AI asgn 1

Himanshu Raj (2022216)

Q1

a) A* - $f(n) = g(n) + h(n)$

Node expanded	g(n)	f(n)	Frontier (priority queue)	Visited nodes
S	0	8	B(2), A(5), C(13)	S
B	1	2	A(5), F(6), D(9), C(13), G3(13)	S, B
A	3	5	F(6), D(9), C(13), G1(13), G3(13)	S, B, A
F	3	6	D(8), C(13), G1(13), G3(13)	S, B, A, F
D	4	8	E(7), G2(9), C(13), G1(13), G3(13)	S, B, A, F, D
E	6	7	G1(8), G2(9), C(13), G1(13), G3(13)	
G1	8	8	G2(9), C(13), G1(13), G3(13)	S, B, A, F, D, G1

Shortest path - S → B → A → F → D → E → G1

b) Uniform cost search

Node expanded	g(n)	Frontier (priority queue)	Visited nodes
S	0	B(1), A(3), C(5)	S
B	1	A(3), F(3), C(5), D(5), G3(13)	S, B
A	3	F(3), C(5), D(5), G1(13), G3(13)	S, B, A
F	3	D(4), C(5), G1(13), G3(13)	S, B, A, F
D	4	C(5), E(6), G2(9), G1(13), G3(13)	S, B, A, F, D
C	5	E(6), G2(9), G1(13), G3(13)	S, B, A, F, D, C
E	6	G1(8), G2(9), G3(13)	S, B, A, F, D, C
G1	8	G2(9), G3(13)	S, B, A, F, D, C, G1

Shortest Path - S → B → A → F → D → E → G1

c) Iterative Deepening A*

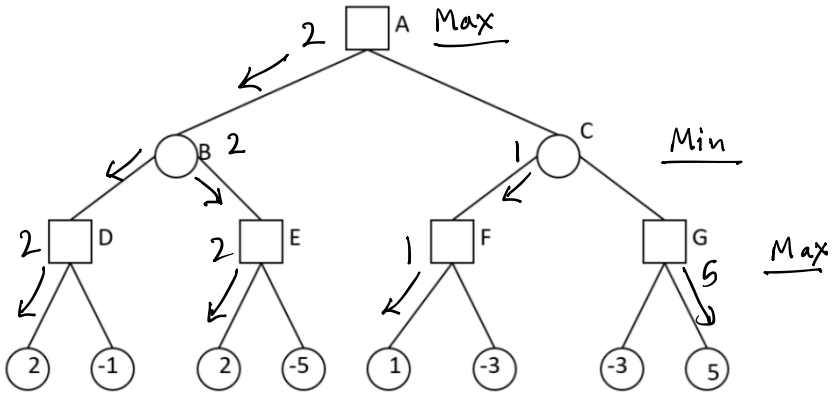
Selected threshold is 8 [we'll explore till node's f(n) value is 8]

Node expanded	g(n)	f(n)	Frontier (priority queue)	Visited nodes
S	0	8	B(2), A(5), C(13)	S
B	1	2	A(5), F(6), D(9), C(13), G3(13)	S, B
A	3	5	F(6), D(9), C(13), G1(13), G3(13)	S, B, A
F	3	6	D(8), C(13), G1(13), G3(13)	S, B, A, F
D	4	8	E(7), G_2(9), C(13), G1(13), G3(13)	S, B, A, F, D
E	6	7	G1(8), G_2(9), C(13), G1(13), G3(13)	
G1	8	8	G_2(9), C(13), G1(13), G3(13)	S, B, A, F, D, G1

Shortest path - S → B → A → F → D → E → G1

Q2

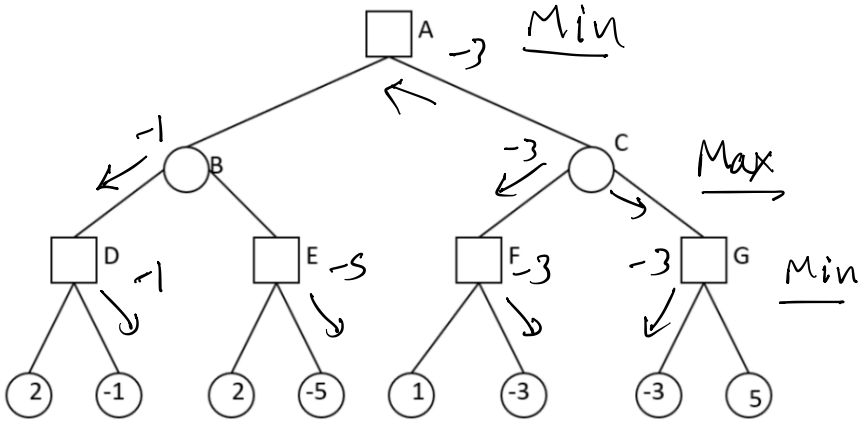
a)



