PUZZLE SOLVING WITH A.I.

Data Analysis Club

Popular Search Algorithms:

- Depth First Search
- Breadth First Search
- A Star
- Complete Search (Not discussed)
- Complete Search with pruning (Not discussed)
- Greedy Search (Not discussed)

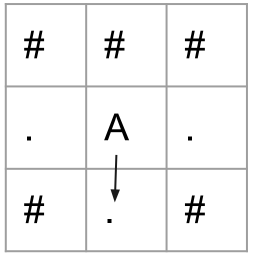
Depth First Search

- In this search algorithm, we go deep exploring a particular direction and return if there is nothing further reachable*.
- In brief, when we are at a particular location, then we try moving towards a fixed position relative to the current position, for example:

  When we are at A, then we always try left up, if it is not possible, then down, then left, then right. What we mean by not possible is that either that location is invalid (doesn't exist) or there is an obstacle or blockage there.

  So we follow: U D L R

Depth First Search



Suppose we are at A, then we first try Up, but since it is blocked, we then try Down. Then from down we try to perform the choose in the same order: U D L R

Did you notice a problem ?

Depth First Search

#	#	#
.	.	.
#	A	#

Now, we are in A, and we go Up.
But then we reach the initial
block from where we came.

This would lead to an infinite loop (a process that doesn’t terminate). So we better store the location which we have been in before (visited*).

→ So, finally the algorithm may be briefly described as choosing where to go according to an order and storing where we have been before so that we won’t go there again.

Depth First Search

Pseudo code:

```
DFS_puzzle(r, c):

visited[r, c] = true
If (r, c) == (r_goal, c_goal) then terminate

If Up is possible and not visited:
        DFS_puzzle(r-1, c)
If Down is possible and not visited:
        DFS_puzzle(r+1, c)
If Left is possible and not visited:
        DFS_puzzle(r, c-1)
If Right is possible and not visited:
        DFS_puzzle(r, c+1)

return
```

Depth First Search

Applications of DFS: -

- Maze solving
- Finding bridges connecting Islands.
- Finding groups of cities/islands connected to each other via 2-way pathways (roads).
- Finding groups of cities/islands from which you may travel between them without getting stuck in any of them, here the roads are 1-way.
- Finding Islands which upon destruction, would cause other islands to get disconnected (no road going there).

Breadth First Search (BFS)

Breadth-first search (**BFS**) algorithm starts at the tree root* (your initial state) and explores all of the neighbors that are directly connected to the current state. It is similar to a web^ of a spider.

Distances of the next states are calculated during the runtime of the algorithm.

It uses the opposite strategy as depth-first search, which instead explores the highest-depth first before being forced to backtrack*.

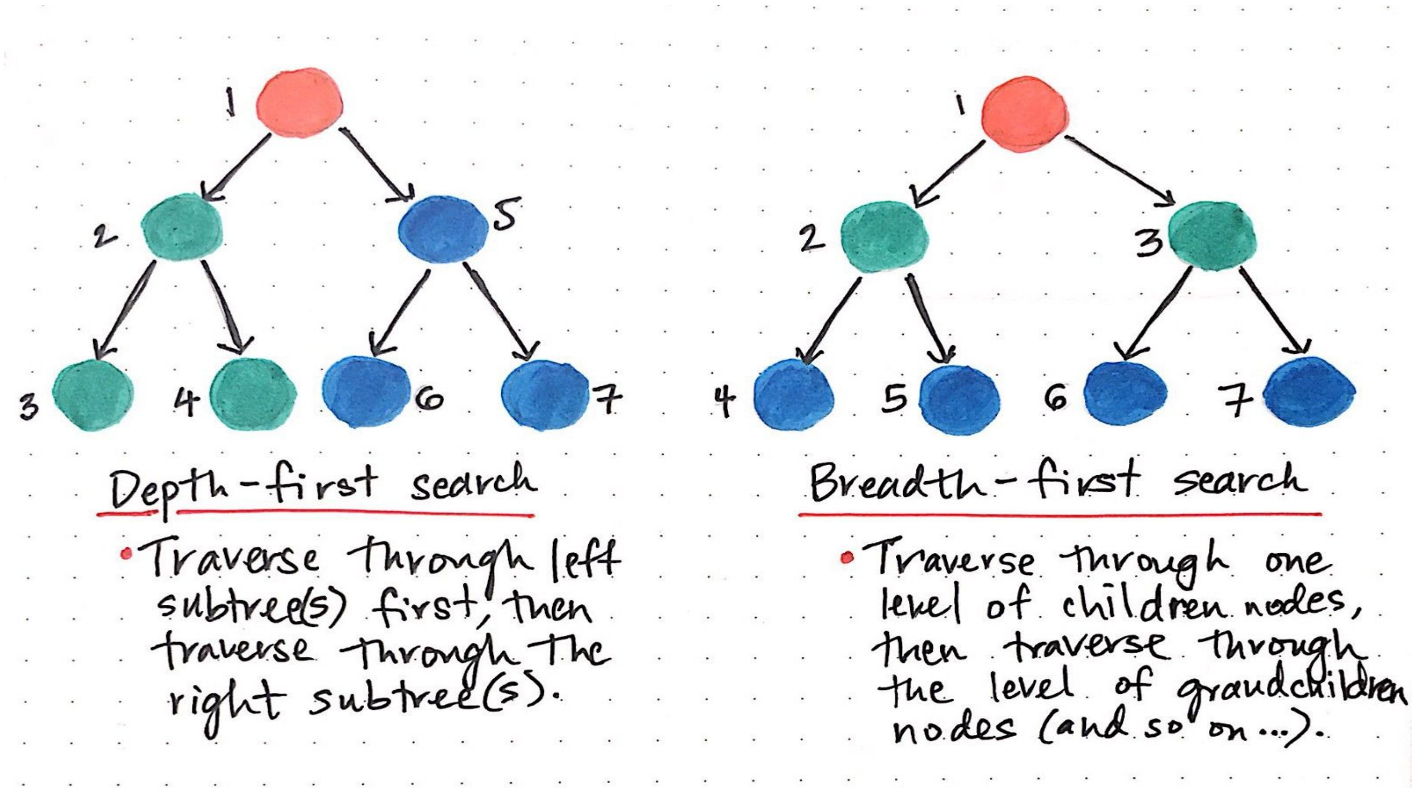
APPLICATIONS/EXAMPLES:

