Andrew Rutherford

CSCI 3104

CPU: 2.8 GHz Intel Core i7

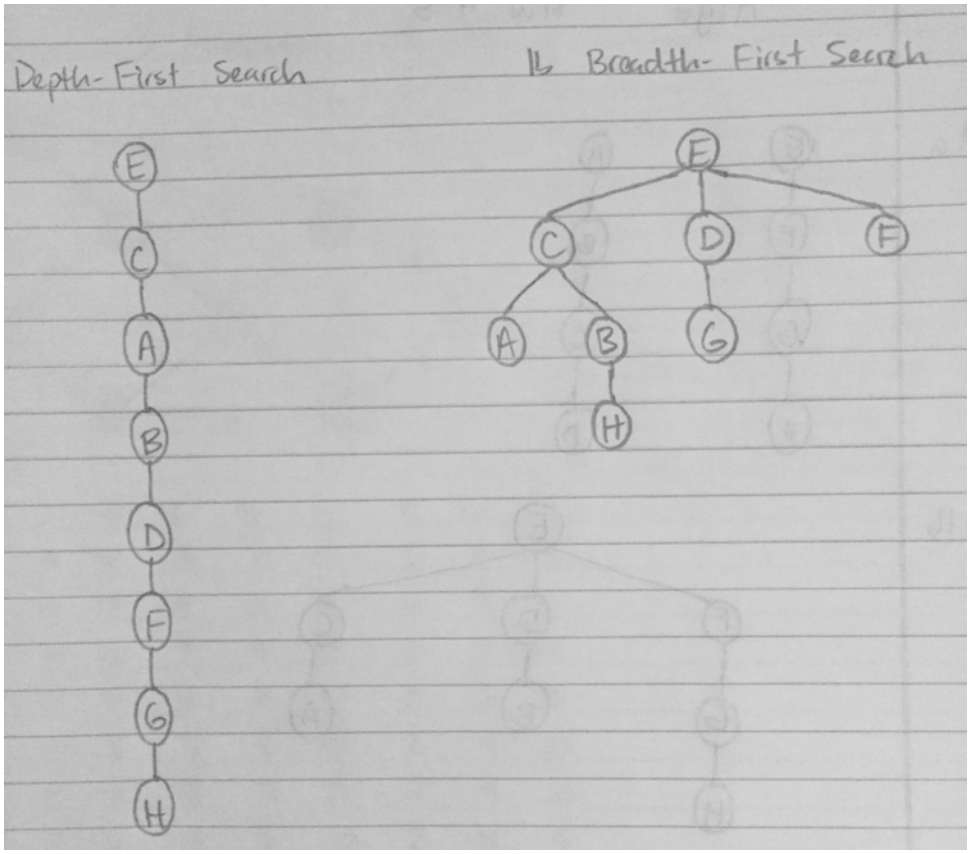
Ram: 16 GB 1600 MHz DDR3

OSX Yosemite

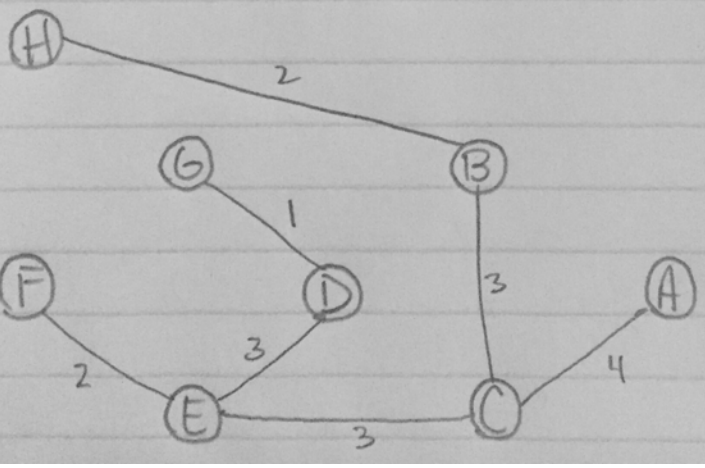
Homework #4

On my honor, as a University of Colorado at Boulder student, I have neither given nor received any unauthorized help.

1.



	Iter.	A	B	C	D	E	F	G	H
E	0	∞	∞	3	3	0	2	∞	∞
F	1	∞	∞	3	3	0	2	8	∞
D	2	∞	12	3	3	0	2	4	∞
G	3	∞	8	3	3	0	2	4	18
B	4	10	8	3	3	0	2	4	10
A	5	7	8	3	3	0	2	4	10
C	6	7	6	3	3	0	2	4	8



2. Perform a Breadth-First Search starting from u, and create variable num_paths(x) for the number of paths from u to x, for all vertices x. If x1, x2, x3,…,xk are vertices of depth I in the BFS tree, and x is a vertex of depth I + 1 such that (x1, x),…,(xk,x) are contained in E. So, num_paths(x) = num_paths(x1) + … + num_paths(xk).

   num_paths(x) = 0 for all vertices x != u and num_paths(u) = 1.

   num_paths(y) = num_paths(y) + num_paths(x) for each edge (x, y) that goes down a level in the tree. BFS is only changed to do one extra operation per edge, so it takes linear time.

3. If P is the shortest path from vertex u to v passing through v0, then between v0 and v, P must follow the shortest path from v0 to v. Also, between u and v0, P must also follow the shortest

path from v0 to u in the reverse graph, which must exist because the graph is strongly connected.

The shortest path from a node u to a node v through node v0 can be found by running Dijkstra's algorithm twice, once on graph G, and once on the reverse graph of G. The time complexity is dominated by looking up all the O(|v^2|) pairs of distances.