Breadth First Search

Problem Solving by search

Prof. Deepak C Vegda
email.:deepakvegda.it@ddu.ac.in

Breadth First Search

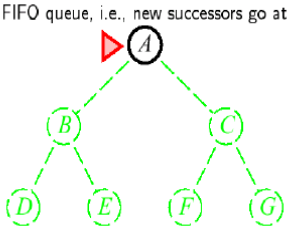
- Searching processes level by level unlike depth first search which goes deep into the tree.
- An operator is employed to generate all possible children of a node.

Breadth-first search

Expand shallowest unexpanded node

Implementation:

fringe is a FIFO queue, i.e., new successors go at end

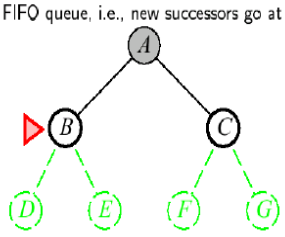


Breadth-first search

Expand shallowest unexpanded node

Implementation:

fringe is a FIFO queue, i.e., new successors go at end

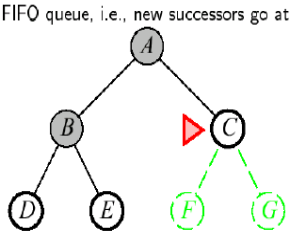


Breadth-first search

Expand shallowest unexpanded node

Implementation:

fringe is a FIFO queue, i.e., new successors go at end

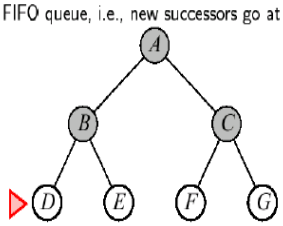


Breadth-first search

Expand shallowest unexpanded node

Implementation:

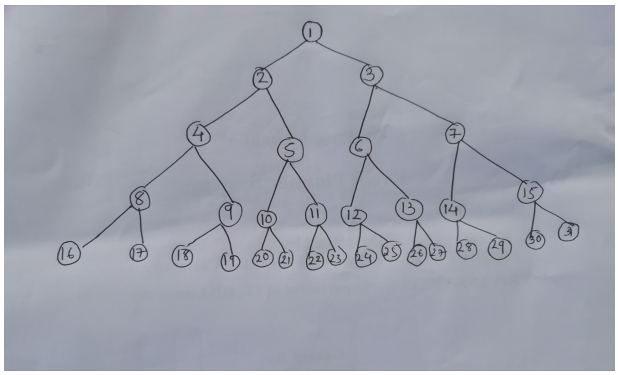
fringe is a FIFO queue, i.e., new successors go at end



Algorithm of Breadth First Search

1. Create a variable called Node-LIST and set it to the initial state.
2. Until a goal state is found or Node-LIST is empty:
   a) Remove the first element from Node-LIST and call it E. if Node-LIST was empty, quit.
   b) For each way that each rule can match the state described in E do:
      i. Apply the rule to generate a new state,
      ii. If the new state is a goal state, quit and return this state.
      iii. Otherwise, add the new state to the end of Node-LIST.

BFS Travel



Time and space complexity

Time Complexity :

$$1 + b + b^2 + b^3 + \ldots + \ldots\ldots b^{d.}$$

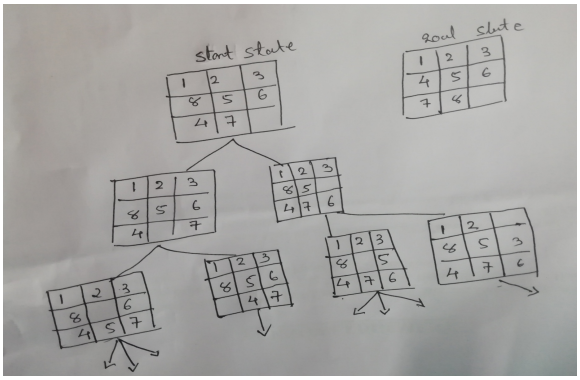
Hence Time complexity = O (b^d)

Space Complexity :

$$1 + b + b^2 + b^3 + \ldots + \ldots\ldots b^{d.}$$

Hence Space complexity = O (b^d)

8 Tile Puzzle



Advantages of Breadth-First Search

i. Breadth first search will never get trapped exploring the useless path forever.
ii. If there is a solution, BFS will definitely find it out.
iii. If there is more than one solution then BFS can find the minimal one that requires less number of steps.

Disadvantages of Breadth-First Search

i. It requires more memory
ii. Searching process remembers all unwanted nodes which is of no practical use for the search.

DFS Vs BFS

DFS	BFS
It require less memory because only the nodes on the current path are stored.	It require more memory because all the tree that has so far been generated must be stored.
It is one in which by luck solution can be found without examining much of the search space at all.	While in BFS all parts of the tree must be examined to level n before any nodes on level n+1 can be examined.
It does not give optimal solution.	It gives optimal solution.
DFS may find a long path to a solution in one part of the tree, when a shorter path exists in some other, unexplored part of the tree.	BFS guarantees to find a solution if it exists. Furthermore if there are multiple solutions, then a minimal solution will be found.
Time complexity: $O(b^d)$ where b : branching factor, d: depth	Time complexity: $O(b^d)$ where b : branching factor, d: depth
Space complexity: O(d) , d: depth	Space complexity: $O(b^d)$ where b : branching factor, d: depth