AI-Assignment 1.

VISION | DATE : | PAGE NO.

Question → 1

a) A* Search

→ In this technique we find hyristic function.

- $f(n) = g(n) + h(n)$
  - $g(n)$ → Path Cost
  - $h(n)$ → estimate Cost

{N(F=g+h)}

→ Expansion Node	frontier Node	Explored Node
start from → S		
• S.	A(5=3+2), B(2=1+1), C(13=5+8)	—
	Select minimul cost.	
• B	A(5), D(9=5+4), F(6=3+3), C(13)	S,
	G3(13=13+0) → 1	
• F	F(6), D(9), C(13), G1(13=13+0) → 2	S, B,
• F	min(D(9), D(8=4+4)), C(13)	S, B, A,
• D	C(13), E(7=6+1), G2(9=9+0) → 3	S, B, A, F

VISION | DATE: PAGE NO.

→

Expansion Node	Frontier Node	Explored Node
• E	C(13)	S, B, A, F, D
	G1(8 = 8+0) → 4	
• C	G3(16 = 16+0) → 5	S, B, A, F, D, E

As above table we observe that 1, 2, 3, 4, 5 is Given below.

1. G2(13 = 13+0)
2. G1(13 = 13+0) → 9 min for G2
3. G2(9 = 8+0) → 8 min for G1
4. G1(8 = 8+0)
5. G3(16 = 16+0) → 13 min for G3

We have to select minimum Cost of each.
G1 → 08 & G2 → 9 & G3 → 13. and path respectively is { S → B → F → D → E → G1 }, { S → B → F → D → G2 }, { S → B → G3 }.

b) Uniform Cost Search.

- In this technique we consider only cost of paths.
- $f(n) = g(n)$
  - g is path cost.

VISION | DATE : PAGE NO.

→

Expansion Node	Frontier Node	Explored Node
StartNode → S,		
S.	A(3), B(1), C(5)	—
(Select minimum cost)		
B	A(3), C(5), F(3) D(5), G3(13) → 1	S.
If tie then Choose Alphabetically.		
A	F(3), C(5), D(5), G1(13) → 2.	S, B,
F	C(5), (D(5), D(4)) (min)	S, B, A,
D	C(5), E(6) G2(9) → 3	S, B, A, F.
C	E(6), G3(16) → 4	S, B, A, F, D.
E	G1(8) → 5.	S, B, A, F, D, C

- Evaluating table we find min cost of G1 & G2 & G3 is.

	Path :	Explored :
G1 → 8	(SBFDEG1)	(SBAFDC)
G2 → 9	(SBFDG2)	(SBAF)
G3 → 13	(SBG3)	(SBAFDC)

c) Iterative deepening A^*

→ It's combination of A^* and Iterative deepening DFS algorithm.

→ There is depth in 1 to 4.

→ Let threshold initially = 8

	expansion Node	~~f-score~~ threshold,	frontier. Node node(f=g+h)	explored Node
→	S [1]	8	A(5=3+2) B(2=1+1) C(13=5+8)	-
→	A [1]	8	D(11=7+4) G1(13=13+0)	S,
→	B [2]	8	D(9=5+4) F(6=3+3) G3(13=13+0)	S, A,
→	F [1]	8	D(8=4+4)	S, A, B,
→	D [1]	8	E(7=6+1) G2(9=9+0)	S, A, B, F
→	E [1]	8	G1(8=8+0)	S, A, B, F, D

→ Here In One (1st) iteration we get the Goal Inthe following Cost.

G1 → 8 & G2 → 9 and G3 → 13.