- Used to find the shortest path out of a maze.
- Social Networking Websites use it to find the people that are within a distance of ‘k’ (Friends of friends….) with you.
- Gps Navigator uses Bidirectional A* at your current location to find the nearest neighboring locations from you.

*: terms to be defined, ^: just an analogy

Pseudo Code:

```
BFS(start state):
    Store start state in a VisitedSet
    While(VisitedSet is not empty) {
        CurrentState = VisitedSet.pop()
        For all neighbours of CurrentState{
            If (neighbour is not visited) {
                VisitedSet.add(neighbour)
                distance[neighbour]=distance[CurrentState]+1
            }}
    }
```



Depth-first search

- Traverse through left subtree(s) first, then traverse through the right subtree(s).

Breadth-first search

- Traverse through one level of children nodes, then traverse through the level of grandchildren nodes (and so on...).

A Star(A*) Search Technique:

Motivation:
“To approximate the shortest path in real-life situations, like- in maps, games where there can be many hindrances.”

“Informally speaking, A* Search algorithms, unlike other traversal techniques, it has “brains”. What it means is that it is really a smart algorithm which separates it from the other conventional algorithms. This fact is cleared in detail in below sections.” **--GeeksforGeeks**

This search technique keeps track of future hindrances using some heuristic approximations, and also tries to reach the goal in the shortest time possible. So in a way it keeps track of “future” as well as “past”. This search algorithm is optimal as well as guarantees to reach the desired goals.

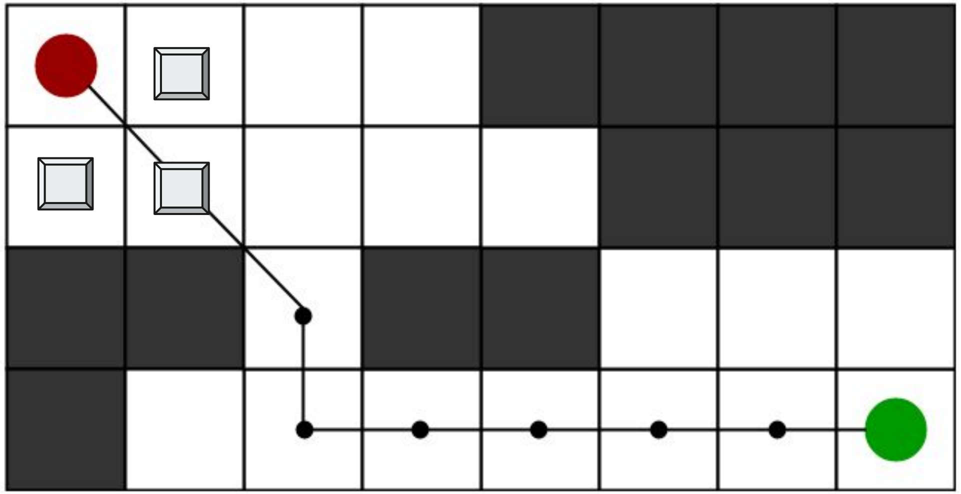
A Star(A*) Search Technique:

Working:

During the runtime of the process, it maintains:

- H distance: Distance of possible next states from the goal.
- G distance: The number of steps already taken from the initial state.

