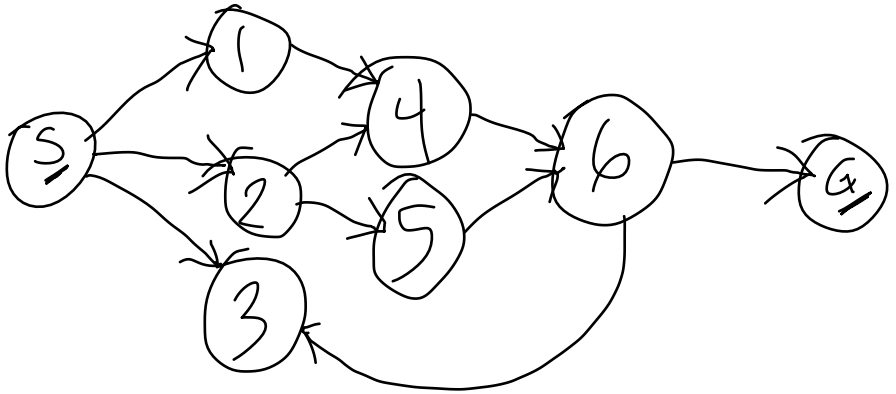
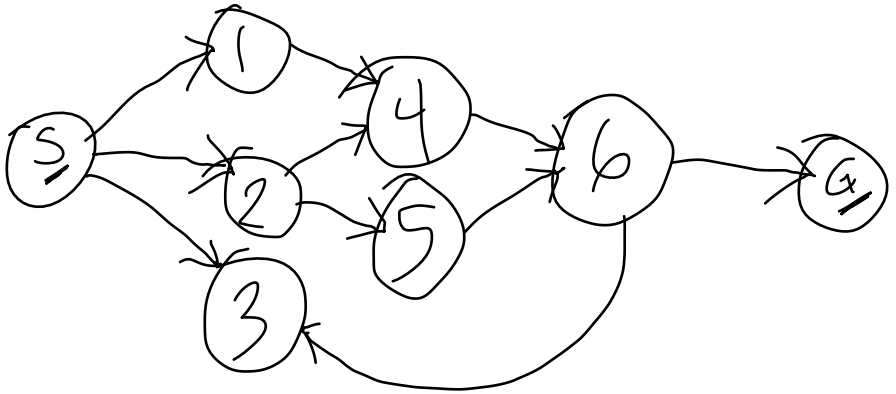
Let's remember BFS.



Draw the search tree



Define Search Properties

1. Completeness
2. Optimality
3. Time Complexity
4. Space Complexity

Breadth-First Properties

Completeness?

Optimality?

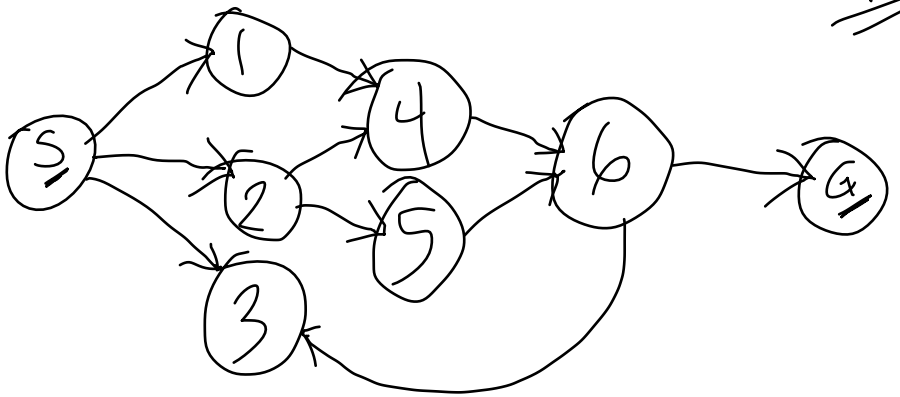
Record your answer here: https://forms.gle/2krMifrptgaEDYyw9

What is the Time Complexity?

What is the Space Complexity?

