AI ASSIGNMENT — Search Algorithms
Deadline : 11:59 PM, 17/9/2024

Total Marks : 100 Marks **Weightage: 10%**

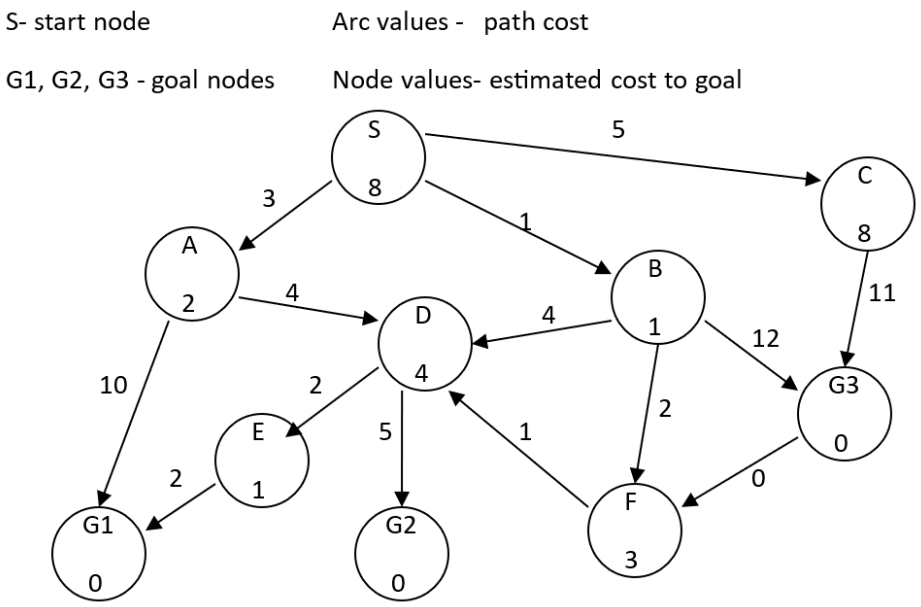
Instructions:

1. Assignments are to be attempted individually.
2. Submit the assignment as a single zipped folder (**Search_⟨RollNumber⟩.zip**) containing a pdf file for all the theory questions and a python file (code_⟨RollNumber⟩.py) for programming questions.
3. Please read the instructions given in the questions carefully. In case of any ambiguity, post your queries on Google Classroom at least a week before the deadline. **No TA will be responsible for responding to the queries after this.**
4. A part of the assignment evaluation involves automatic testing of your submitted code on private test cases. **Please make sure that you do not change the structure of the methods provided in the boilerplate code.**
5. All the TAs will strictly follow the rubric provided below. **No requests will be entertained related to scoring strategy.**
6. **The use of generative tools (such as ChatGPT, Gemini, etc.) is strictly prohibited.** Failure to comply may result in severe consequences related to plagiarism.
7. **Extension and Penalty clause:**
   - Even a 1 minute late submission on google classroom will be considered as late. Please turn-in your submissions atleast 5 minutes before the deadline.
   - Not explaining the answers properly will lead to zero marks.

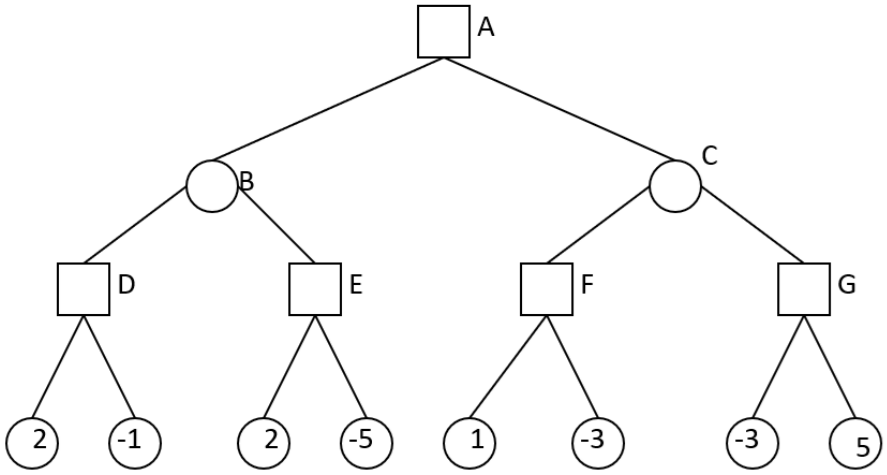
1. Consider the search space below where an edge from node x to node y means that y can be generated from x by an operator. The edges are labeled with the actual path cost and the estimated costs to a goal (h-values) are provided inside the nodes.
   You are required to find the shortest path from S to goal G1 using the following algorithms:
   (a) A* (5)
   (b) Uniform cost search (5)
   (c) Iterative deepening A* (5)

