Autori : Lobascio Giuseppe Pio, MATR: 916981, Mansi Matteo, MATR: 929683

Relazione Esercizi 3 – 4

In questa ultima parte di progetto, si chiedeva di realizzare una libreria (esercizio 3) che implementasse l'algoritmo di UnionFindSet.

Per far funzionare questo algoritmo, si è richiedeva l'utilizzo di strutture dati generiche.

E' stato infatti utilizzata la struttura dati List per memorizzare i vertici, un semplice array di interi per gli archi inizializzato a null.

Questa classe permette di fornire un makeSet iniziale, di Unire due insiemi/gruppi con parent differenti e di trovare il parent di un dato insieme, infine di procedere con union by rank e compressione del cammino.

La struttura dati UnionFind è generica in quanto è stato d'aiuto nello sviluppo e nell'implementazione dell'esercizio 4.

Per l'esercizio 4, è stata implementata la struttura dati Grafo contenente un Map così formato :

Map<Disjoint<T>, Map<Disjoint<T>, Double>>, dove Disjoint<T> è il nodo di partenza, e Map<Disjoint<T>, Double> invece identifica il nodo di arrivo e il peso.

Il grafo non è diretto: ogni volta che un Edge viene aggiunto del tipo (a,b,w) anche l'Edge (b,a,w) viene aggiunto.

Usando l'Algoritmo di Kruskal abbiamo potuto calcolare il cammino minimo per ottenere un unico MST, in modo da attraversare ogni suo nodo una sola volta. Infine sono realizzati altri CSV per verificarne l'efficienza. Per il CSV proposto ovvero italian_dist_graph, l'esercizio restituisce 18640 vertici, 18637 archi e come peso del cammino minimo 89.939,913 Km.