HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
BBM204 PROGRAMMING LAB.
ASSIGNMENT #5

Subject: Shortest Path Algorithm
Submission Date: April 30,2014
Deadline: May 15,2015
Programming Language: Java
Advisors: TAs.Levent Karacan

1 Finding K Shortest Paths

There are different algorithms to find shortest path in a given direct or indirect graph data structure for example Breadth First Search and Dijkstra. However, for some practical applications, it is not enough to find single shortest path. If some part of the shortest path is not available or we need to several options, what will we do? We should find shortest paths rather than single shortest path. Consider a navigation system, if a single shortest path is not available because of traffic jam or any reason, the navigation system should offer an alternative way that must be shortest in remaining possible ways.

In this experiment, you will implement and analysis an algorithm that finds K shortest paths between two locations.

2 Problem

Given a weighted graph representing locations as nodes and roads as edges with related distances, you will modify BFS algorithm to find K shortest paths with their lengths. The order of paths are found must be correct and no path must be abandon.

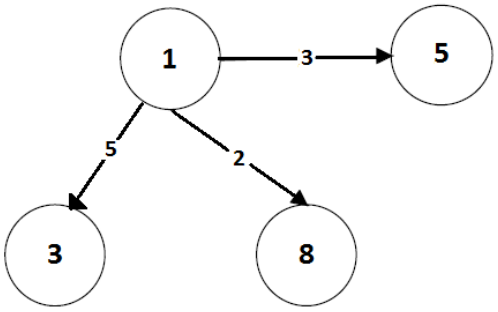
Input: A map with locations and roads which are described by a text file with starter and destination locations.

Output: K shortest paths(1. shortest path, 2. shortest path, 3. shortest path ... K. shortest path) with their lengths

You are given an input file in the following format:

S:5,D:13
1. 5(3),8(2),3(5)
2. 3(10),4(4)
3. 6(2),3(5)
⋮
N. . . . ,. . . ,. . .

,where first line indicates ID numbers of starter and destination vertices and the following lines specify graph vertices with ID numbers(1,2,3 . . . N). After the ID number for each line , connected locations(5(3) 8(2) 3(5)) are indicated with distances in the brackets. For example the first line draws a graph structure as:



Original BFS algorithm does not support computing shortest path in a weighted graph therefore you will use a priority queue to hold distances in ascending order, then every vertex stores the length from the starter to the vertex itself. Consider the graph given in Figure 1 with 12 locations connected to each other with roads indicated by arrows with distances.

When a user asks for road from a starter location to destination, your program will offer alternative paths according to the path lengths as shown in Figure 2. In this example starter and destination vertices are given with ID numbers 1 and 12 respectively.

To find all shortest paths from Vertex 1 to Vertex 12, you can use priority queue that traces minimum paths from starter vertex to destination vertex according to BFS search algorithm. When the destination vertex is reached at the head of the priority queue, you can pop the minimum distance from the heap tree structure and backtrack to find related path. As can be seen in the example given in Figure 3, you will keep the distances of partial paths between starter and current vertex on heap tree so that when you reach the target vertex you will be able to find the shortest path at the head of the heap tree structure. If you find next shortest path, you should continue to operation on heap tree by searching by BFS algorithm until target vertex is reached at the head of the heap tree.