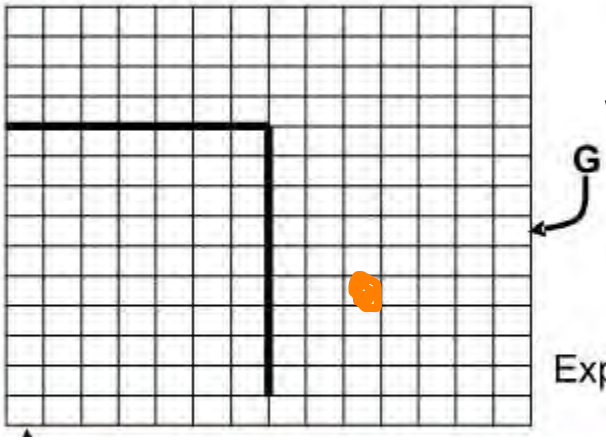
Record your answer here: https://forms.gle/2krMifrptgaEDYyw9

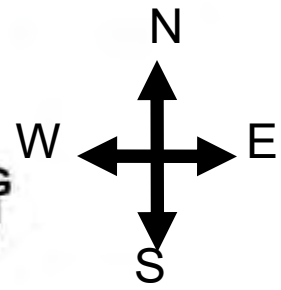
Let's search the same graph with Depth First Search



Maze example

Imagine states are cells in a maze, you can move N, E, S, W. What would plain DFS do, assuming it always expanded the E successor first, then N, then W, then S?





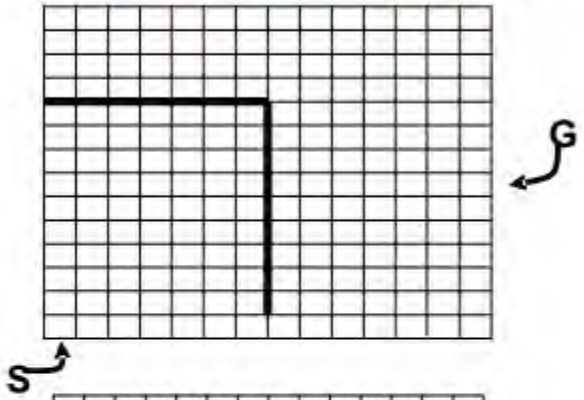
Expansion order E, N, W, S

start

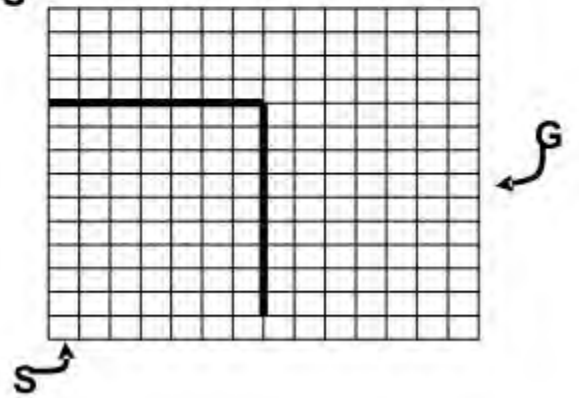
What would BFS do?

Record your answer here: https://forms.gle/2PFoGVQG9Yiu95hV6

Two other DFS examples



Order: N, E, S, W?



Order: N, E, S, W
with loops prevented

Depth-First Properties

Complete?

Optimal?

Depth-First Properties

Time Complexity?

Space Complexity?

Depth Limited Search

```
DLS (Frontier, Successors, Goal?) /* Call with Frontier = {<START>} */

    WHILE (Frontier not EMPTY) {
        n= select first node from Frontier
        Curr = terminal state of n
        If(Goal?(Curr)) return n

        If Depth(n) < D    //Don't add successors if Depth(n) = D!!
                Frontier= (Frontier- {n}) U Successors(Curr)
        Else
                Frontier= Frontier- {n}
                CutOffOccured = TRUE.
    }
    return FAIL
```

Iterative Deepening Search

- Solve the problems of depth-first and breadth-first by extending depth limited search.
- Starting at depth limit L = 0, we iteratively increase the depth limit, performing a depth limited search for each depth limit.
- Stop if a solution is found, or if the depth limited search failed without cutting off any nodes because of the depth limit.
  - If no nodes were cut off, the search examined all paths in the state space and found no solution → no solution exists.