   List the order of nodes expanded (not generated) by these algorithms together with the f-values and total path costs. Show detailed steps of each algorithm in a table with node

expansions, frontier nodes, explored nodes (where applicable) for each algorithm.
Note: Explore nodes in counter clock-wise direction (left to right) when no other criterion is specified. Also, if two nodes are at the same cost, pick nodes alphabetically for tie-breaking.



2. Consider the following game tree. Note that each square signifies a Max node and each circle is a Min node.



(a) Part A

- Use min-max algorithm to determine best play for both players, i.e. best moves at all levels of the tree for both players. Use arrows to represent the best moves. Show all legitimate moves. (2)

- Alpha-beta pruning is a directional search algorithm, i.e. it explores children from left to right. Apply alpha-beta pruning to the given game tree. Cross-out the branches where pruning can be done and mark them with alpha or beta depending on its type. (3)

(b) Part B

- Best case: Rearrange the leaf nodes (and internal nodes if necessary) of the given tree so that the maximum pruning is achieved by alpha-beta pruning that explores branches from left to right. Justify your answer. (4)
- Worst case: Rearrange the leaf nodes again to make the worst case for alpha-beta pruning. Justify your answer. (4)

(c) Part C
Based on your best case, briefly explain why the best-case complexity is $O(b^{d/2})$, where b and d are the branching factor and look-ahead depth, respectively. (2)

3. Consider a graph $\mathcal{G} = (V, E)$ representing our institute (Fig 1), where V is the set of nodes and E is the set of edges. Each node $u \in V$ represents a unique geographical location and each edge $e_{uv} \in E$ represents a road connecting the locations u and v. The graph $\mathcal{G}$ is given in the form of an $n \times n$ adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{uv} > 0$ indicates the travel cost between locations u and v along edge e_{uv}, whereas $A_{uv} = 0$ means that locations u and v are not directly connected, implying an infinite cost for direct travel between them. The locations are named as $\{0, \cdots, n-1\}$ and all the nodes in the graph are attributed with (x, y) denoting the latitude and longitude of the corresponding location. The dataset and the boilerplate code can be downloaded from this link.

The primary objective of this programming question is to find a path (sequence of nodes traversed including u and v) from a source node u to destination v for a given graph $\mathcal{G} = (V, E)$ using various algorithms and compare them. The output of a few 'public' test cases is provided in the shared code. The final scoring will be done automatically using 'multiple private' test cases. For this purpose, you must not change the method definitions provided in the boilerplate code; otherwise, your submission will not be graded.

(a) Implement the following uninformed search algorithms: $(10 + 10 = 20)$

- Iterative Deepening Search
- Bidirectional Breadth-First Search

(b) For each public test case, analyze whether the path obtained to travel from source u to destination v using both algorithms is the same. If you find that the path obtained is identical (or different), comment whether it will always be identical (or different) for any pair of nodes in the given graph $\mathcal{G}$. $(2.5 + 2.5 = 5)$

(c) Obtain the path between all pairs of nodes using both algorithms. Compare the memory usage and time of execution for both algorithms. $(2.5 + 2.5 = 5)$

(d) Let $dist(u, v)$ be a function that calculates the Euclidean distance between two nodes u and v using the node attributes (x, y). Use the heuristic function given