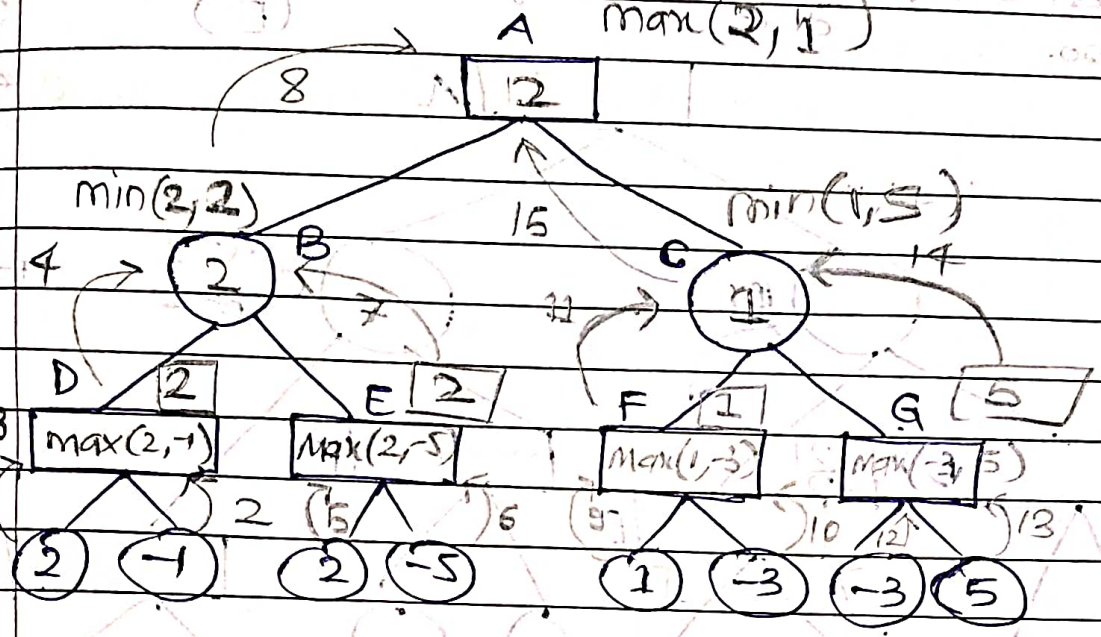
explored Node = {S, A, B, F, D, E, G1}

VISION | DATE : | PAGE NO.

question: 02

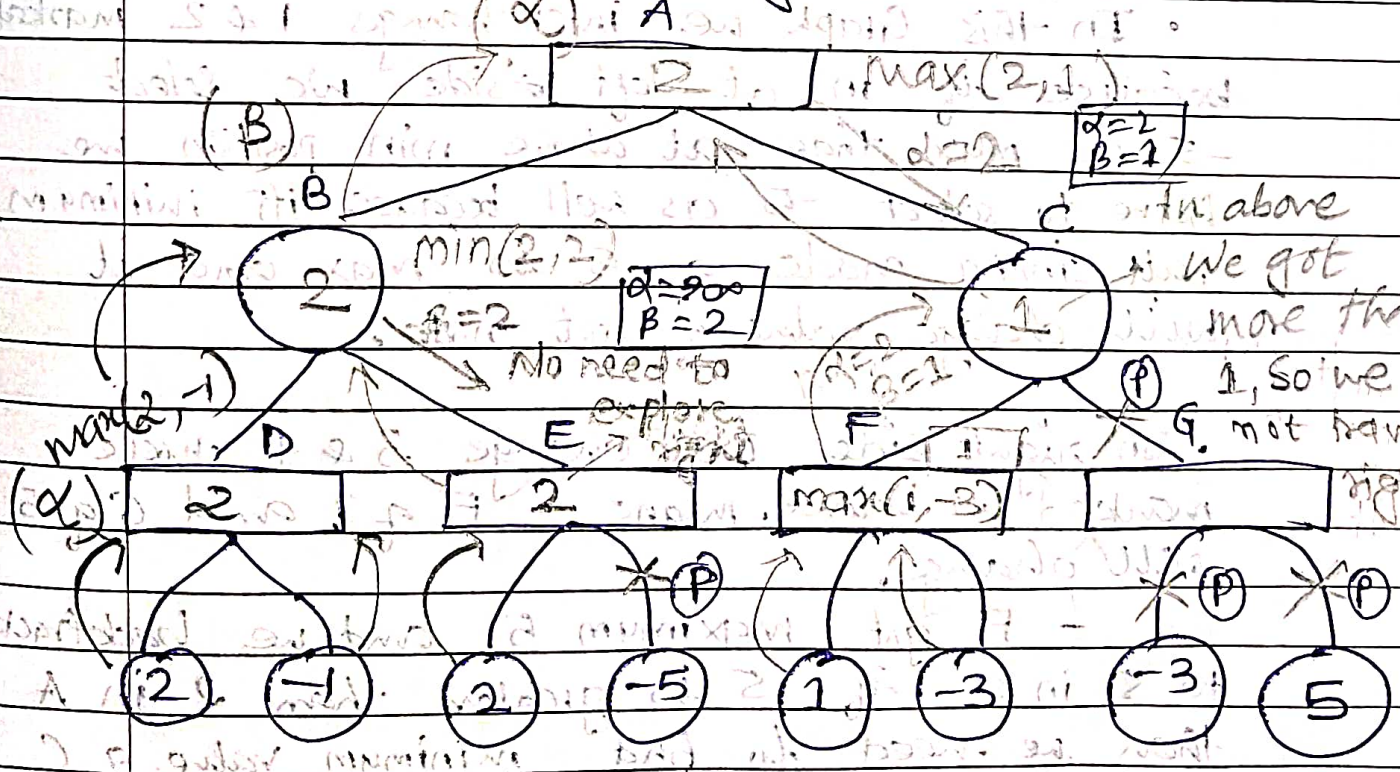
a). Part A:

→ Min-Max,



→ A Got max value is 15. {A → B → E → 2, A → B → D → 2}

→ Alpha - Beta Pruning.

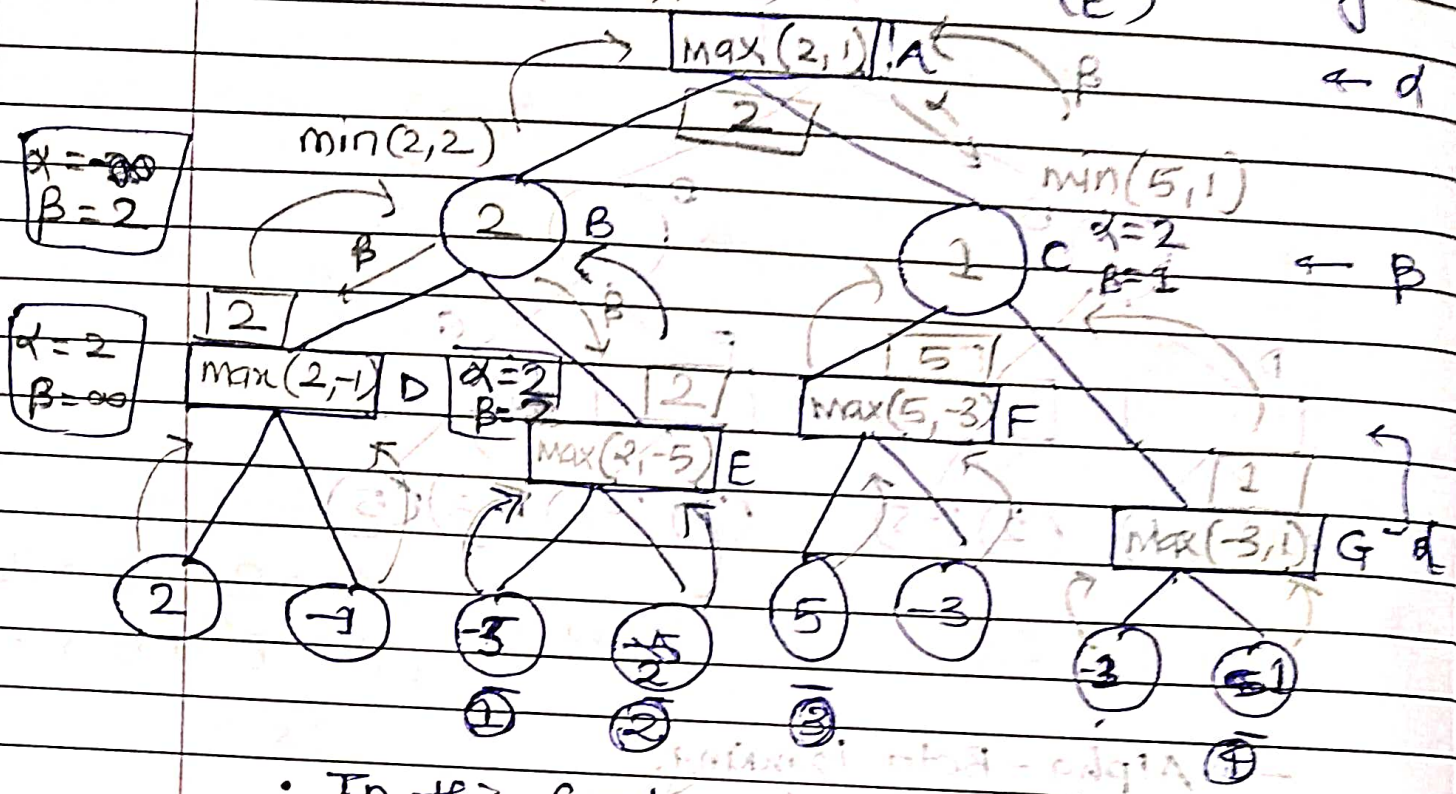


→ A Got Max value 2 from {A, B, D, 2}, {A, B, E, 2}.

Part B

b) Worst Case:- (F) (G)

→ change Node 1 and Node 5. make it worst case. and In left side α & -5 change also. (α=2, β=1) (E)



• In this Graph we interchange 1 & 2 marked because if in at left side we select -5 as max than at above min position we have to select -5 as well because its minimum but sibling node of -5 is max and it will change value of root node.

• In right side we change 3 & 4 node marked as below. means F of 1 and G of 5 will change.

- F got maximum 5 and we backtrack to 5 in C if 5 is greater than 2 in A than we need to find minimum value of C than 5.

VISION | DATE: | PAGE NO.

- So we need to traversal G node as well and we got 1 max in G and 1 minimum for C.

- In worst case we traversal all nodes in tree (or) graphs.

→ Best Case:

- In best case we no need to change because minimum (maximum) pruning is 4 times occurs.