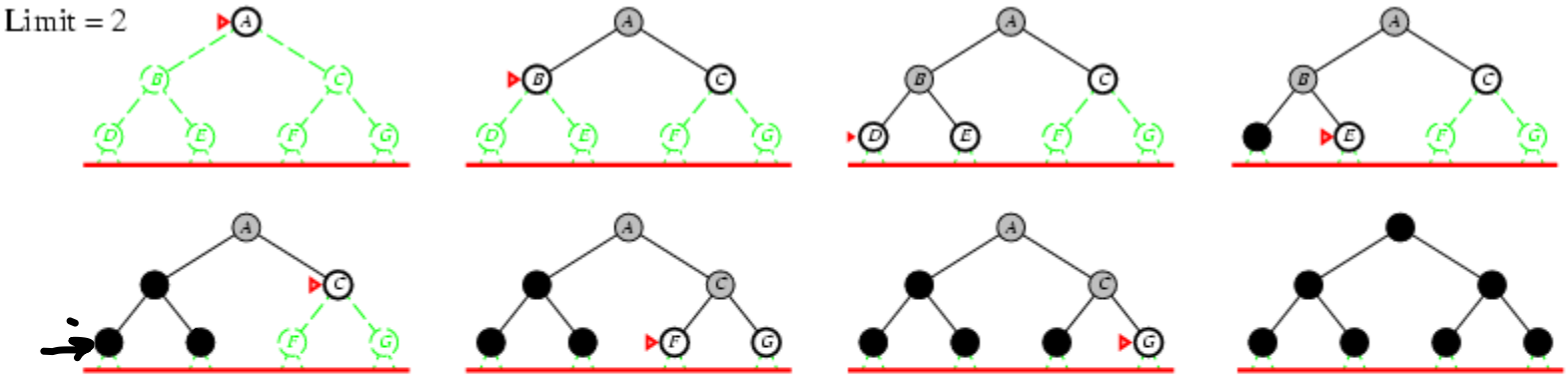
Iterative Deepening Search

Limit = 0

Iterative Deepening Search

Limit = 1

Iterative Deepening Search



Iterative Deepening Search

Completeness?

Optimality?

Space Complexity?

Iterative deepening complexity?

what's better? BFS or iterative deepening DFS?

Path Checking

Recall that paths are commonly stored on the Frontier.

If nk represents the path $<s_0,s_1,\ldots,s_k>$ and we expand s_k to obtain child c, we have

$$<s_0,s_1,\ldots,s_k,c>$$

as the path to "c".

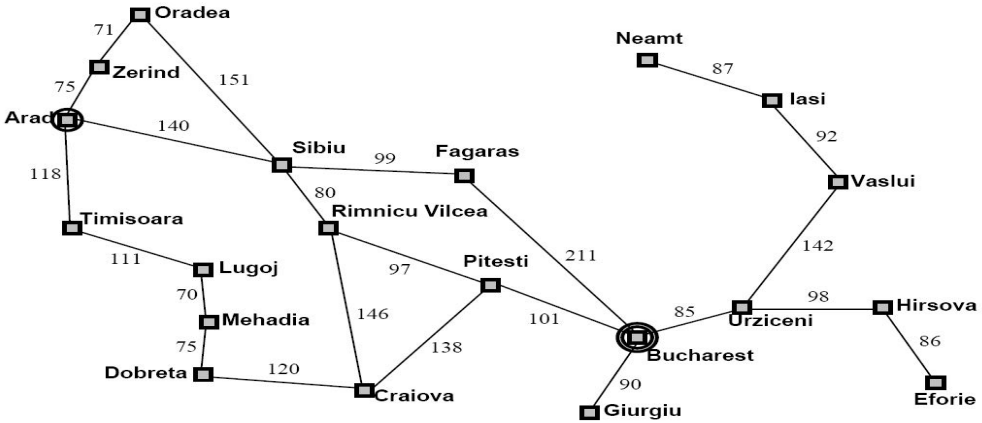
Path checking:

–Ensure that the state c is not equal to the state reached by any ancestor of c along this path.

–Paths are checked in isolation!

