Esercizio 4

In questo esercizio viene chiesto di implementare delle librerie per la gestione dei grafi con la successiva esecuzione dell'algoritmo di Dijkstra su una "mappa" contente alcune città italiane, in formato csv, per calcolarne il percorso minimo.

Strutture dati utilizzate per la libreria:

Le strutture dati che si è scelto di implementare per la corretta esecuzione dei metodi nei tempi richiesti sono:

- HashMap per la struttura a grafo (caso medio con operazioni in O(1))
- HashSet per contenere gli archi (caso medio con operazioni in O(1))
- LinkedList necessaria per tenere conto, per ciascun nodo, degli archi che partono da quest'ultimo per raggiungere gli altri nodi presenti all'interno del grafo.

Come visto a lezione l'HashMap (implementazione di una tabella hash) si occupa di mappare i valori chiave, consentendo all'utente di avere un accesso diretto alla posizione desiderata.
Necessitiamo di questa struttura dati in quanto l'utilizzo di un "classico array" implicherebbe l'avere le classiche operazioni di ricerca e cancellazione in tempo O(n)(nel caso peggiore necessito di scorrere tutta la struttura) che le hash table offrono in O(1) nel caso medio.
Discorso analogo per quanto riguarda l'HashSet sulla questione tempo.

Funzionamento:

La classe principale Graph, all'interno di graphlibrary, contiene il grafo principale come un HashMap di tipo <G-generico, LinkedList<Route<G-generico, T-generico>>>, questo serve perché nel nostro caso la chiave sarà il nodo a cui ci riferiamo, mentre la LinkedList conterrà tutti gli archi che a partire dal nodo in questione ci permettono di raggiungere i suoi adiacenti, che abbiamo chiamato "Route".
La seconda struttura contenuta è l'HashSet di tipo <EntireRoute<G-generico, T-generico>>, questa serve per contenere quelli che noi abbiamo chiamato "EntireRoute", ovvero oggetti del tipo "primo nodo, secondo nodo e peso dell'arco" che memorizzano quindi tutti gli archi presenti nel nostro grafo e il relativo peso. Tale struttura viene usata come supporto per alcune operazioni che altrimenti, solo con la struttura che rappresenta il grafo principale, non si riuscirebbero ad effettuare in tempi O(1). Per esempio, per verificare l'esistenza di un determinato arco senza questo HashSet, dovrei scorrere l'HashMap iterando sulle adiacenze dei vari nodi, quindi senza riuscire ad effettuare un accesso diretto(O(1)) o comunque senza dover confrontare l'arco desiderato con gli adiacenti, presenti nella lista di adiacenza, del nodo iniziale dell'arco.

Dijkstra:

Lalgoritmo di Dijkstra si appoggia su strutture implementate nell'esercizio 3 del MinimalHeap, questo consente all'algoritmo di rintracciare sempre il nodo minimo, estrarlo, calcolare la distanza da quel nodo ai suoi adiacenti, impostare il nuovo valore dei nodi adiacenti, ed estrarre il successivo con valore minimo. Procedendo in questo modo si crea uno heap con i valori minimi trovato dalla sorgente A alla destinazione B impostati nel main.

ESEMPI:

- Torino(sorgente) dista da Catania(una delle destinazioni) : 1207.68km;
- Milano(sorgente) dista da Venezia(una delle destinazioni) : 309.852 km;
- Gemini(sorgente) dista da Tarvisio(una delle destinazioni) : 974.263 km;

Vista la natura non orientata del grafo, la distanza da una "sorgente" ad un a "destinazione" è equivalente in entrambi i sensi.

Riccardo Venturini & Cosmin Lavric