The next step in the process is selected by choosing the state with minimum H+G distance value.



A Star(A*) Search Technique:

Pseudo Code:

```
Repeat
{
        From the current state find the neighbouring possible reachable states.
        Calculate their H, G distance and hence F distance.
        Store them to collection set of unvisited states.
        Select the next “current state” with minimum F distance, and then remove that state
        from the set of unvisited states.
}
```

Get ready for Puzzles Solving !!

The N-puzzle:

Any Idea how to solve this, using the search problems we just discussed?

1		2
7	5	4
8	6	3

The N-puzzle:

Solving the unsolved !!!

Best possible Approach:

- A Star : Take G as the number of steps already taken.
  Take H as the number of blocks displaced at a given state

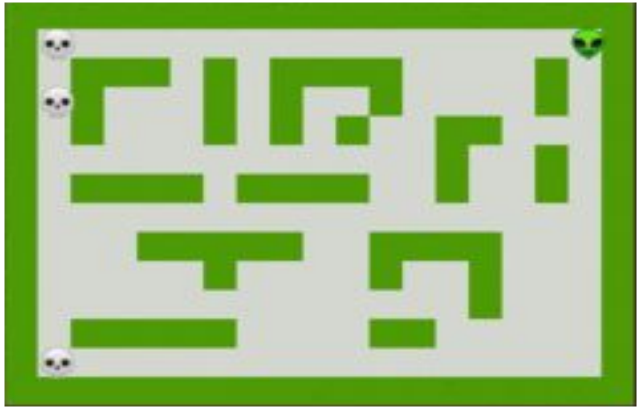
- BFS: Can find the optimal solution but taking more steps as compared to A Star.

Note:

1. Here our single grid formed at certain iteration is one single state.
2. We need to keep track of all visited states, and ensure not to visit that again.

Bot game (By Rakesh Kumar, CSE([yours + just 1] year))

- Idea is that each enemy will use BFS to come towards U.
- You have the control to go in all the 4 directions.
- When you eat the fruit each enemy will try to move away from U.
- Rest all are the blocked states and some graphics.



Bot game (By Rakesh Kumar, CSE([yours + just 1] year))

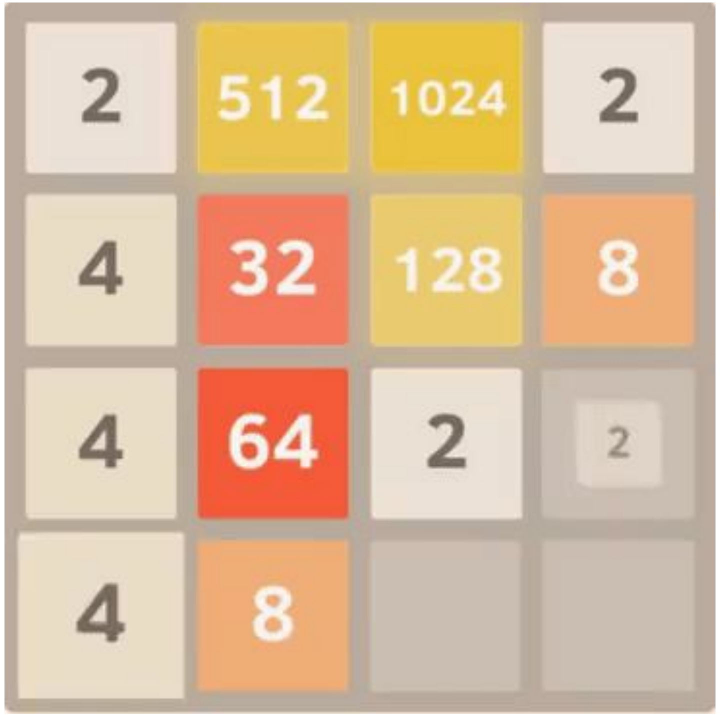
Building technique:

- The bots in the game have two version: one is the skull and one is the devil.
- The skull runs away from the player - i.e when the player picks up a potion, then the bots run away from the player using **Greedy Search.**
- The devil wants to eat the player, hence it finds a shortest path to the player. The bot in this case uses **Breadth First Search.**
- We may also automate the player to play optimally, but it requires advanced AI techniques.

The 2048 puzzle:

The Task :
Suggest the best Idea/method possible to solve it ?

Hint:
Greedy Search



Thank you...

CREDITS:

Ahmed Zaheer Dadarkar
Devansh Rathore
Vaibhav Jindal

Special thanks to: Rakesh Kumar