Cycle Checking

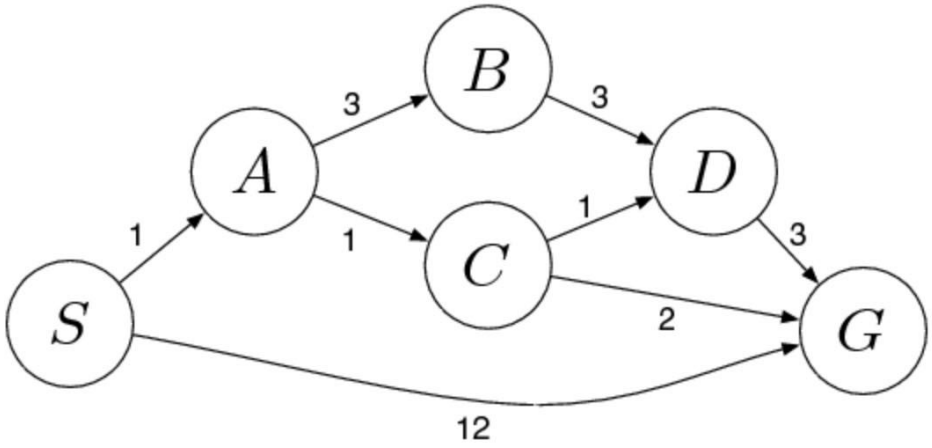
- Keep track of all states previously expanded during the search.
- When we expand n_k to obtain child c
  - Ensure that c is not equal to **any** previously expanded state.
- This is called cycle checking, or multiple path checking.
- What happens when we utilize this technique with depth-first search?
  - **What happens to space complexity?**


Oradea
71
Neamt
87
Zerind
151
Iasi
75
Arad
140
Sibiu
99
Fagaras
92
Vaslui
118
80
Rimnicu Vilcea
Timisoara
142
111
Lugoj
97
Pitesti
211
98
70
Hirsova
Mehadia
146
101
85
Urziceni
75
86
Bucharest
138
Dobreta
120
Craiova
90
Eforie
Giurgiu

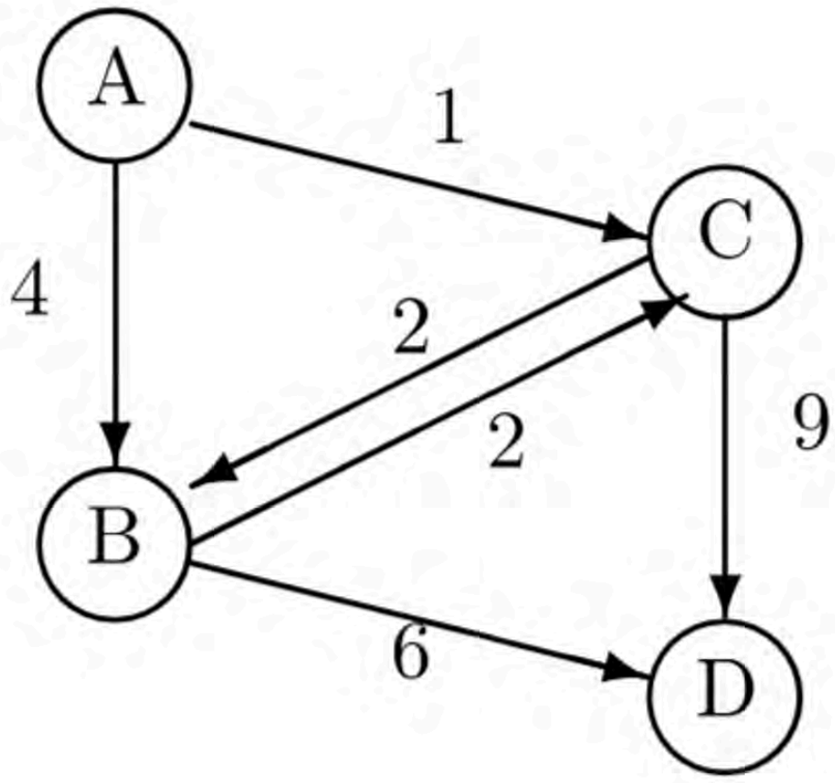
Uniform-Cost Search

- Keeps Frontier ordered by increasing cost of the path *(know a good data structure for this?)*
- Always expand the least cost path.
- Identical to Breadth First Search if each action has the same cost

1. Illustrate the frontier at each iteration of Uniform Cost Search for a search from the start (S) to the goal (G) for the following problem:



Problem!



$$h(A) = 8$$
$$h(B) = 3$$
$$h(C) = 7$$
$$h(D) = 0$$

START = A
GOAL = D