Best moves - A → B → D, A → B → E } both are equally good

I st player

Min- max algo

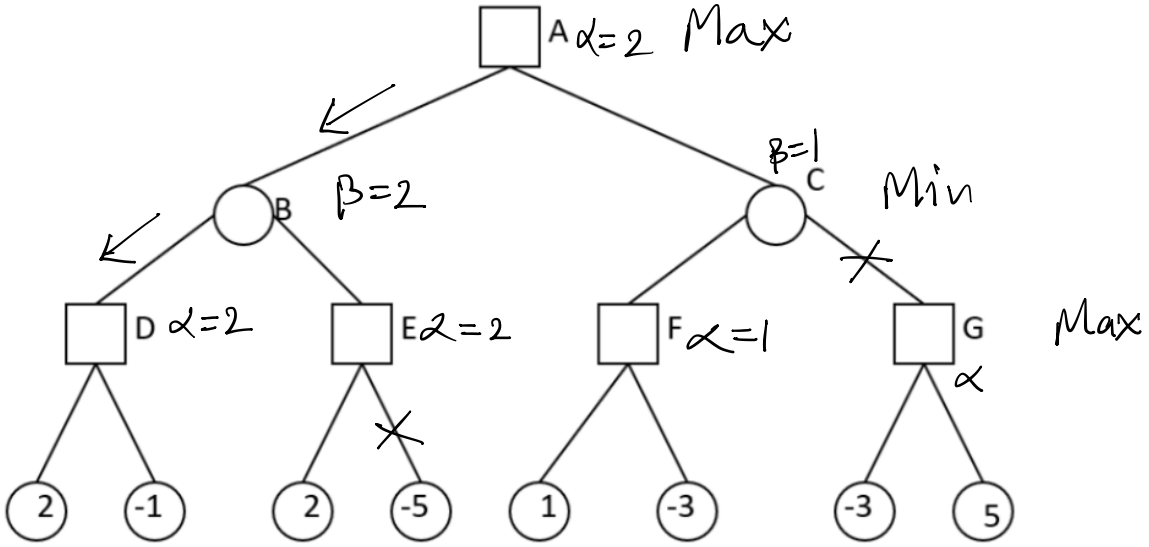


Best moves - A → C → G } both are equally good
→ F }

IInd player

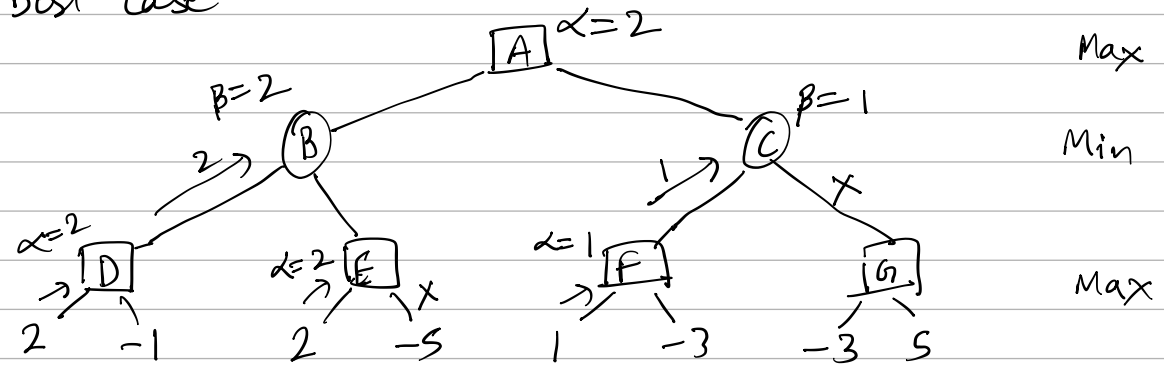
Min- max algo

α-β pruning



Moves — A → B → D

b) Best case -

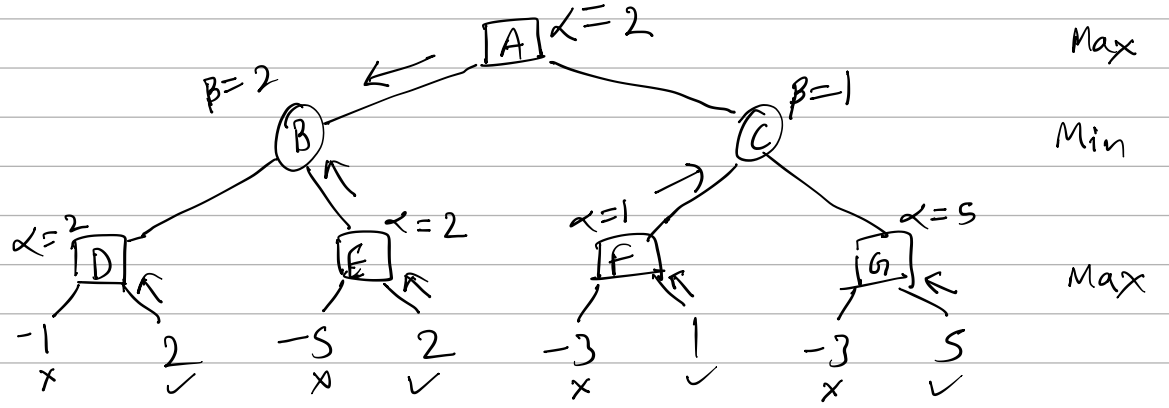


This is the best case because we have pruned the max possible branches.

1st prune - E — -5 (because 2 was the min we could take via B)

2nd prune - C — G (because we could already take 2 to A via B)

Worst case -



We reshuffled the best case to make it so that we are never sure of next branches and are unable to prune any branch at all.

CSE643: Artificial Intelligence
Assignment 1: Search Algorithms

Himanshu Raj (2022216)

September 17, 2024

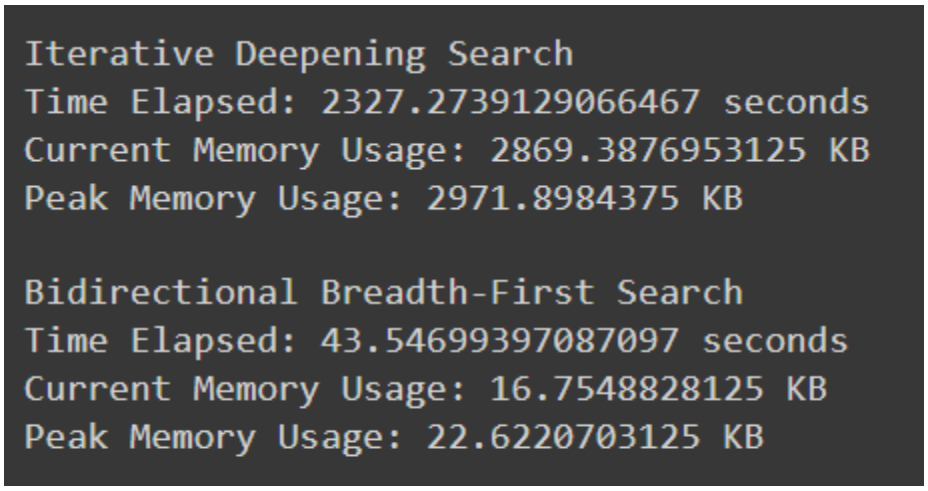
Question 3

b) For this part, we will only comment on paths given by Iterative Deepening Search and Bidirectional BFS. Below are the outputs for all public test cases:

```
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 1
