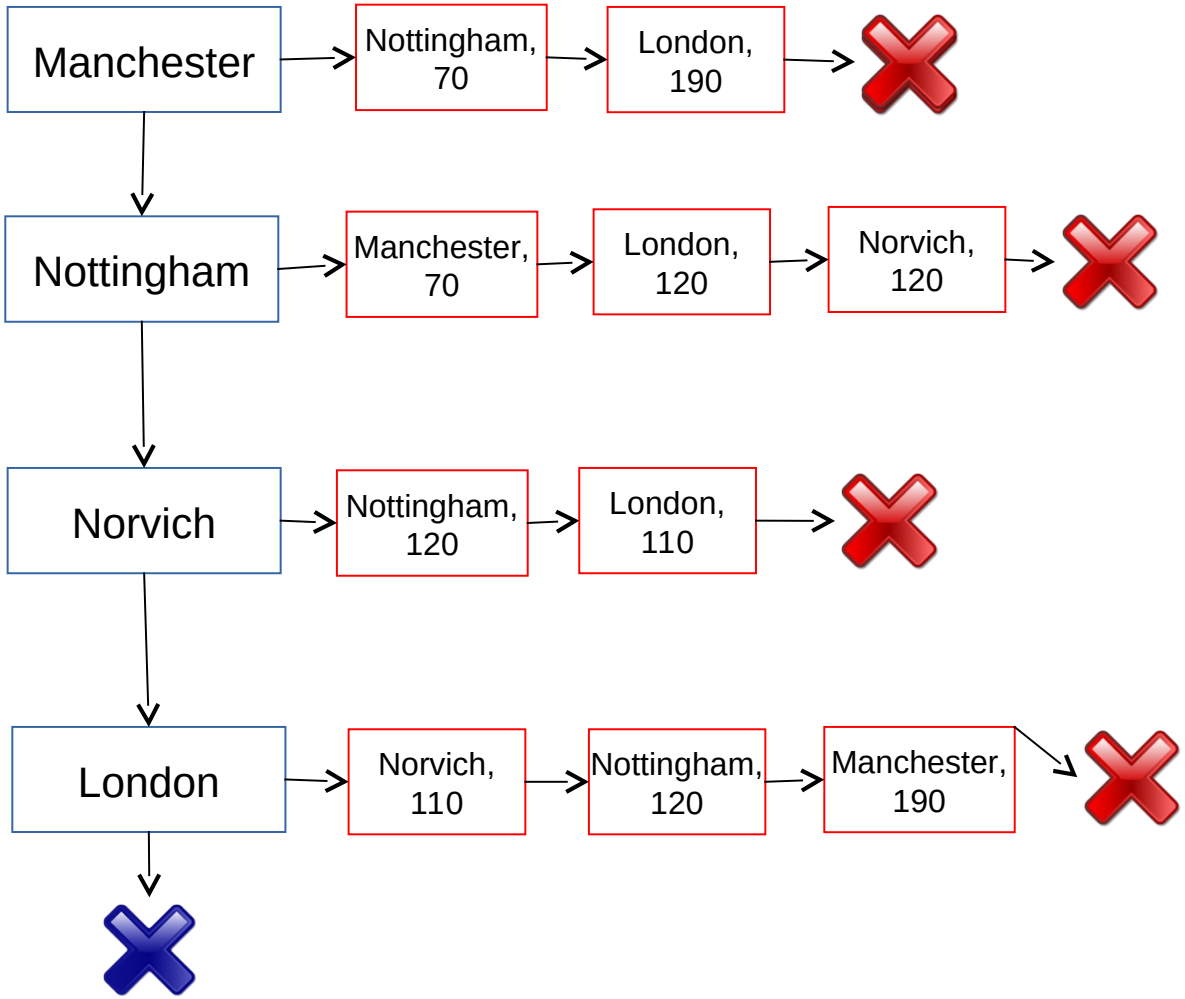
Si noti lo "0" in fondo, utilizzato per terminare la sequenza di lettura

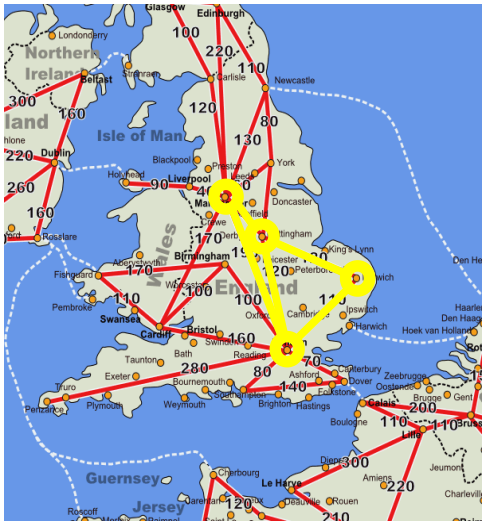
LABORATORIO 8

- 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 città. 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