- If we change node position to maximize pruning than we got max is 4 pruning.

- Because we need to traverse all nodes of left side in 'D' node and atleast one node of 'E'. for determining min value of 'B'. And we did it.

- Second we atleast traversal one node of right side of left 'F' because we determining 'A' with respect to right side & left side.

- right side 'C' return min. element (value) than left side of 'B'. So we don't need to traverse more node in Best Case.

c) Part C:

→ Time Complexity: $O(b^{d/2})$

↳ b → branching factor

↳ d → look ahead depth

VISIO DATE: PAGE NO.

- Here we takes b^d because at every b nodes we traversal upto d nodes. because b nodes are depth of d.

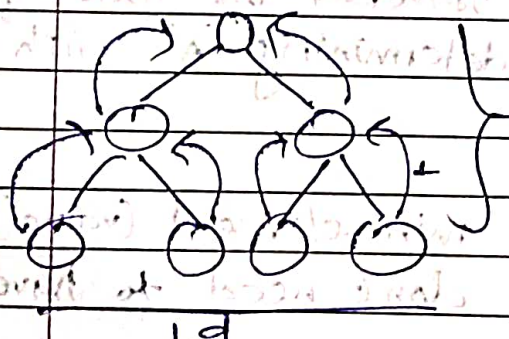
- In efficient of alpha-beta pruning reduce no. of traversal nodes upto half. so be covered $b^{d/2}$ instead of b^d.

- Because we not needed to traversal in depth for every branching factor (leaf node) as aspect of α & β value.

$$\sum_{i=0}^{d} b^i = b^0 + b^1 + \dots b^d$$ - here d is depth of each node covered.

$$= \frac{b^{d+1} - 1}{b - 1}$$ if all node we covered then we got.

