This is a guide for how each graph search algorithm works, and will help you as you trace through the example solutions in the Excel file. Have fun!

Also, if you can’t follow this file or need supplement help these videos are pretty useful.

* + BFS & DFS: <http://www.youtube.com/watch?v=zLZhSSXAwxI>
  + Dijkstra’s: <http://www.youtube.com/watch?v=UG7VmPWkJmA>

**Depth First Search:** Worries about getting from a to b in graph.

* Keep up with a visited nodes stack
* Keep up with a node path stack
* Move from current node to next node by alphabetical priority
* Lower letter gets priority
* Do not newly visit nodes you already have in your visited stack
* When you have no more options of nodes to visit from current node move off your path stack until another option appears. Keep visiting nodes and removing off and putting on the path stack until either desired node is found or all nodes in graph has been visited.
* SIMPLE ENOUGH RIGHT☺.

**Breadth First Search:** Worries about getting from a to b in graph with the least amount of nodes.

* Have a queue list to keep up with paths taken
* Have a dequeued list to keep up with expended paths.
* List all the surrounding nodes from starting node in alphabetical order
* Ex. B – A, D – A, T – A. if A is a node and nodes B, D, and T are connected to it.
* Once all surrounding nodes are listed the path from that starting node is then dequeued.
* Move your vertex node to the node which is least alphabetically, B – A in this case and look at all surrounding nodes. Once you’ve listed all of these, dequeued it and move to D – A.
* Next, move to T – A path and list all the nodes connected to that and dequeued it. Continue doing this with all the paths listed in alphabetical order. Go one level of paths at a time until you’ve spread out to all possible nodes in the graph from point A, or found your node of interest.
* EVEN SIMPLER RIGHT☺☺…here’s the hard one.

*Go To Next Page:*

**Dijkstra’s Algorithm:** Worries about getting from a to b with the cheapest edge weighted cost.

* Have a queue list to keep up with paths taken and their total edge weight costs.
* Have a dequeued list to keep up with paths that are chosen as least weighted. At any time you choose a path to be the cheapest when at a certain vertex or node, you will put that path into the dequeued list.
* Both this lists are important to keep up with when doing this tracing!!!
* List all edge weight costs in order from least cost to highest cost from starting node and make sure these are put into the queue list.
* Choose the least weighted path and put that path in the dequeued list. Move to that node and list all the edge weight costs from that node. Check to see what weight from the **WHOLE** queue list is the least and choose that as your next dequeued item and node to move to.
* Again, list out all paths and their weights from that node and check against the **\*WHOLE\*** queue list. Whichever path is with the least cost, choose that to go into your dequeued list and add all the paths connected to that node in order from least cost to highest cost.
* If you have two paths with the same cost to choose from, chose the path which is higher up in the queue list.
* If you’re listing two paths from a certain node that have the same cost value then list them in alphabetical order. Such as 18-e-d-a, 18, u-d-a.
* Things to note from the dequeued list:
  + When a certain node in a path is dequeued for the first time then that path is established as the cheapest path from say, node a to node b.
  + For the node you’re interested in, when you dequeued, **\*not queue\*** your node of interest, that is when you’re done with the problem. That path is the answer.
* Again, below are some YouTube videos that really helps in understand:
  + BFS & DFS: <http://www.youtube.com/watch?v=zLZhSSXAwxI>
  + Dijkstra’s: <http://www.youtube.com/watch?v=UG7VmPWkJmA>
  + Any of the related videos from these links are also pretty good. I hope this helps and best of luck. Make sure to practice each of these with a few graphs!