Jeffrey Lansford

11/13/2019

Lab 7

Introduction

We are tasked with implementing the BFS algorithm for our Graph class. We used the iterative solution for BFS and had to make modifications to the graph class to account for distance from source node. Then, we measure the runtime with R.

Methods

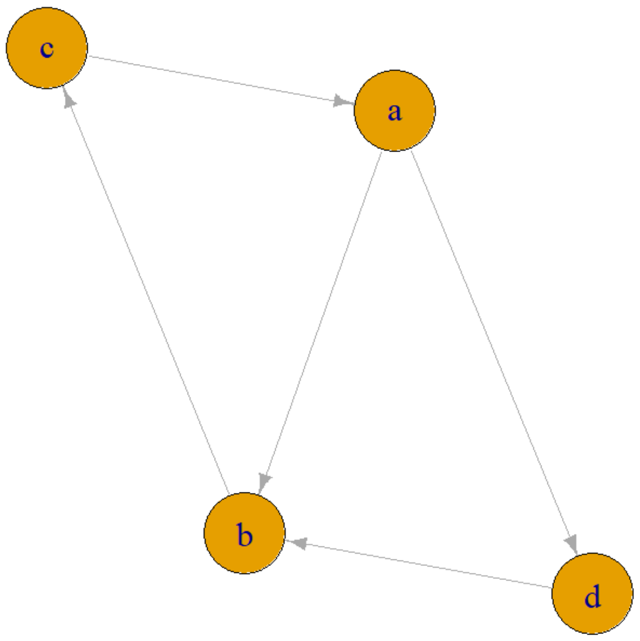
I add a distance variable that is set to infinity and the Graph class can change it with methods I created.

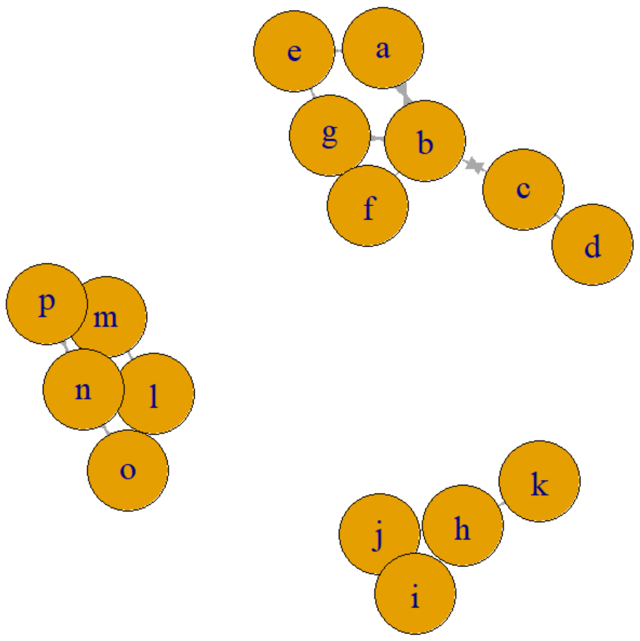
I also implemented a iterative BFS algorithm that our book provide the pseudocode:

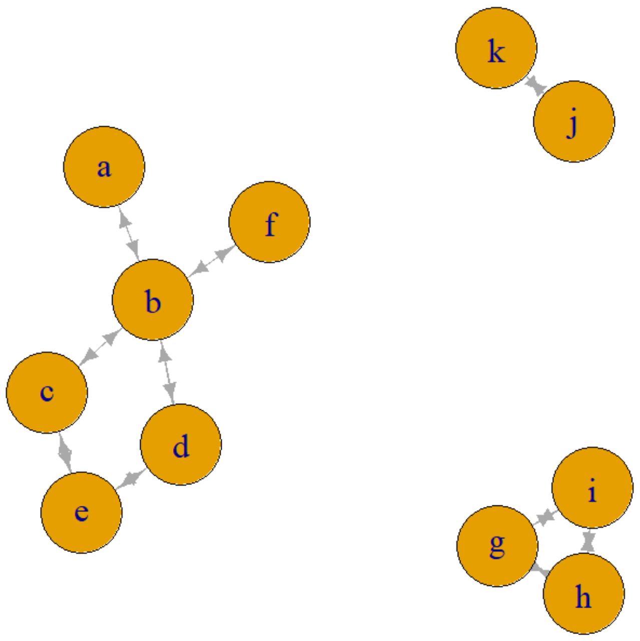
```
BFS(G, s)
    For all u ∈ $V$:
        $\text{dist}(u) = \infty$
    $\text{dist}(s) = 0$
    Q=[s]
    While Q is not empty:
        u = eject(Q)
        for all edges (u,v) ∈ $E$:
            if $\text{dist}(v) = \infty$
                $\text{inject}(Q,v)$
                $\text{dist}(v) = \text{dist}(u) + 1$
```