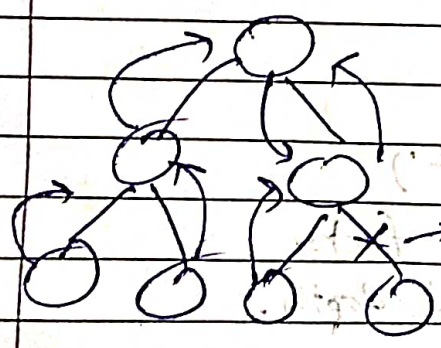
$$= O(b^d)$$



b^d

if all node traverse then we got b^d but in α-β pruning we reduce depth of branching factor b to $\sqrt{b}$.

→ Half branching → $\sqrt{b}$



→ depth skip. and reduce traversal. and also reduce complexity due to reduce branching.

PAGE NO.

Question: 03

(b) * Iterative Deepening Search:-
- It's provided shortest path, always, among all the paths.

→ Gives Output:-
Test Case 1:- [1, 7, 6, 2]
Test Case 2:- [5, 97, 98, 12]
Test Case 3:- None
Test Case 4:- [4, 6, 2, 9, 8, 5, 97, 98, 12]

* Bidirectional Search:-
- It's return shortest path but not necessary path is unique or same as IDS.
- Because It select shortest path route from both side until meet at one points.

- Give Output:-
Test Case 1:- [1, 7, 6, 2]
Test Case 2:- [5, 97, 28, 10, 12]
Test Case 3:- None
Test Case 4:- [4, 6, 2, 9, 8, 5, 97, 98, 12]

→ Not necessary both Gives path same but BDS give same ~~until~~ some path until meet points.

(c) Memory:- IDS maintain one stack but BDS maintain two queue so. space captains BDS is more than IDS.

VISION DATE: PAGE NO.

Time:- IDS cover unidirectional so they cover every branch so almost their Complexity is $O(b^{(d/2)})$.

But BDS search from both side that's why their branch factor almost half. and significately faster. time complexity is $O(b^{d})$

→ Time and Memory taken by IDS is slightly more than bidirection search path. as we observe in Image 1.

(e) A* search Algorithm:-

- Used to find shortest path from graph.
- It used heuristic function to find b' path

$f(n) = g(n) + h(n)$

actual cost. (path cost) heuristic cost (estimate)

Gives Output:-

Test Case 1:- [1, 27, 9, 2]

Test Case 2:- ~~[1, 27, 6, 2]~~ [5, 97, 28, 10, 12]

Test Case 3:- None

Test Case 4:- [4, 6, 27, 9, 8, 5, 97, 28, 10, 12]

Bidirectional heuristic search Path:-

- It follows bidirectional search but here including heuristic cost as well,
- It searching path from both side.

VISION | DATE : PAGE NO.

Output:-

Test Case 1:- [1, 27, 6, 2]

Test Case 2: [5, 97, 28, 10, 12]

Test Case 3: None

Test Case 4: [4, 34, 33, 11, 32, 31, 3, 5, 97, 28, 10, 12]

→ Time analyse :-

As per given test case we observe that most bidirectional takes less time but some time take much time than A* search.

That concept also used in memory usage by algorithm, we observe all case in Image 2

(f) In uniform and inform search we observed :-

→ Metrics → Time, space, path length.

→ efficiency (Time VS Memory)

- Here we observe informed and uninform both takes almost same memory but some cases uninformed like ids take more space.
- But in time informed both algorithm takes less time than IDS (uninformed) but BDS (uninformed) are faster than all algorithms.

→ Optimal (Path length VS Path Time)

- Informed search are gives optimal path and shortest path in average time.

- Uninformed algorithm does not provide always optimal path and takes longer time due to node explore.

VISION DATE : PAGE NO.

→ Informed Search Algorithms:-

• Benefits

- No need for heuristic information, making versatile in different domain implementation easily.