Enter the end node: 2
Iterative Deepening Search Path: [1, 7, 6, 2]
Bidirectional Search Path: [1, 7, 6, 2]
A* Path: [1, 27, 9, 2]
Bidirectional Heuristic Search Path: [1, 27, 6, 2]
Bonus Problem: []
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 5
Enter the end node: 12
Iterative Deepening Search Path: [5, 3, 35, 98, 12]
Bidirectional Search Path: [5, 97, 28, 10, 12]
A* Path: [5, 97, 28, 10, 12]
Bidirectional Heuristic Search Path: [5, 97, 28, 10, 12]
Bonus Problem: []
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 12
Enter the end node: 49
Iterative Deepening Search Path: None
Bidirectional Search Path: None
A* Path: None
Bidirectional Heuristic Search Path: None
Bonus Problem: []
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 4
Enter the end node: 12
Iterative Deepening Search Path: [4, 6, 2, 9, 8, 5, 3, 35, 98, 12]
Bidirectional Search Path: [4, 6, 2, 9, 8, 5, 97, 98, 12]
A* Path: [4, 6, 27, 9, 8, 5, 97, 28, 10, 12]
Bidirectional Heuristic Search Path: [4, 34, 33, 11, 32, 31, 3, 35, 98, 12]
Bonus Problem: []
PS D:\ai>
```