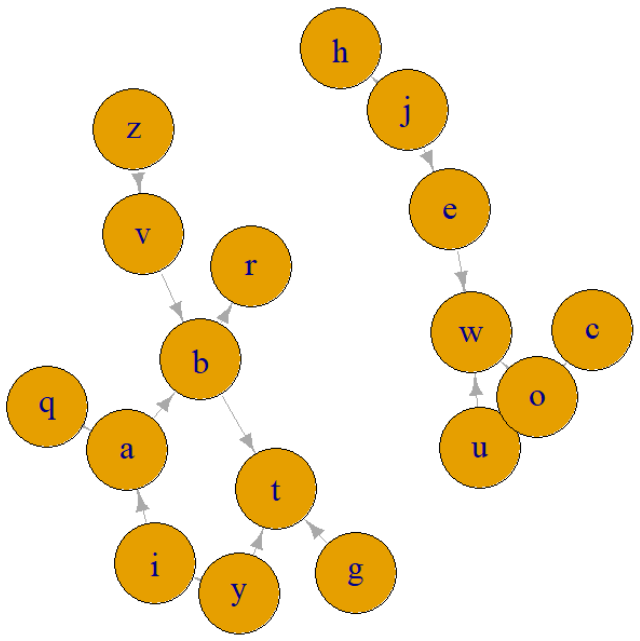
Results

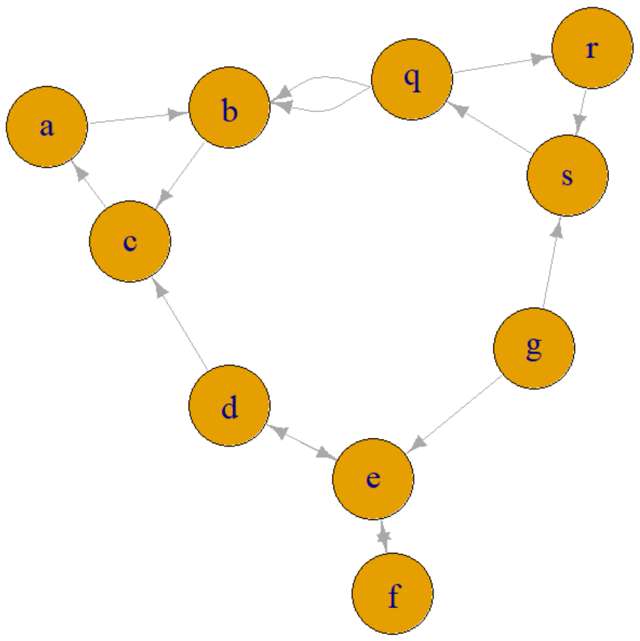
The five graphs I tested with BFS:


