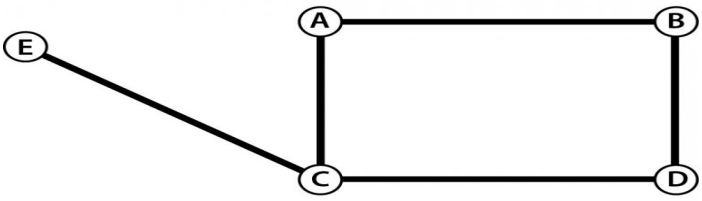
Breadth First Search example (BFS) – How GPS navigation works

What is a graph?

A graph usually looks like the image below and is made up of vertices and edges (represented by lines and circles, respectively).



The objective of a graph is to represent a problem as a set of points that are connected in various ways using edges. With the help of such graphs, we tend to solve our problems by applying various algorithms.

Let's take an example to understand better.

Facebook is a good example to understand graph theory.

Facebook has millions of users. If a person needs to find a friend, he can use an array and search. But that would take a lot of time and memory to search for so many people, making the problem quite complex.

But if the same scenario is represented using a graph, the problems tend to get solved easily. With a graph, you know that these two people are actually friends(Though real-life scenarios are not exactly that simple!).Check this video on how graph theory is used in social networks

Graph theories are frequently used in various other fields, such as maps, e-commerce, and computer games.

What's the difference between a Graph and a Tree?

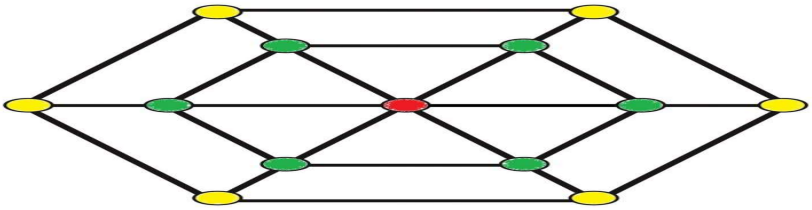
A tree is a special type of graph, i.e., a minimal graph, where there is only one path between two vertices.

So what is Breadth First Search and how does it work?

Breadth First Search (BFS) is algorithm, or in simple terms, it is a method to traverse a graph.

Take a graph with 13 nodes. When Breadth First Search is applied to this graph, the algorithm traverses from node 1 to node 2 and then to nodes 3, 4, 5, 6 (in green) and so on in the given order.

If you consider 1 (in red) as the first node, you observe that Breadth First Search gradually moves outward, considering each neighboring node first.

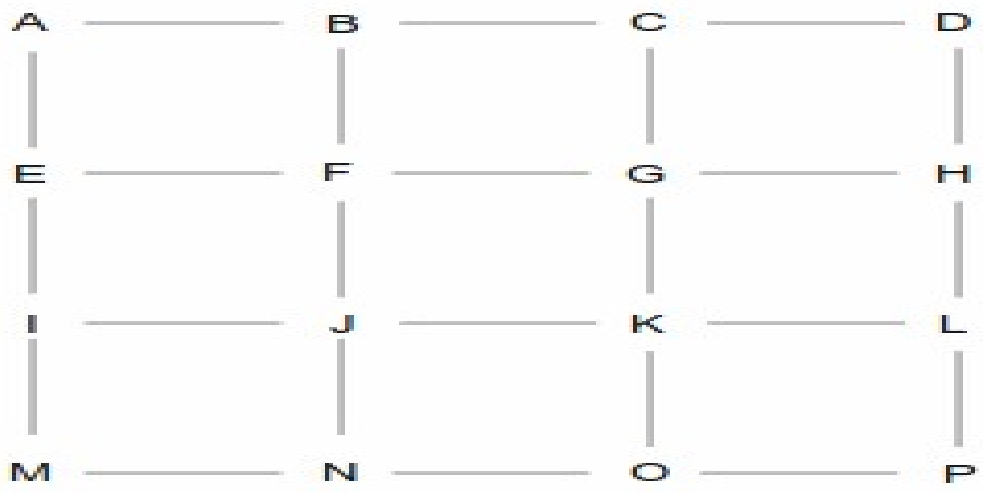


This eventually brings us to the accepted definition of the Breadth First Search algorithm:

"Breadth First search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a "search key"') and explores the neighbor nodes first, before moving to the next level neighbors."