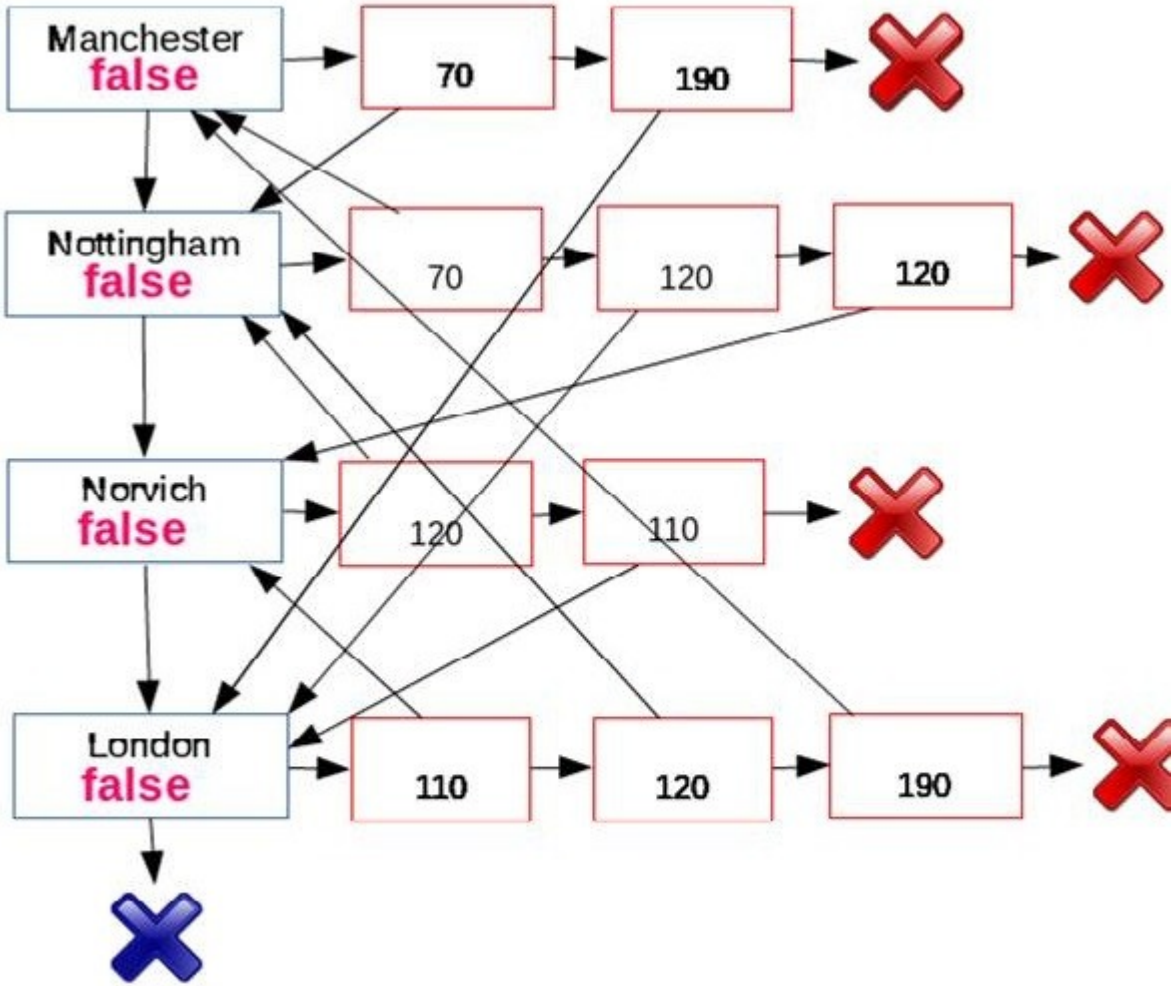
IMPLEMENTAZIONE DEL GRAFO

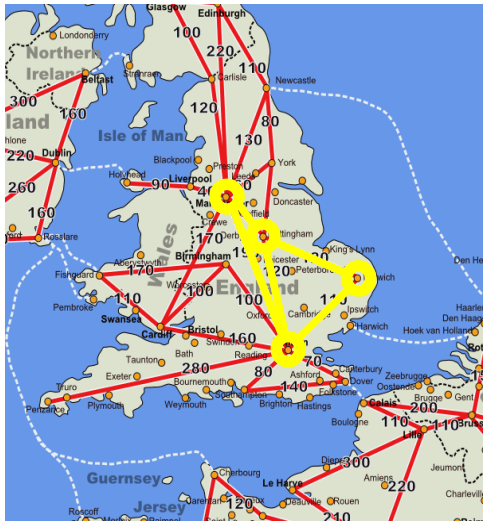
- L'implementazione deve sfruttare l'approccio a **liste di adiacenza** visto a teoria





