HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGİNEERİNG

BBM204 PROGRAMMING LAB.

ASSIGMENT 5

NAME, SURNAME: Daghan Emre Aytac

IDENTITY NUMBER: 2128051

SUBJECT : Shortest Path Algorithm

DATE DUE: May 15, 2015

ADVISOR: TAs. Levent Karacan

Programming Environment：Eclipse IDE for Java Developers

1.Finding K Shortest Paths:

What's the Problem:

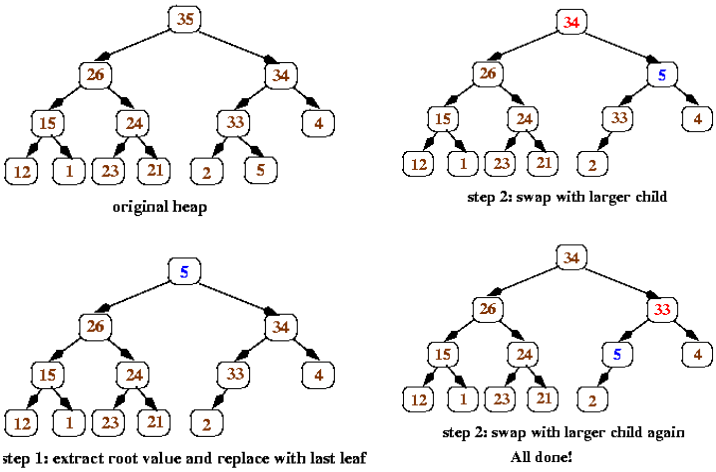
K Shortest Path problems wants to us to find k shortest way to get destination in efficient way. Nowadays, most of electronic, mekanik, and biolojik innovations such as robot motion planning, highway and power line engineering, and network connection routing, navigation systems and so on face with that problem. In these fields, programs needs more flexibility to handle with some of problems. Because a strait algoritm can cause a failure of that tecnologic innovation. So, to have an substitute way to get solution is better way than to have only one solution in technology.

(Referance: http://i.stack.imgur.com/9gFLP.png)

That map is a simple example for navigation technologies. There can be lots of unexpexted factors to affect this one solution such as traffic jam, traffic accedents, natural events. In that situation finding k shortest path would be saver.

2. Algorithmic Approaches to find solution for that problem:

While, Dijkstra and BFS algorithms can find only one way(shortest way), minimum priority queues can easily find number of ways how many that you want in efficient way. Priority queues hold priorities of your datas. To be obvious, you can insert an random priority data that you don't know how much. Whenever you want to get an data from that priority queue, that algorithm gives you one of the highest priority item of them. Priority queues are implemented using heaps. Heaps are represented like balenced binary tree in theoric. In implementation, they are produced from arrays. There are two basic operations to make an priority queue. One of them is insertion to heap. That operation consist of two step which are inserting the the new value at the and of array and swapping with its parent if it is bigger then its parent. When parent "i" index of the array its left child is stored at "2i" and the right child is stored "2i +1" . Another operation to manage the heap is removing from array.It has three steps to make that. One of that replacing the root(first priority value) of the array with the value at the end of the array. Second, remove that ex priority one from the end of the array. And lastly, swapping with the bigger child. In that part we should get bigger child to process.



A visualisation of removing binary heap

3.Solution of the Assigment:

I used an arraylist to hold all vertices of the graph. These vertices also linked to their edges with another linked list that is called targetEdges. And also, according to my design these edges also hold their "targetVertex" because of their property from being "Edge" class. In that case you can travel from first vertex to last vertex of graph. And also i used an priority queue(from "java.util") that uses a comparator to implements Comparator class from "java.util" My algorithm adding target edges to the path with some of operations, and then adding the priority queue that is called "minHeap". When you pull a path from priority queue, it gives you the least total distance path from queue.You add edges to the paths, and romeves, and when you go like this way 2 thing can finish that loop. First one is when you find the destination vertex from pulling paths the program stops to adding more paths to the priority queue.And, there is no more paths in the queue, the program will otomatically finish. Another way, when it is given a "K" value from user, and the size of "shortestPaths" arraylist reaches to the "K" the program will stop the loop.

4.To Compare My Solution With Another Solutions:

My algorithm contains lots of "for" and "while" crowded loop. So this situation, makes the algorithm very slow. In first and third input, the program gave a fast output . However, Yen's algorithm (O(n^3) complexity) and Eppstein's algorithm(O(n*m) complexity for n nodes and m edges in directed graph) are very fast then mine.

References:

http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html

https://www.ics.uci.edu/~eppstein/pubs/Epp-SJC-98.pdf

http://www.programcreek.com/2009/02/using-the-priorityqueue-class-example/

http://interactivepython.org/runestone/static/pythonds/Trees/heap.html

http://www.java-tips.org/java-se-tips/java.lang/priority-queue-binary-heap-implementation-in-3.html