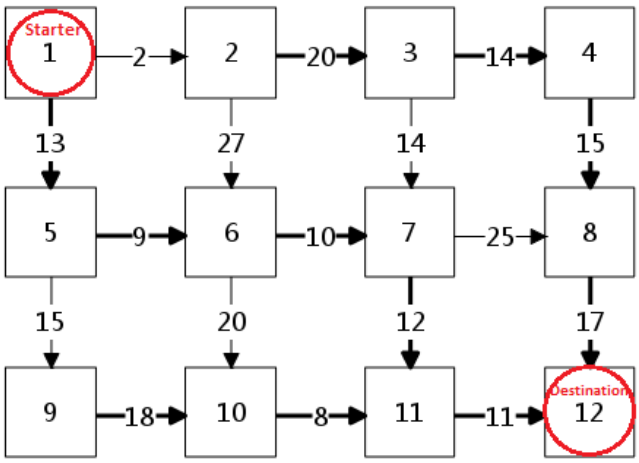


Figure 1: Example graph with 12 vertices

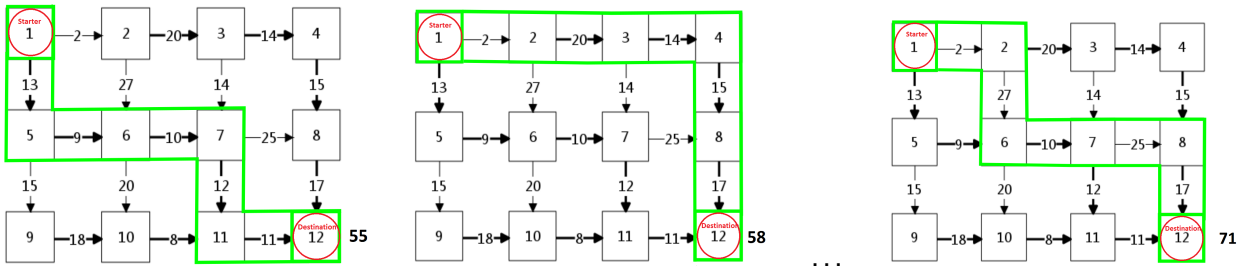


Figure 2: Illustriation of all shortest paths for given example graph in Figure 1

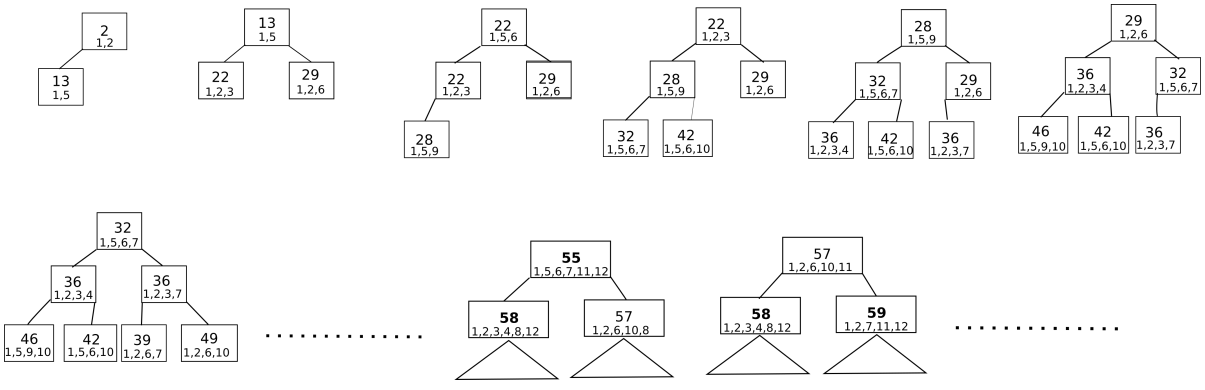


Figure 3: Heap tree structure operations for given graph to find K shortest paths

3 Details

- Read vertices from input file
- Create a graph structure
- Implement a function takes starter and destination vertices, number of shortest paths K that will be listed as parameters and returns K shortest paths with distances
  - Start from starter vertex
  - Search according to BFS algorithm
  - Keep distances of vertices to starter vertex on heap tree when walking around on graph
  - When reaching destination, backtrack to starter point and find shortest paths.
- Print K shortest paths to a output file in the order of ascending distance

4 Input and Output

Input file format will be as in Figure 4 and your program must print results to output file as shown in Figure 5. You will be given example input and output files.

```
S:1,D:12
1. 2(2),5(13)
2. 3(20),6(27)
3. 4(14),7(14)
4. 8(15)
5. 6(9),9(15)
6. 7(10),10(20)
7. 8(25),11(12)
8. 12(7)
9. 10(18)
10. 11(8)
11. 12(11)
```

Figure 4: input.txt

```
 1. Shortest Path: 55   1     5     6     7    11    12
 2. Shortest Path: 58   1     2     3     4     8    12
 3. Shortest Path: 59   1     2     3     7    11    12
 4. Shortest Path: 61   1     5     6    10    11    12
 5. Shortest Path: 62   1     2     6     7    11    12
 6. Shortest Path: 64   1     5     6     7     8    12
 7. Shortest Path: 65   1     5     9    10    11    12
 8. Shortest Path: 68   1     2     3     7     8    12
 9. Shortest Path: 68   1     2     6    10    11    12
10. Shortest Path: 71   1     2     6     7     8    12
```

Figure 5: output.txt

Report

Report must be 2 pages at most.

1. Brief overview of problem (1/2 page)
2. Include solutions to problem in an algorithmic way(1/2 page)
3. Discuss the solution(1/2 page)
4. Research another solutions and talk about how they solve the K shortest paths problem and what advantages they provide (1/2 page)

Note: You will earn $\%10$ of total points from your report. Best report will earn extra points.

Notes

Your experiments will be executed in DEV machine, please make sure whether it works properly on Dev Machine before or not. You should use comment lines to explain your code. Give your report in the necessary details. Save all your work until the assignment is graded.You can ask your questions about the experiment on Piazza.

You have to give your experiment files in the given format below;

[Student id]
- [code]
  - *.java
  - Main.java
- [report]
  - pset5.pdf

Your assignment will not be evaluated If you do not build a valid code.

Policy

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone elses work(from internet), in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.