

Design Document

Assignment 2 – PS11 - [Hospital Emergency]



September 2, 2020

GROUP-037

# Algorithm Details

The given use-case demands to find the shortest path from the Hospital to Airport, hence we find Dijkstra's algorithm suitable for the implementation.

## Description

Dijkstra's algorithm (or Dijkstra's Shortest Path First algorithm, SPF algorithm) is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. This is asymptotically the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded non-negative weights. Dijkstra's algorithm is applicable for:

* Both directed and undirected graphs
* All edges must have nonnegative weights
* Graph must be connected

Dijkstra's algorithm was, originally, published by Edsger Wybe Dijkstra, winner of the 1972 A. M. Turing Award.

# Design Strategy

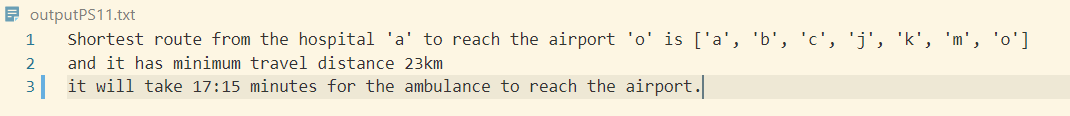
Following points are taken into account while designing the program:

1. The input is designed to take from **inputPS11.txt** file which has a which has the fixed format mentioned below using the “/” as a field separator:

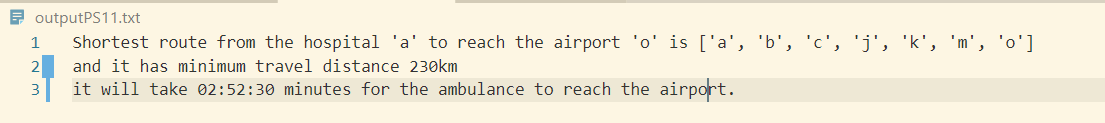
<node 1> / <node 2> / <distance in km>

1. All the lines in the input is converted into lower case to avoid ambiguity in the name of the nodes.
2. The output is written into a file **outputPS11.txt.**

**Sample:**

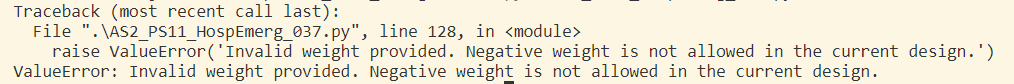


Output where the time taken is in hours:

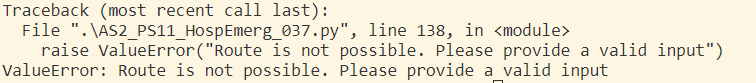


1. The algorithm supports only non-negative weights, hence it is taken care that the application rejects an invalid graph with negative weight.

**Sample error thrown:**



1. Suppose a graph is given where a route is not possible from the hospital node to the airport node, the error is handled by providing the below notification on execution of code.



1. The problem statement is implemented using python 3.7.
2. The problem statement is implemented with the help of python dictionary.

# Time Complexity

Dijkstra's algorithm uses a data structure for storing and querying partial solutions sorted by distance from the start.

* Worst case time complexity: **Θ(E+V log V)**
* Average case time complexity: **Θ(E+V log V)**
* Best case time complexity: **Θ(E+V log V)**
* Space complexity: **Θ(V)**

Time complexity is **Θ(E+V^2)** if priority queue is not used.