August Sandoval

1.The motivation is to create a implantation of dijkstra’s algorithm and us it on a practical setting such as on a map. Using a linked list representation of a heap and a regular heap priority queue.

2. How I implemented the remove-min and decrease-key operation for the list is treat the list like a array or vector and sift the nodes in to place, by using iterators since in linked list I cannot directly access a element with an index and remove to first element in the list afterwards. For the decrease-key I will look for the node that needs to be decreased then change its key then swap it with the last element and sift it into place.

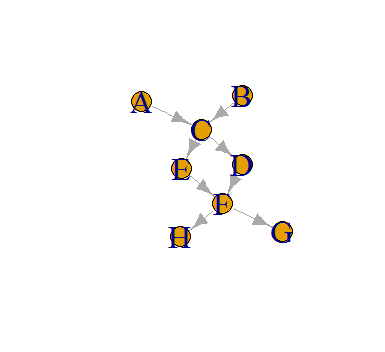
For heap I used the already implemented heap in c++ make\_heap on a vector contaiting the elements. Make\_heap already sorts the elements into place and remove the first element in the vector. For decrease-Key is did the same thing as the list representation, but have sort\_heap put the nodes in place.

The runtime for the two implementation is the heap should be faster then list. Being close to nlog(n) for heap and close to linear for list.

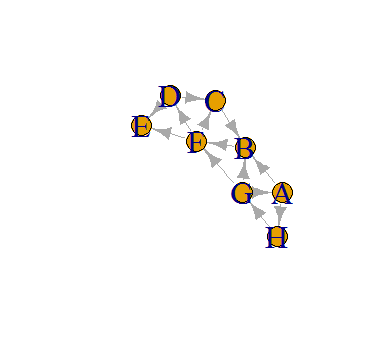
3.

A. test Graphs

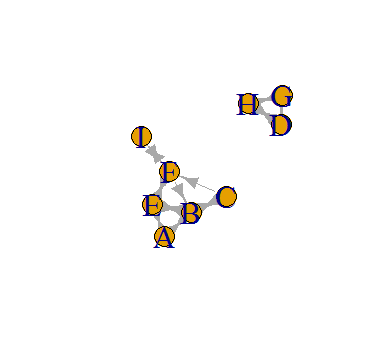
Graph 2.



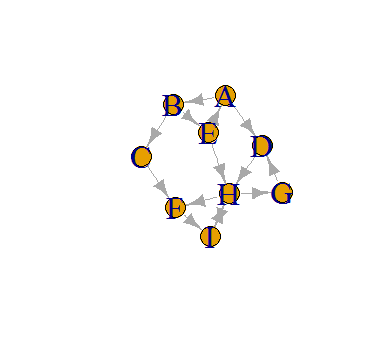
Graph 3.



Graph 4.

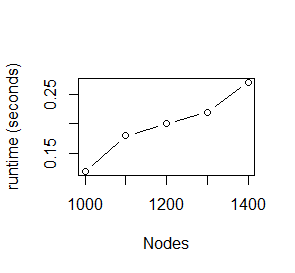


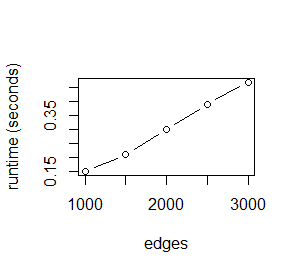
Graph 5.



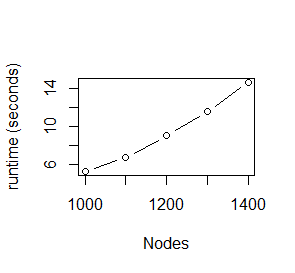
B.

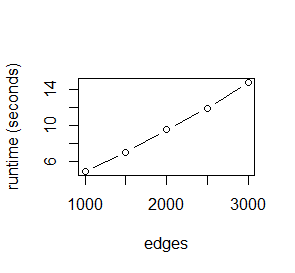
Heap:





Linked List:





4.

Using graph 1, changing one of the edges to be negative result in dikjstra to give the wrong answer. The weights for graph1 are [ (A->B) = 4, (A->C) = 2, (B->D) = 2, (B->C) = 3, (B->E) = 3, (C->B) = 1, (C->D) = 4, (C->E) = 5, (E->D) = 1] to

[ (A->B) = 4, (A->C) = 2, (B->D) = -2, (B->C) = 3, (B->E) = 3, (C->B) = 1, (C->D) = 4, (C->E) = 5, (E->D) = 1]

Results being A(inf), B(3), C(2), D(1), E(6) instead of A(0), B(3), C(2), D(5), E(6) for the distance for the nodes.

5.This lab show how some practical problems are implemented and solved. As well as showing that how these problems are solved can make the difference and save time in the real world.