OPPURE LA VARIANTE CON PUNTATORE





FUNZIONALITÀ OFFERTE

```
a: inserisci la mappa (grafo) da tastiera
b: inserisci la mappa da file
c: visualizza la mappa
d: inserisci una città (vertice) nella mappa
e: inserisci una nuova strada (arco) nella mappa
f: stampa il numero di città presenti nella mappa
g: stampa il numero di strade nella mappa
h: stampa il numero di strade che portano a una città (il suo grado)
i: verifica se due città sono adiacenti (collegate da una strada)
j: stampa le città adiacenti a una data città
k: calcola un cammino tra due città
l: svuota la mappa

digita p per stampare il menu, q per terminare
>
```

Zip-file contenente 5 file

OPERATIVAMENTE (I)

- **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!

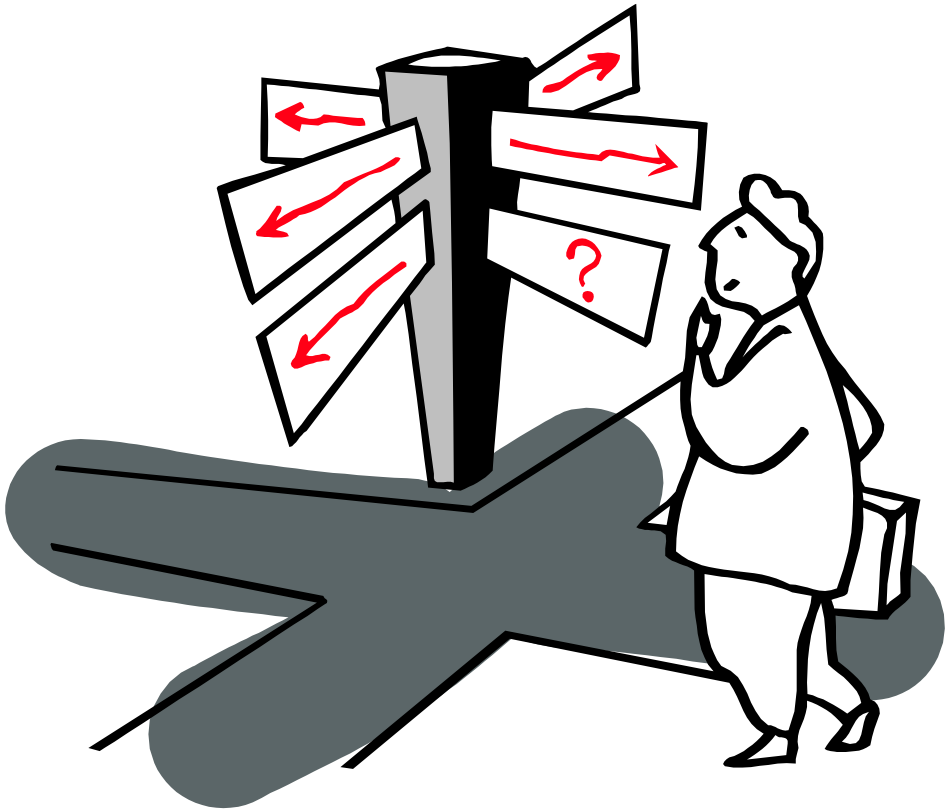
OPERATIVAMENTE (2)

- 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()*

TEST

- Funzionano come visto finora
  - python3 ./run_test.py ./NOME_ESEGUIBILE
- **Attenzione!** Non verifichiamo che i percorsi trovati siano corretti, quello dovete **testarlo voi**
- Se un test fallisce, guardate il file **test.json** e verificate cosa fa il vostro codice con quell'input, confrontandolo con quello atteso
- Per fare prima, fate copia e incolla dei comandi di input
  - Eseguite **echo "MY_INPUT"|./a.out q**
  - Questo passa MY_INPUT allo standard input del programma, come se fosse dato da tastiera

THE END ...



Domande?