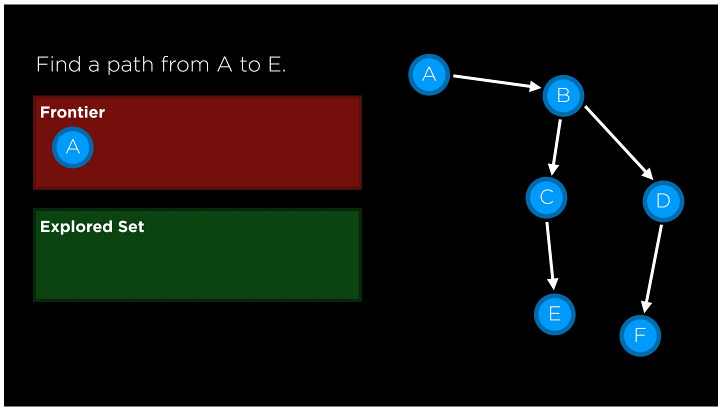
Assignment 1

Problem 1: BFS vs. DFS Implementation and Analysis

Implement both **Breadth-First Search (BFS)** and **Depth-First Search (DFS)** in **Python** for the examples introduced in *Lecture – Search (1)*.

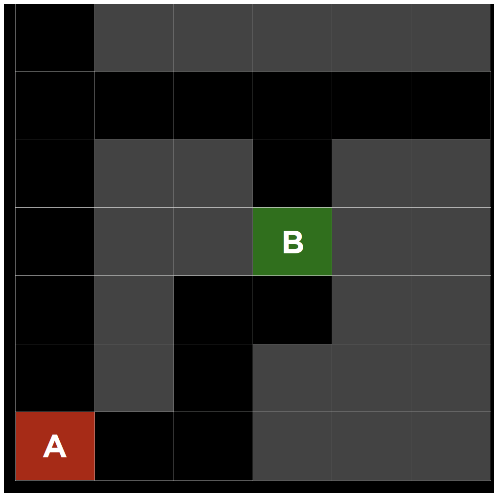
(a). Find a path from A to E using BFS in graph.

(b). Find a path from A to E using DFS in graph.



(c). Find a path from A to B using BFS in grid.

(d). Find a path from A to B using DFS in grid.



Compare the results and write your findings on the differences between BFS and DFS (e.g., traversal order, memory usage, path length).

Problem 2: DLS on the grid.

Using the provided above grid (start **A**, goal **B**), **implement Depth-Limited Search in Python** and run it with two depth limits.

Run DLS with **limit = 4** and **limit = 8** on the grid.

Compare the two runs. Discuss how changing the depth limit impacts success/completeness, node expansions, and path quality. Briefly contrast DLS with your BFS/DFS results.

Problem 3: Pathfinding in a Weighted Graph

Nodes and Heuristics(h(n)):

S: h(S)=7

A: h(A)=6

B: h(B)=5

C: h(C)=1

D: h(D)=4

G: h(G)=0

Edges and costs (g(n)):

S -> A: 2

S -> B: 4

A -> C: 5

A -> D: 10

B -> D: 4

C -> G: 3

D -> G: 2

(a) Draw out the weighted graph.

(b) Find a path from S to G using A* algorithm. Provide step-by-step update. (Follow the lecture notes, each step updates the frontier and explored sets and calculate the cost)

(c) Show the paths explored by your algorithm.

Problem 4: Tic-Tac-Toe Minimax

Consider the current state - s:

O	O	X
X	X	
O		

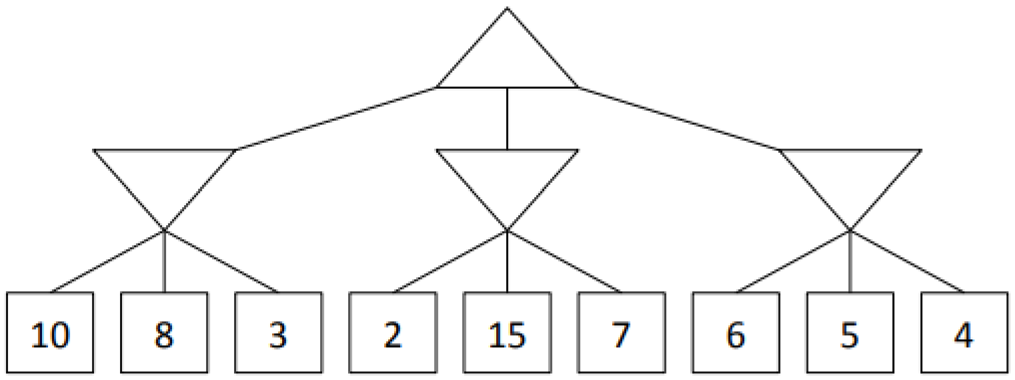
(a) Draw out the game tree from current state to the terminal, include all possible moves for both X and O until terminal states are reached (win, loss, or draw).

(b) How many layers (space complexity) in your tree(including root)?

(c) How many terminal states/nodes here?

(d) Draw out the zero-sum game tree (follow problem 5), label all minimax values.

Problem 5: Consider the zero-sum game tree shown below. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Assuming both players act optimally, fill in the minimax value of each node. Fill in the values of the states on the maximizers and minimizers



Which nodes can be pruned from the game tree above through pruning? If no nodes can be pruned, explain why not. Assume the search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.