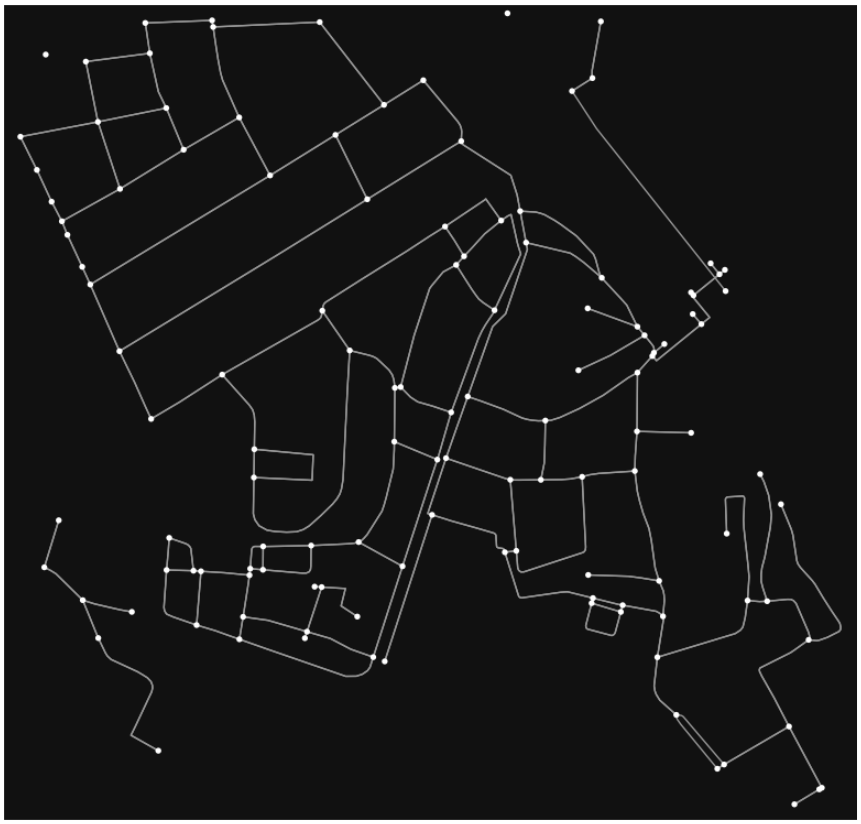


Figure 1: A graph representing IIIT Delhi

below to implement the following informed search algorithms. $(10 + 10 = 20)$

$$h(w) = dist(u, w) + dist(w, v)$$

- A* Search
- Bidirectional A* Search

(e) Repeat the exercises (b) and (c) using both these informed search algorithms. $(2.5 + 2.5 + 2.5 + 2.5 = 10)$

(f) Analyze the results obtained using all the uninformed and informed search algorithms. Using scatter plots, compare and contrast the efficiency (in terms of space and time) and optimality (in terms of cost of traveling) of all the algorithms. Explain how the metric to generate the scatter plots was obtained. Also, comment on the benefits and drawbacks of using informed search algorithms over uninformed ones, supported via the empirical analysis. $(5 + 5 = 10)$

(g) **(BONUS)** There might be geographical locations that are only connected via one road. Removal of such a road will disconnect the geographical locations. For example, India and Sri Lanka were only connected via The Ram Sethu bridge but due to its submersion the cities in India are no more connected to that of Sri Lanka

via roads. In order to reduce the vulnerability of such road networks, the minister of road transport and highways wants to construct new roads within our country. For this purpose, the ministry has asked you to develop an algorithm that can be used to identify all vulnerable roads. Specifically, for the given graph G, identify all the edges whose removal would increase the number of disconnected components. (10)

SOLUTIONS

1. The tables may not show the frontier in the right order, but the expansions should be done correctly.
   We want the order of nodes expanded and the basic steps like frontier/explored nodes if maintained by the algorithm. The data structure for maintaining shortest paths can be skipped, but they need to still keep track of those. The algorithms are supposed to be the versions from the text as it was mentioned in the comments where there were doubts about different variants.

   - Best-first Search (A*)
     S(8) → B(2) → A(5) → F(6) → D(8) → E(7)
     G1 (8) identified as goal.

Node expanded	Frontier	Explored
S 8	A 5, B 2, C 13	S
B 2	A 5, C 13, D 9, F 6, G3 13	S, B
A 5	G1 13, C 13, D 8, F 6	S, B, A
F 6	D 8, C 13, G1 13, G3 13	S, B, A, F
D 8	E 7, G2 9, C 13, G1 13, G3 13	S, B, A, F, D
E 7	G1 8, G2 10, C 13, G1 13, G3 13	S, B, A, F, D, E
G1 8		

     A*
     The idea is that while breadth-first search spreads out in waves of uniform depth, uniform-cost search spreads out in waves of uniform path-cost. Marks are distributed uniformly according to the steps shown in the frontier and the correct node expansions. If node expansion or criteria used is incorrect then award marks only for steps up to that point. Please see that they do not necessarily have to show how shortest paths are being maintained.

   - Uniform cost search
     S(8) → B(1) → A(3) → F(3) → D(4) → C(5) → E(7)
     G1(8) identified as goal.

