Weekly Assignment 4

25th September 2019

Exercise 1.

Give an algorithm that converts an array that represents a Max-Heap into an array that represents a Min-Heap. Your algorithm should run in linear time.

Exercise 2.

1. Draw the heap obtained by inserting 11 in the heap shown in Figure 1
2. Draw the heap obtained by removing 14 from the heap shown in Figure 1

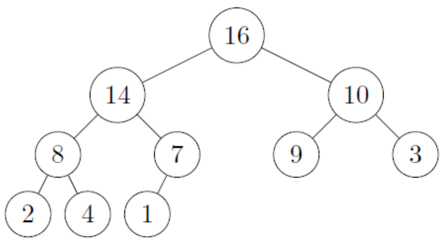

Figure 1: A Max-Heap with 10 elements

Exercise 3.

Run Dijkstra's algorithm on the weighted graph below, using vertex 0 as the source. List the vertices in the order which they are extracted from the queue, and compute all distances at each step.

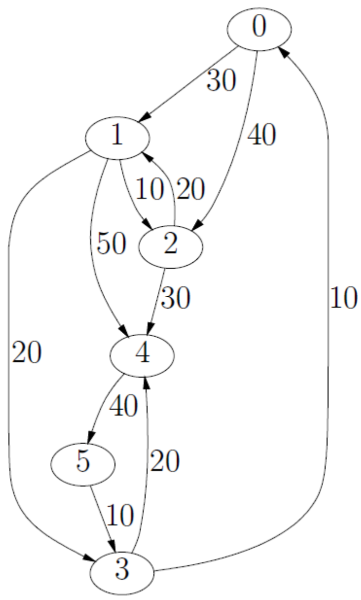

Figure 2: Graph of Exercise 3.

Exercise 4.

Present an example that illustrates why Dijkstra's algorithm does not always work in case edges may have negative weights.

Exercise 5.

Figure 3: Airlines network.

Airlines companies sell to travelers flights with or without connections, mostly depending on their airlines network. It can happen that secondary airlines are covered with small and slow aircrafts, hence a longer transit time. It can then be faster to fly via another city (connecting flight) rather than taking a direct flight, but that's not always the case. As connections are unpleasant, direct flights are preferred if the total flight time is equal. Otherwise, if the direct flight takes longer, then the connecting flight is preferred.

Let us define $easiest[destination]$ = minimum number of successive flights in a shortest path from $origin$ to destination.

In our airlines example, the *easiest* values are (for origin Madrid):

- 0 for Madrid.
- 1 for Amsterdam, Berlin, Rome.
- 2 for Stockholm, Kiev.
- 3 for Moscow.

We can transform the problem of the airline company into a graph problem. Give an efficient algorithm taking as input a weighted graph $G = (V, E)$ and a source vertex s, and as output the values of *easiest* for all vertices.

Exercise 6

You are given a strongly connected directed graph $G = (V, E)$ with positive edge weights along with a particular node $v \in V$. Write an efficient algorithm for finding shortest paths between all pairs of nodes, with the one restriction that these paths must all pass through v.

Exercise 7

There is a network of roads $G = (V, E)$ connecting a set of cities V. Each road in E has an associated length. There is a proposal to add one new road to this network, and there is a list $list$ of pairs of cities between which the new road can be built. Each such potential road $r \in list$ has an associated length $len(r)$. As a designer for the public works department you are asked to determine the road $r \in list$ whose addition to the existing network G would result in the maximum decrease in the driving distance between two fixed cities s and t in the network. Design an efficient algorithm for solving this problem.