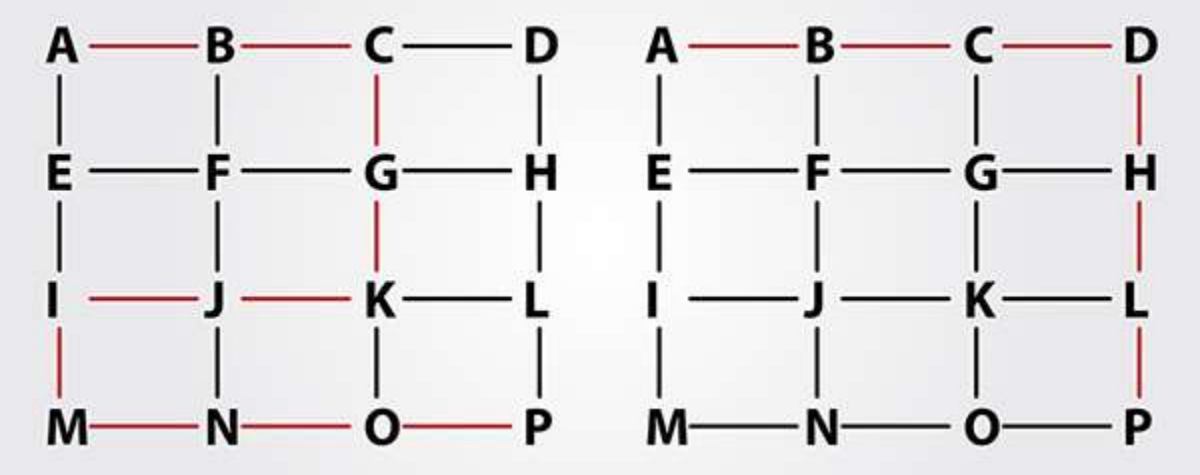
Graph Traversal in Maps

Take a look at this simple "Gridworld" which is used for various graph traversal algorithms. Your digital map considers your world a similar grid, which is made up of intersections connected to each other.



Now for the grid shown, there could be N number of ways to traverse from point A to point P.

Following are two of these N ways in which one can travel from point A to point P.



So how does an algorithm decide which the shortest way to reach a destination is?

Graph Traversal Algorithms!

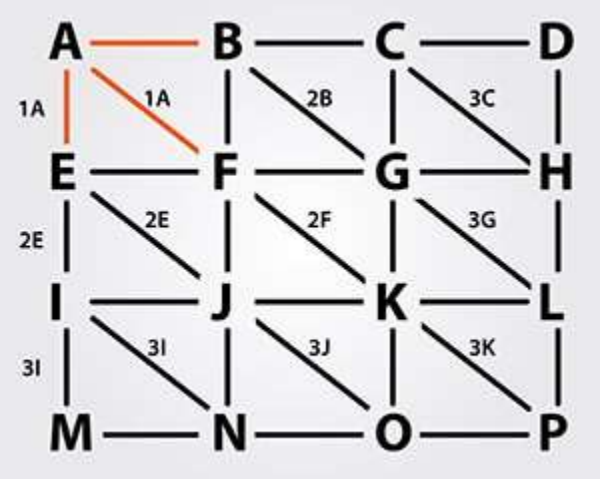
The Breadth First Search algorithm looks at the map as we do; it just can't perceive it completely. When you have to travel from one destination to another, you draw a line from point A to point B, and then chose the road closest to that line. Algorithms repeat the same method choosing the node nearest to the intersection points, eventually selecting the route with the shortest length.

Let's take a simple example of GridWorld given above and try solving it using Breadth First Search search. Assume you need to travel from location A to location P.