We can see that the paths given by IDS and Bidirectional BFS are the same for test cases 1 and 3, and different for test cases 2 and 4.
To answer the question of whether they will be identical for all pairs of nodes or not, we need to understand why they are identical or not in the first place. The underlying reason for this is the structure of the graph.
If there is only one path between a pair of nodes, both of the algorithms will always give the same path. But if there are multiple paths, the paths can be different or the same, it depends on the algorithmic implementation.

c) Below are the results after running the algorithm for all pairs of nodes:



$$Memory\ Usage = Peak\ Memory\ Usage - Current\ Memory\ Usage$$

For IDS, the time of execution is 2327 seconds (equivalent to 38.78 minutes) and memory usage is 102.5106 KB.
For Bidirectional BFS, the time of execution is 43.5 seconds and memory usage is 5.867 KB.

We can see that IDS takes a lot of time and memory compared to Bidirectional BFS.
IDS takes a lot of time because it performs DFS for all depths from 0 to 124 (in our case) if no path exists, which explains the gigantic time of execution.

e) For this part, we will only comment on paths given by the A* search and the Bidirectional A* search. Below are the outputs for all public test cases:

```
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 1
Enter the end node: 2
Iterative Deepening Search Path: [1, 7, 6, 2]
Bidirectional Search Path: [1, 7, 6, 2]
A* Path: [1, 27, 9, 2]
Bidirectional Heuristic Search Path: [1, 27, 6, 2]
Bonus Problem: []
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 5
Enter the end node: 12
Iterative Deepening Search Path: [5, 3, 35, 98, 12]
Bidirectional Search Path: [5, 97, 28, 10, 12]
A* Path: [5, 97, 28, 10, 12]
Bidirectional Heuristic Search Path: [5, 97, 28, 10, 12]
Bonus Problem: []
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 12
Enter the end node: 49
Iterative Deepening Search Path: None
Bidirectional Search Path: None
A* Path: None
Bidirectional Heuristic Search Path: None
Bonus Problem: []
PS D:\ai> python -u "d:\ai\code_2022216.py"
Enter the start node: 4
Enter the end node: 12
Iterative Deepening Search Path: [4, 6, 2, 9, 8, 5, 3, 35, 98, 12]
Bidirectional Search Path: [4, 6, 2, 9, 8, 5, 97, 98, 12]
A* Path: [4, 6, 27, 9, 8, 5, 97, 28, 10, 12]
Bidirectional Heuristic Search Path: [4, 34, 33, 11, 32, 31, 3, 35, 98, 12]
Bonus Problem: []
PS D:\ai>
```

We can see that the paths given by A* search and Bidirectional A* search are the same for test cases 2 and 3, and different for test cases 1 and 4.
To answer the question of whether they will be identical for all pairs of nodes or not, we need to understand why they are identical or not in the first place. The underlying reason for this is the structure of the graph.
If there is only one path between a pair of nodes, both of the algorithms will always give the same path. But if there are multiple paths, the paths can be different or the same, it depends on the algorithmic implementation.

Below are the results after running the algorithm for all pairs of nodes:

```
A* Search
Time Elapsed: 95.19481134414673 seconds
Current Memory Usage: 5.154296875 KB
Peak Memory Usage: 14.451171875 KB

Bidirectional A* Search
Time Elapsed: 65.551518201828 seconds
Current Memory Usage: 6.5400390625 KB
Peak Memory Usage: 15.2822265625 KB
```

$$Memory\ Usage = Peak\ Memory\ Usage - Current\ Memory\ Usage$$

For the A* search, the time of execution is 95.2 seconds and memory usage is 9.297 KB.
For the Bidirectional A* search, the time of execution is 65.55 seconds and memory usage is 8.74 KB.