Node expanded	Frontier	Explored (Not necessary in UCS)
S	A 3, B 1, C 5	S
B 1	A 3, C 5, D 5, F 3, G3 13	S, B
A 3	C 5, D 5, F 3, G3 13, G1 13	S, B, A
F 3	C 5, D 4, G3 13, G1 13	S, B, A, F
D 4	C 5, G3 13, G1 13, E 7, G2 10	S, B, A, F, D
C 5	G3 13, G1 13, E 7, G2 10, G3 16	S, B, A, F, D, C
E 7	G3 13, G1 8, G2 10, G3 16	S, B, A, F, D, C, E
G1 8		

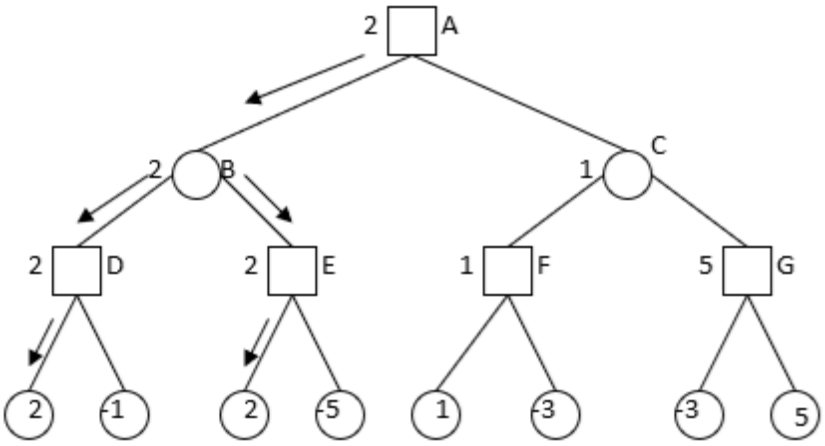
Uniform cost search
Marks are distributed uniformly according to the steps shown in the frontier and the correct node expansions. If node expansion or criteria used is incorrect then award marks only for steps up to that point. Please see that they do not necessarily have to show how shortest paths are being maintained.

- Iterative Deepening A*
  1st iteration:
  Cut-off threshold alpha = 8,
  S(8) $\rightarrow$ A(5) $\rightarrow$ B(2) $\rightarrow$ F(6) $\rightarrow$ D(8) $\rightarrow$ E(7)
  G1(8) identified as goal node.
  The rest of the nodes are pruned.

  IDA* works with f-values similar to A*, but it does not maintain a frontier or explored nodes list. It is simply ID, but with f values instead of depth values. It always continues to proceed deeper along a path as long as the threshold isn't exceeded or the goal is not found. Every iteration starts from scratch with an updated threshold value, but this is not required since the algorithm manages to find the goal in the first run.

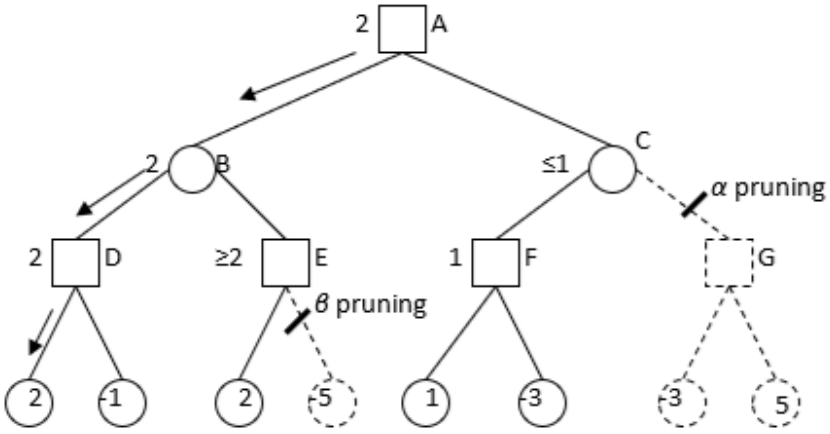
2. A

- Using Min-Max algorithm the solution is as given below.
  The arrows show both solutions.
  Give 1 mark if all node values have been correctly specified. And 0.5 marks each for the two equivalent paths shown by arrows. There can be two paths since B would have identical options below it due to the nature of min-max.