c
a
b
d

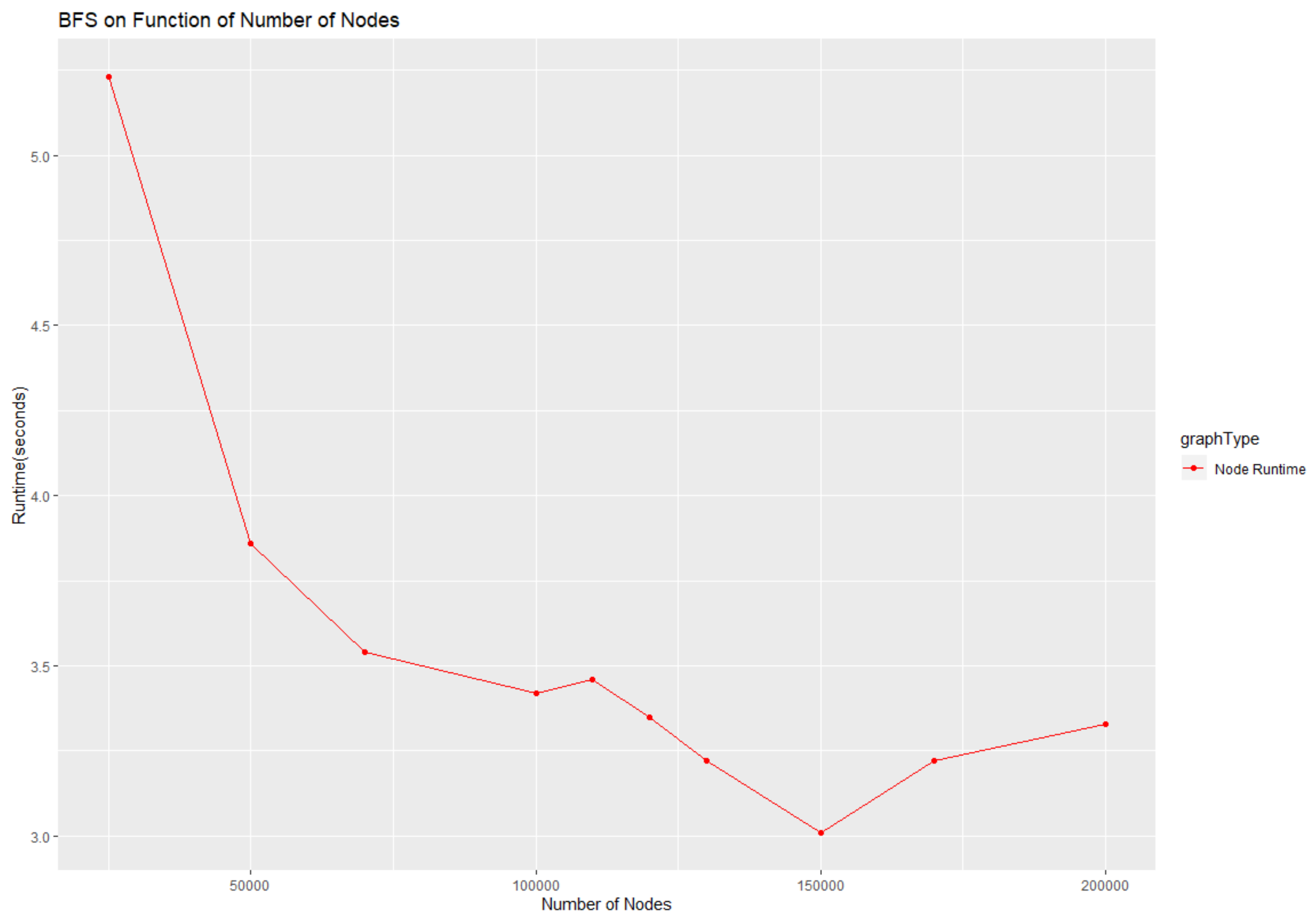

e
a
g
b
f
c
d
p
m
n
l
o
k
j
h
i

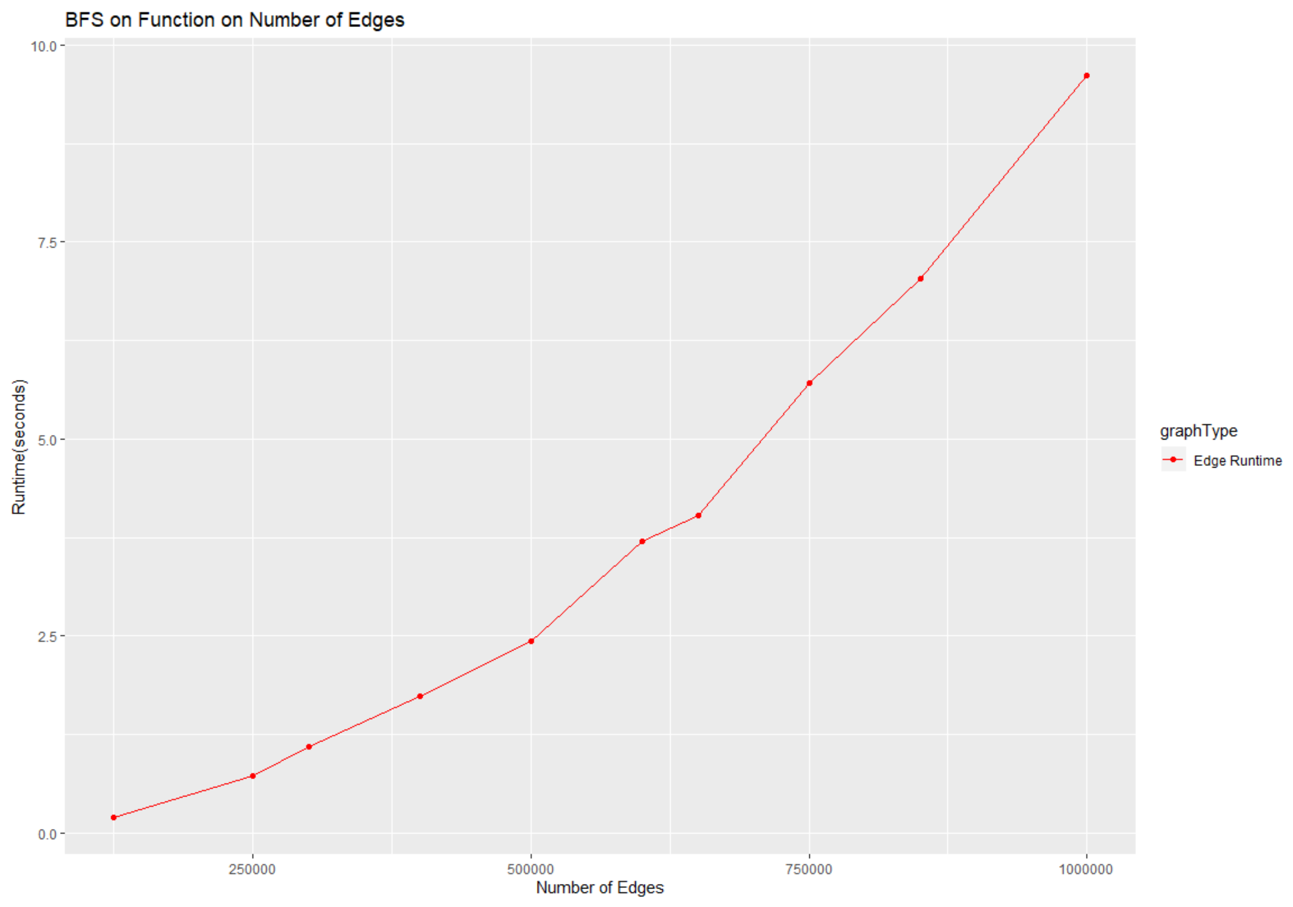

k
j
a
f
b
c
d
e
i
g
h


h
j
z
e
v
r
w
c
b
o
q
a
u
t
i
y
g



Then the two curves of runtime:


BFS on Function of Number of Nodes
Runtime(seconds)
5.0
4.5
4.0
3.5
3.0
50000
100000
150000
200000
Number of Nodes
graphType
Node Runtime



The curve for edges looks like is linear, which is what is expected. Though the node graph is drastically not linear. After looking at graph as the time decrease with more nodes, I believe I found a correlation. How I design the random graphs is that there is a set number of edges, and edges must be larger than or equal number of nodes. Thus, all graphs have the same number edges. The increase of time could be how I go through the adjacently list for each node as with a smaller number of nodes and more edges, your adjacent list becomes bigger. So maybe it can be cause by the way I am doing the for loop. To make my coding simpler, I used a for each loop implemented in c++ and I might be slowing down how c++ iterates through the list.

Discussions

I did not have much issues with this lab. The pseudocode was very straight forward, and I made some helper functions to have it look more like the pseudocode. I expanded on the R code I did on previous labs to have more points on the runtime graphs to give better estimated graphs. My main issue was trying to get the node runtime graph to make any sense. Before, the graph would have little runtime and crazy spikes everywhere. This also led me to the decision to have more points on the graphs. Also, the graphs are on a different machine than my laptop. It is a Ryzen 7 2700x @ 4.0Ghz with 16GB od ram.

Conclusions

Overall, this lab task us to create the BFS algorithm in our graph class. This will help us understand the distance variable of the Graphs as our next Lab deals with Dijkstra's algorithm, which will be more complicated.