Esercizio 4 - Kruskal

UnionFind

Per realizzare la struttura dati abbiamo utilizzato la classe HashMap (piattaforma Java SE 8) (oracle.com) che associa un elemento dell'insieme al nodo di quell'elemento, in più ogni nodo contiene l'indirizzo del nodo "genitore" in modo da poter ricavare la radice dell'albero partendo da un qualsiasi nodo.
Abbiamo scelto di aggiungere il metodo find() all'interno della classe Node per permettere delle semplici chiamate ricorsive del metodo.
Il resto dell'implementazione della classe è come quella riportata sul testo Cormen et al., Introduzione agli algoritmi e strutture dati, McGraw-Hill, nel capitolo Strutture dati per insiemi disgiunti, paragrafo Foreste di insiemi disgiunti.

Struttura dati grafo

Per implementare la struttura dati **grafo** ottimale per dati sparsi abbiamo scelto di utilizzare una lista di adiacenza.
Per rispettare le complessità richieste, abbiamo scelto di implementare la lista di adiacenza con la classe HashMap (piattaforma Java SE 8) (oracle.com) in cui:

- Ogni chiave rappresenta un nodo;
- Il valore associato ad ogni chiave/nodo è la lista di adiacenza del singolo Node rappresentata come una lista di oggetti Arc;

Classe Arc

Classe che implementa una lista di archi che partono da un nodo specifico.
Attributi:

1. elem: oggetto generico contenente il nodo raggiunto dall'arco. Il nodo utilizzato per raggiungere elem è quello utilizzato come chiave nell'HashMap per ottenere l'oggetto Arc;
2. next: oggetto di tipo Arc necessario per implementare la lista;
3. weight: "peso" dell'arco, quest'attributo implementa l'interfaccia comparable, necessaria per poter garantire che due oggetti generici possano essere confrontati.
4. Fake: attrito booleano
   Per **implementare grafi indiretti** abbiamo scelto di inserire due archi diretti per ogni arco diretto (es. per due nodi u,v inseriamo l'arco u->v e l'arco v->u). L'attributo fake serve a riconoscere l'arco copiato(il secondo arco);

Gestione degli errori

Nel caso avvenga un operazione errata viene lanciato un errore specifico: AdjacencyListException

Classe AdjacencyList

Questa classe implementa una lista di adiacenza.
Attributi:

1. Graph: un HashMap i cui ogni chiave rappresenta un nodo del grafo
2. Direct: attributo booleano che differenzia grafi diretti da grafi indiretti.
3. numberArcs: contatore del numero di archi all'interno del grafo. Questo attributo permette di abbassare la complessità della determinazione del numero di archi da O(n) a O(1)

Per il recupero degli archi del grafo abbiamo deciso di utilizzare un'ulteriore struttura dati: ArrayList di FullArc. La classe FullArc permette una più semplice rappresentazione di un arco dato che contiene semplicemente due nodi ed un peso.

Esecuzione dell'algoritmo di Kruskal

Per la parificazione del file ci siamo appoggiati alla classe BufferedReader (Java Platform SE 7) (oracle.com). L'algoritmo di kruskal eseguito con i dati contenuti nel file italian_dist_graph.csv ci ha dato come risultato archi al cui somma corrisponde 89939912,586 m.