Min-max

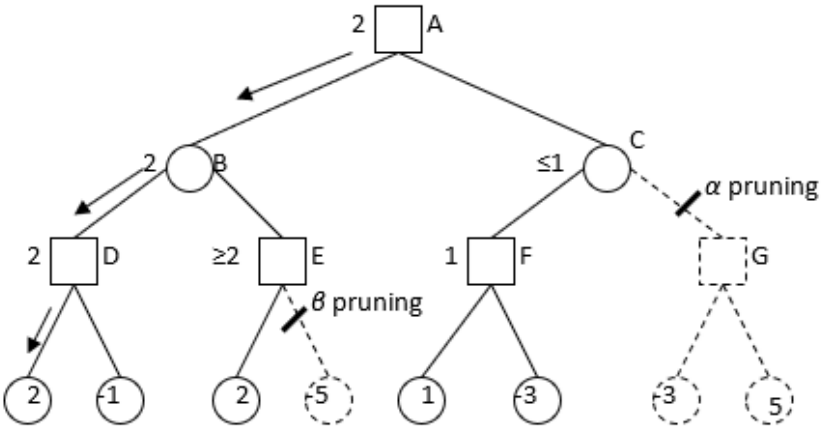
- Using Alpha-Beta Pruning algorithm the solution is given below.
  The pruning at the -5 node is beta pruning and the one at the G node is alpha pruning.
  1 marks each for crossing out the correct branches to denote pruning and 0.5 marks each for labeling it correctly as alpha or beta.



Alpha-beta pruning

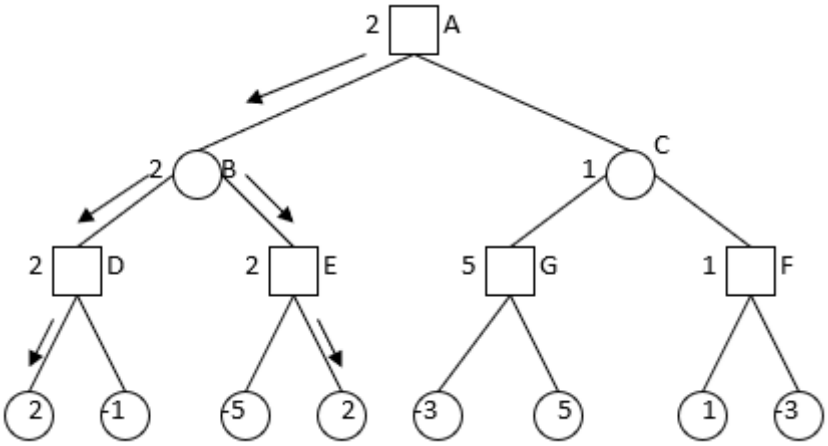
B

- Best-case:
  The leaf nodes and internal nodes can be arranged so that the maximum pruning is achieved by alpha-beta pruning. This arrangement happens to be the same as given in the original problem!
  3 marks for the correct arrangement. If arrangement is mixing nodes across levels, then 0 marks. 1 mark for the justification that why this is the best case. Justification could be simply suggesting why lesser pruning is happening in other arrangements. Equivalent solutions, if any, should be considered. Only one solution is needed!

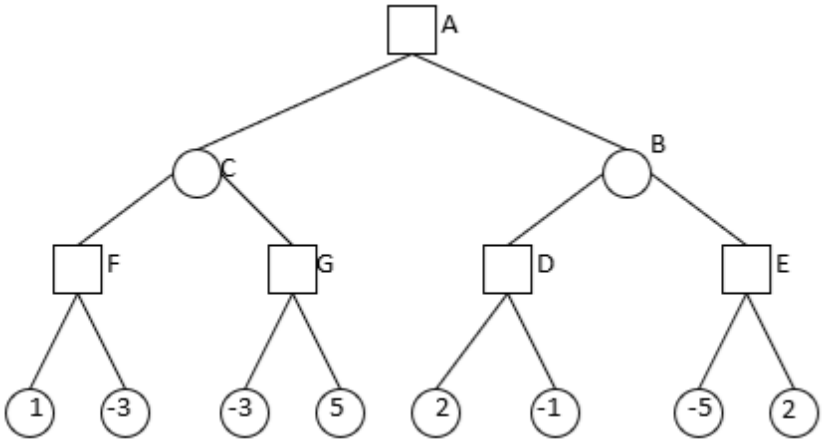


Min-max: Best case

- Worst-case: The leaf nodes and internal nodes can be arranged so that the no pruning is achieved by alpha-beta pruning. Two possible arrangements are shown below. Only one solution is needed!
  3 marks for the correct arrangement. If arrangement is mixing nodes across levels, then 0 marks. 1 mark for the justification that why this is the worst case. Justification could be simply suggesting that pruning is happening in other arrangements. Two solutions are listed here and they are both correct. Only one worst-case example is enough.



