Un caso di studio: Navigatore stradale

Navigatore stradale

Si vuole creare un navigatore stradale, sfruttando la struttura dati del grafo

In particolare ciò che il navigatore deve realizzare è:

- Poter leggere da file un grafo stradale e stamparlo
- Un Sistema per chiedere all'utente una vertice di partenza e un vertice di arrivo mostrando il percorso ottimale
- Poter navigare il risultato ottenuto e poter cambiare il percorso a ogni incrocio
  - E ricalcolare il percorso migliore nuovamente

Implementazione

Si mantiene il grafo stradale in un grafo G=(V,E) orientato e pesato in cui

- V è l'insieme dei vertici, ovvero degli incroci. Ogni incrocio ha un identificativo intero e un nome
- E è l'insieme degli archi, ovvero delle vie che collegano due incroci

- I vertici e gli archi sono memorizzati in un file come segue
  - Nella prima riga si trova il numero di vertici N
  - Nelle N righe successive c'è il nome dell'incrocio, uno per riga. L'ordine determina l'identificativo dell'incrocio
  - Al termine delle N righe, si trova l'elenco degli archi, uno per riga, nella forma "u v w", dove u è l'identificativo del vertice sorgente, v è l'identificativo del vertice di arrivo, w è il peso

Esempio file

5

Incrocio 1

Incrocio 2

Incrocio 3

Incrocio 4

Incrocio 5

1 3 0.5

1 5 0.1

2 4 0.3

3 5 0.2

Inizio navigazione

- Il sistema deve chiedere all'utente l'identificativo di un incrocio di partenza e di un incrocio di arrivo
- Nel caso in cui uno dei due non sia presente nel grafo, deve fornire all'utente un opportuno messaggio di errore
- Deve quindi trovare il percorso ottimale, se esiste, dalla partenza all'arrivo, e memorizzarlo
- Nel caso in cui il percorso non sia presente, deve fornire all'utente un opportuno messaggio di errore
- Altrimenti deve far partire la navigazione, indicando il nome del primo incrocio e suggerendo il successivo

File di partenza

- **grafo.h, grafo.cc, tipo.h, tipo.cc**
  - Per la gestione del grafo stradale
- **strade**
  - Che contiene i dati di incroci e strade

Punti da realizzare – Punto 1

Creare il file `compito.cc` contenente:

Funzione **`grafo load`**`(int& dim)`

Carica dal file il grafo e lo restituisce

La dimensione del grafo viene assegnata a dim

Procedura **`stampa`**`(grafo*, int)`

Stampa il contenuto del grafo, indicando per ogni incrocio tutti gli incroci adiacenti con il rispettivo peso

Un `main` che richiama in sequenza le due funzioni.

Creare un makefile per la generazione dell'eseguibile.

Il makefile deve contenere i phony target clean e cleanall.

Punti da realizzare – Punto 2

Funzione di calcolo percorso:

- aggiungere la funzione **int*** `calcolo(grafo* g, int n, int s, int d, int& w)`

Calcola il percorso migliore da s a d, se esiste, ovvero quello con peso totale minore w

Il percorso viene restituito come un vettore di interi che rappresenta la sequenza di incroci

Il vettore può essere preallocato alla massima dimensione dei nodi del grafo

Stampare il percorso migliore come sequenza di incroci, indicando poi il peso totale

Punti da realizzare – Punto 3

Aggiungere la funzione ricalcola, che chiama **calcolo** chiedendo all'utente una sorgente e una destinazione e stampa il percorso migliore

Per ogni incrocio del percorso migliore, mostra all'utente il prossimo incrocio da visitare e anche tutte le altre possibilità.

- Se l'utente seleziona l'incrocio consigliato si continua la stampa normale
- Se l'utente seleziona un altro incrocio occorre ricalcolare il percorso migliore a partire dal punto in cui si trova l'utente in quell momento, e riproporre successivamente lo stesso Sistema per continuare la navigazione
- Estendere il main affinché chieda all'utente l'incrocio di partenza, l'incrocio di arrive e chieda passo passo gli incroci da raggiungere

Punti da realizzare – Punto 4

- Generare la documentazione con Doxygen
- Aggiungere una breve descrizione del programma realizzato per la main page
- Commentare la procedura stampa descrivendo i parametri in ingresso e aggiungendo una breve descrizione sull'obiettivo

Dati di prova

Il punto 1 deve stampare l'elenco degli incroci come da richiesta.

Il punto 2 con s=2 e d=8 deve stampare

2 3 5 4 8 – Peso totale=9

Il punto 3 con s=7 e d=10 deve stampare

7 1 3 5 6 10 – Peso totale=16

Incrocio 7, prossimi incroci possibili: 1(peso 2),4(peso 9)

Utente seleziona 1 (che è gia nel percorso migliore)

Incrocio 1, prossimi incroci possibili: 3(peso 3), 4(peso 5)

Utente seleziona 4 (che non è nel percorso migliore)

ricalcolo

4 8 5 6 10 – Peso totale=10

Incrocio 4, prossimi incroci possibili: 3(peso 2),2(peso 8),1(peso 1),8(peso 2)

…