Relazione progetto totale II

Algoritmi e Strutture Dati Laboratorio

10 settembre 2021

Michele Pioli

Matricola: 110388

Alessio Tozzi

Matricola: 109376

Table of Contents

[Algoritmo di Dijkstra 3](#_Toc82105313)

[Algoritmo di Bellman-Ford 4](#_Toc82105314)

La preparazione alla stesura del codice si è incentrata sulla ricerca e studio del funzionamento dei vari algoritmi. Lo studio della teoria è avvenuto tramite il libro, nonché attraverso siti web. Di particolare aiuto sono stati YouTube.com e le immagini in formato .gif su Wikipedia.com, che hanno permesso di vedere una rappresentazione più dinamica e intuitiva degli algoritmi.

In seguito sulla teoria, si è passati alla scrittura. Le parti centrali dell’algoritmo sono risultate, come previsto, le più laboriose.

# Algoritmo di Dijkstra

Dopo una stesura approssimativa dell’algoritmo di Dijkstra si è passati ai test, grazie ai quali si sono potute osservare le mancanze nel codice. Un aiuto decisivo è stato fornito dalla realizzazione di un grafo, che è poi stato riutilizzato con piccole variazioni per gli altri algoritmi, e la scrittura della relativa struttura dati seguendo l’algoritmo precedentemente scritto.

A picture containing diagram

Description automatically generated

Figure - Algoritmo di Dijkstra

Seguendo passo passo il codice siamo riusciti ad individuare quali sezioni del codice fosse necessario correggere. Un esempio di codice di codice di Dijkstra che ha richiesto lavoro è il seguente:

GraphNode<L> node;  
for (int i = 0; i < this.graph.nodeCount(); i++) {  
 node = (GraphNode<L>) this.heap.extractMinimum();  
 if (node.getPriority() == Double.POSITIVE\_INFINITY) {  
 node.setPrevious(null);  
 continue;  
 }  
 node.setColor(GraphNode.COLOR\_BLACK);  
 checkNewPriority(node);  
}

Questo è il fulcro dell’algoritmo di Dijkstra. Il ciclo for itera tutti i nodi nel grafo al fine di trovare il nodo con priorità inferiore ed estrarlo dalla struttura dati (heap, in questa implementazione). In seguito, si controlla che non vi siano nodi che non possono essere raggiunti dal nodo di partenza. Se ve ne dovessero essere, questi saranno privi di previous node, e dunque questo verrà posto a null. Successivamente, si pone nero il colore del nodo visitato. Infine, per semplificare e rendere modulare il metodo, si è scelto di creare un metodo separato per il controllo e assegnazione della priorità ai nodi (checkNewPriority(node)).

# Algoritmo di Bellman-Ford