Note: Every vertex in the image is given a number, which is the total distance from the source and an alphabet which represents the previous node.

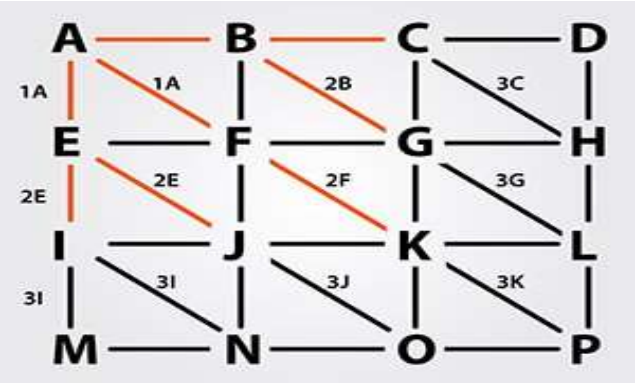
Breadth First Search for GridWorld

Step 1 – Visit neighboring nodes to A, i.e, B, E, and F. The vertex to B would become 1-A and since E and F are also at an equal distance as B, hence vertices to both E and F from A, could be denoted as 1-A too. *There is the no particular order for node preference but to make it simple, we began with B.*



Step 2 – Since all the neighboring nodes of A have now been traced, we will mark "A" as visited.

And take the next visited node as the source node, which in this case is node B. Visit all the adjacent nodes to B, which are C (2B) and G (2B). Node F is considered in the previous step from A, hence it is not visited again. Once all neighboring nodes from one node are visited, mark them as visited and move to the next step.

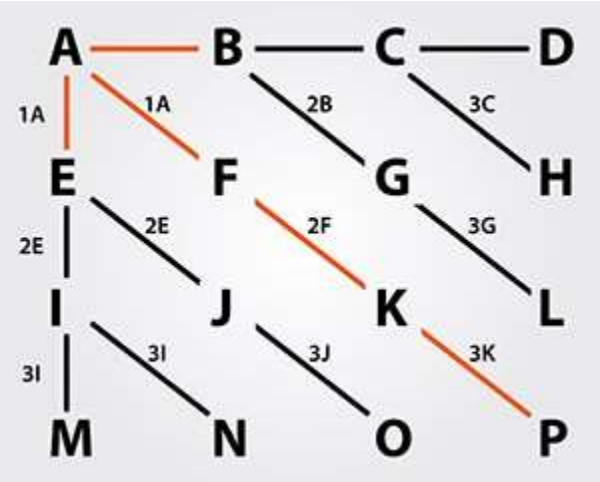


Step 3 – Visit all neighboring nodes of Node E, which are I (2E) and J (2E), and mark E as visited.

Step 4 – Visit neighboring nodes of Node F, which is K (2F). Since all the adjacent nodes have been visited by either B or E, mark node F as visited.

Step 5 – Repeat the process until all the nodes on the grid are visited at least once.

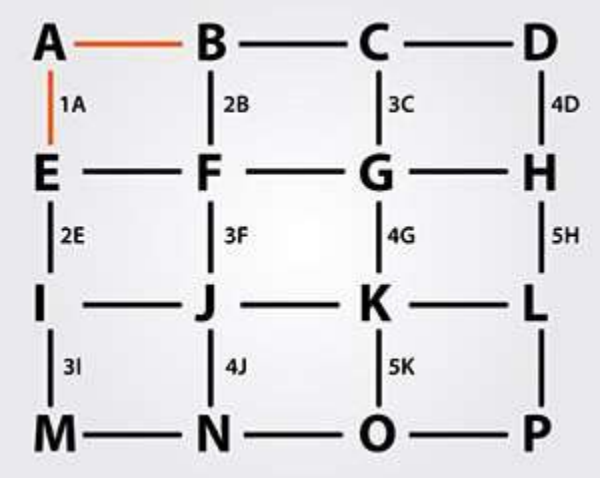
Step 6 – Once all nodes are visited, you would find that the distance required to reach from A to P is 3 and the shortest route is diagonal.



