CMPE 343 HW 4 Report

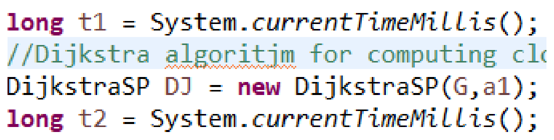
4865115490,
Fatih Mehmet Yiğitel,
Section 02,
HW4

Testing:

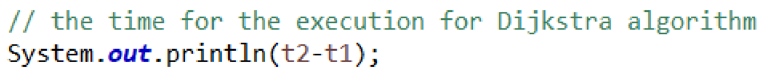
2nd Task:

In the testing part I use just turkeymap.txt because usa.txt input type is not valid for my implementation. I check the internet but I couldn't find a dataset. So I create my files with some for loops and random class in java.

I create small.txt and big.txt. small.txt has 400 vertices and 798 edges. big.txt has 400.000 vertices and 799998 edges. So using System.Millis.Current() method before and after Dijsktra algorithm I find the passed time for the execution it.



```
long t1 = System.currentTimeMillis();
//Dijkstra algoritjm for computing clc
DijkstraSP DJ = new DijkstraSP(G,a1);
long t2 = System.currentTimeMillis();
```



```
// the time for the execution for Dijkstra algorithm
System.out.println(t2-t1);
```

Console Input	Vertices number	Edges number	Execution time(avg)
400bin.txt kamiorvy bwcdumai	400000	399999	140
big.txt pmvcgxgp xkaxdktq	400000	799998	414
minifile.txt ar mr	10	9	1
sample_input2.txt İstanbul Ağrı	6	9	1
small.txt xzyuhzgg bflgbubu	400	798	2
turkeymap.txt Kırklareli Iğdır	81	198	2
turmap2.txt Kırklareli Iğdır	81	82	2

How does your program scale with changing input size?

Firstly, if we keep size of vertices same and we increase Edges number you will see the execution time is increasing also You can observe that with comparing 400bin.txt and big.txt. One of them has execution time of 140, big.txt has execution time of 414. So, it is increased also. Two times so we can say the number of Edges effect the almost algorithm linearly. E.

Secondly, we compare big.txt and small.txt the edge number ratio between them is almost 1000. And also vertices number ratio between them is almost 1000.

Big.txt edges/small.txt edges = 1000. So execution time is should increased 1000 times but it is not. It is just 200 times increased so we can say there is some calculation about number of vertices also. We should check the number of vertices. It can be log V because log 400000 is 5.60 and log 400 is 2.50 so with using these information we can say our algorithm complexity for the worst case is to O (E l o g V).

• Does it have the expected computational behavior as explained in the textbook?

Yes it is excepted results for our textbook. Textbook says the time complexity of Dijkstra is time proportional to E log V (in the worst case).

• Where is the performance bottleneck of your program?

Dijkstra's algorithm solves the single-source shortest-paths problem in edge-weighted digraphs. It just for the single source problem if we want to find shortest path on destination to source we have to reverse the graph . and it uses the extra space proportional V. I think these are the bottleneck of my program.

• What can be done to speed it up ?

We can use hash table for the storing strings in the input file instead of array.

If we have small range weights Dial’s Algorithm Dijkstra’s algorithm can be modified by using different data structure, buckets, which is called dial implementation of Dijkstra’s algorithm. (Optimized Dijkstra for small range weights). We can decrease the complexity O(E + WV) where W is maximum weight on any edge of graph. So we can say we should use buckets for the Dijsktra implementation.

Time Complexity of normal Dijkstra's Algorithm is O (V 2) but with min-priority queue it drops down to O (E l o g V) for the worst case. So we can say we should use min-priority queue for the Dijsktra implementation.