- BDS perform better than IDS and works efficiently.

• Drawbacks:-

- IDS takes much time on when path doesn't exist.

- Also IDS explore unnecessary node some times and return first path not optimal.

- Memory space depends on large search space.

→ Uninformed Search Algorithms:-

• Benefits

- Work intelligently.

- faster searching and provided optimal path.

- It balanced b space and time that's why work efficiently.

• Drawback

- Depends on quality (strongest) heuristic value. Poor heuristic may doesn't provide optimal.

- May overhead in simple problem due to heuristic.

```
Source: 1, Destination: 2
Iterative Deepening Search Path: [1, 7, 6, 2], Time: 1.9958019256591797 ms , Memory: 0.0 KB
Bidirectional Search Path: [1, 7, 6, 2], Time: 0.0 ms , Memory: 0.0 KB
--------------------------------------------------
Source: 5, Destination: 12
Iterative Deepening Search Path: [5, 97, 98, 12], Time: 5.002498626708984 ms , Memory: 0.0 KB
Bidirectional Search Path: [5, 97, 28, 10, 12], Time: 0.9839534759521484 ms , Memory: 0.0 KB
--------------------------------------------------
Source: 12, Destination: 49
Iterative Deepening Search Path: None, Time: 15001.412630081177 ms , Memory: 0.0 KB
Bidirectional Search Path: None, Time: 0.0 ms , Memory: 0.0 KB
--------------------------------------------------
Source: 4, Destination: 12
Iterative Deepening Search Path: [4, 6, 2, 9, 8, 5, 97, 98, 12], Time: 26.00264549255371 ms , Memory: 0.0 KB
Bidirectional Search Path: [4, 6, 2, 9, 8, 5, 97, 98, 12], Time: 1.001119613647461 ms , Memory: 0.0 KB
--------------------------------------------------
Source: 7, Destination: 12
Iterative Deepening Search Path: [7, 1, 27, 9, 8, 5, 97, 98, 12], Time: 38.001298904418945 ms , Memory: 0.0 KB
Bidirectional Search Path: [12, 10, 28, 60, 83, 59, 58, 1, 7], Time: 1.0008811950683594 ms , Memory: 0.0 KB
--------------------------------------------------
```

Image 1: Uninformed Search

```
Source: 1, Destination: 2
Astar Search Path: [1, 27, 9, 2], Time: 1.9989013671875 ms, Memory: 0.0 KB
Bidirectional Heuristic Search Path: [1, 27, 6, 2], Time: 1.0004043579101562 ms, Memory: 0.0 KB
--------------------------------------------------
Source: 5, Destination: 12
Astar Search Path: [5, 97, 28, 10, 12], Time: 2.999544143676758 ms, Memory: 0.0 KB
Bidirectional Heuristic Search Path: [5, 97, 28, 10, 12], Time: 1.0001659393310547 ms, Memory: 0.0 KB
--------------------------------------------------
Source: 12, Destination: 49
Astar Search Path: None, Time: 7.999181747436523 ms, Memory: 60.0 KB
Bidirectional Heuristic Search Path: None, Time: 1.001119613647461 ms, Memory: 0.0 KB
--------------------------------------------------
Source: 4, Destination: 12
Astar Search Path: [4, 6, 27, 9, 8, 5, 97, 28, 10, 12], Time: 1.9984245300292969 ms, Memory: 0.0 KB
Bidirectional Heuristic Search Path: [4, 34, 33, 11, 32, 31, 3, 5, 97, 28, 10, 12], Time: 2.0003318786621094 ms, Memory: 0.0 KB
--------------------------------------------------
Source: 7, Destination: 12
Astar Search Path: [7, 1, 58, 59, 83, 60, 28, 10, 12], Time: 3.9992332458496094 ms, Memory: 0.0 KB
Bidirectional Heuristic Search Path: [12, 10, 28, 60, 83, 59, 58, 1, 7], Time: 4.000186920166016 ms, Memory: 0.0 KB
--------------------------------------------------
```

Image 2: Informed Search