We can see that the A* search takes more time of execution than the Bidirectional A* search and both have similar memory usage. The bidirectional A* search is faster because it stops as soon as it finds an intersecting node while the A* search terminates when the most optimal path to the goal node is found.

f) Below are the combined results of all the uninformed and informed search algorithms:

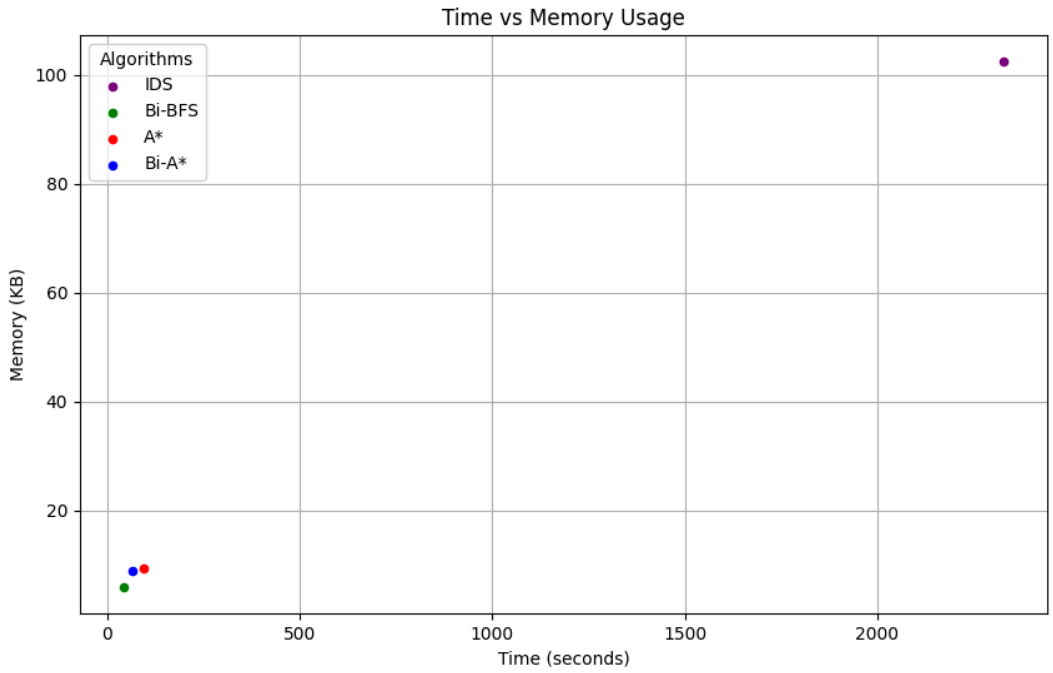
```
Iterative Deepening Search
Time Elapsed: 2327.2739129066467 seconds
Current Memory Usage: 2869.3876953125 KB
Peak Memory Usage: 2971.8984375 KB

Bidirectional Breadth-First Search
Time Elapsed: 43.54699397087097 seconds
Current Memory Usage: 16.7548828125 KB
Peak Memory Usage: 22.6220703125 KB

A* Search
Time Elapsed: 95.19481134414673 seconds
Current Memory Usage: 5.154296875 KB
Peak Memory Usage: 14.451171875 KB

Bidirectional A* Search
Time Elapsed: 65.551518201828 seconds
Current Memory Usage: 6.5400390625 KB
Peak Memory Usage: 15.2822265625 KB
```

$$Memory\ Usage = Peak\ Memory\ Usage - Current\ Memory\ Usage$$

For IDS, the time of execution is 2327 seconds (equivalent to 38.78 minutes) and memory usage is 102.5106 KB.
For Bidirectional BFS, the time of execution is 43.5 seconds and memory usage is 5.867 KB.
For the A* search, the time of execution is 95.2 seconds and memory usage is 9.297 KB.
For the Bidirectional A* search, the time of execution is 65.55 seconds and memory usage is 8.74 KB.

We can see that Bidirectional BFS is the most optimum in terms of memory and time, and IDS takes the most time and memory.

Below are the scatter plots:



The informed algorithms are more efficient and optimal overall in terms of time and memory regardless of the structure of the graph.
The informed algorithms require a heuristic function, which will not give correct results if not well defined. Also, informed algorithms are harder to implement.