If you remove all the vertices which are not used to connect the nodes, as in the graph above, then such graphs are called minimum spanning trees, where each node is connected to at least one vertex.

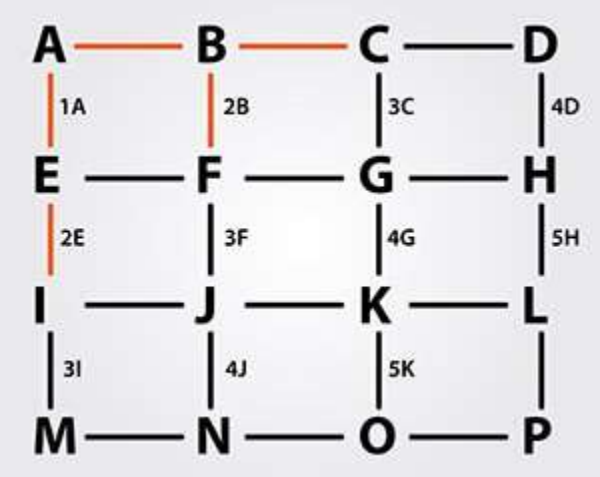
But in the real world, you can't always move diagonally. Most of the portions in between the intersection are occupied by houses, shops, malls, and metro stations. So how does BFS work in such circumstances? Let's understand it with the same GridWorld example, but in this case, the algorithm is not allowed to move or visit a node diagonally.

Step 1 – Consider node A as source and visit all neighboring nodes of A, which are B(1A) and E (1A). Mark A as visited.



Step 2 – Visit all neighboring nodes of B, node C (2B) and node F (2B), and mark B as visited.

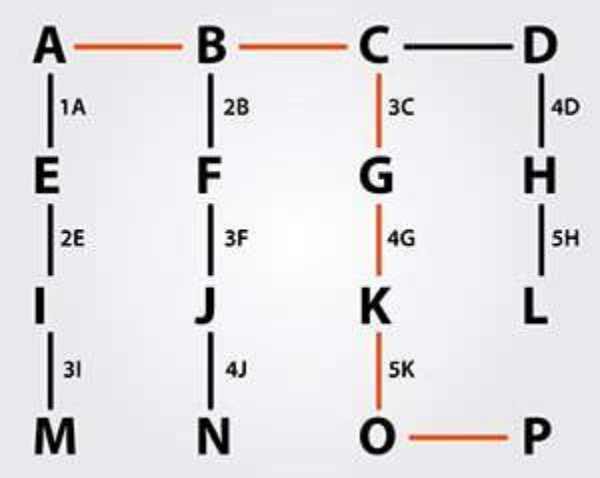
Step 3 – Visit neighboring nodes to E, since F is already visited by B. Visit node I (2E) and mark E as visited



Step 4 – Repeat the steps for all nodes until each of them has been visited at least once. Mark nodes you considered visited.

Step 5 – Remove all the unconnected/unwanted vertices, and convert the graph into a minimum spanning tree connecting each node at least once.

– Highlight the nodes connecting source node A to node P, which has distance 6 and is the shortest path between two nodes.



You now understand why GPS navigation didn't suggest the path A, E, I, M, N, O, P or A,B,C, D, H, L, P though they were equidistant.

Once you've understood the way GPS works, you'd wish the world could be a simple Grid! But to a programmer's disappointment, it isn't. Hence, for a GPS, distance is not the only factor in choosing a route, rather elapsed time, the speed limit on a route, live traffic update, the number of stop signals all has to be taken into consideration. That's why you would find your GPS occasionally suggesting winding state highways to travel instead of the usual national highways.

Most of the GPS or digital maps have evolved over Breadth First Search to A* algorithm due to better complexity over a period of time.

Yet, GPS is one of the most amazing devices. Connected to satellites 12,000 miles above the planet, it calculates your position in real time with more than 50,00,000 possibilities for a particular route.