Solution 1



Solution 2

- Complexity:
  The complexity of the method depends on the number of nodes to be examined for taking the decision. In the worst case all the nodes have to be examined and hence the time complexity is O(b^d). In the best case with alpha-beta pruning, we need to examine only O($b^{d/2}$) nodes to decide the best move. It can be seen from the best case that the effective branching factor has been reduced to $b^{1/2}$ instead b.
  If the proof is incomplete, however, it includes some argument regarding how the branching factor is reduced due to pruning then you can give 1 mark. If the proof is similar to the one given here and justifies the branching factor then 2 marks.

3. B

- Award 2.5 marks if the analysis is done; the remaining 2.5 marks depend on the reasoning provided.

  For Iterative Deepening Search (IDS) and Bidirectional Breadth-First Search (BFS), if there exists a unique shortest path between the start and goal nodes, both algorithms will reliably return the same path. However, in cases where multiple shortest paths are possible, the paths returned by IDS and BFS may be either identical or distinct, depending on the order of node expansion and selection criteria.

  Any reasoning that leads to identical or different paths is acceptable, provided it is justified correctly.

  Reasoning for identical paths: If both IDS and Bidirectional BFS expand nodes in a consistent sequence (for instance, by exploring neighboring nodes in the same specified order), the paths produced may match exactly, as node expansions align.

  Reasoning for different paths: In graphs with multiple shortest paths, both algorithm may still guarantee a shortest path, though they may return differing paths. This variation arises due to the potentially different sequences in which nodes are explored, influenced by the algorithms' specific expansion and traversal choices.

C

- Check explanation or time / space complexity. 2.5 for memory usage and 2.5 for time usage.
  Note: Award full marks if either the explanation or the time/space complexity is correct. Do not deduct marks if either is missing.

  Time Usage: Iterative Deepening Search (IDS) is more time-intensive than Bidirectional Search when locating a path between two nodes. In contrast, Bidirectional Search is typically faster, as it can reduce the search space significantly by simultaneously expanding from both the start and goal nodes, aiming for an intersection between the two searches.
  IDS Time Complexity: $O(b^d)$.
  Bidirectional Time Complexity: $O(b^{d/2})$, where d is the depth of the solution, or in terms of nodes and edges, $O(n + m)$, where n and m denote the number of nodes and edges, respectively.

  Memory Usage: When it comes to memory efficiency, IDS has a significant advantage, as it only needs to store a single path from the root node to the frontier. In contrast, Bidirectional Search requires additional memory because it maintains two queues (or frontiers) that expand from both the start and goal nodes.
  IDS Space Complexity: $O(bd)$ or $O(d)$
  Bidirectional Space Complexity: $O(b^{d/2})$. Alternatively, in terms of nodes, the space complexity may be approximated as $O(n)$.

E

- Award 2.5 marks if the analysis is done; the remaining 2.5 marks depend on the reasoning provided.

  For A* and Bidirectional A*, the paths produced by the two algorithms for the same node pair may differ due to their reliance on heuristic guidance. As given heuristic is inadmissible (overestimates the cost to reach the goal), they may find paths that are feasible but not necessarily optimal, and the paths obtained may vary between the two methods.
- Check explanation or time / space complexity. 2.5 for memory usage and 2.5 for time usage.
  Note: Award full marks if either the explanation or the time/space complexity is correct. Do not deduct marks if either is missing.

  Time Usage: Bidirectional A* typically outperforms A* in speed by reducing the search space through simultaneous exploration from both the start and goal nodes.

This dual search strategy requires expanding fewer nodes, enhancing overall efficiency.

Space Usage: A* consumes more memory as it retains all generated nodes in its open set, leading to higher memory overhead. In contrast, Bidirectional A* is more memory-efficient since each search only covers about half of the graph, resulting in lower memory requirements.

F

- Plot (2.5 marks): To analyze the performance of time, space, and cost, 2-3 scatter plots are expected.
  2.5 marks for explanation of how the metric to generate the scatter plots was obtained.

- Benefits (2.5 marks):
  Informed search algorithms, offer advantages over uninformed searches by using heuristic functions to more effectively guide the search toward the goal. This heuristic guidance generally enables faster path finding by prioritizing more promising nodes, reducing the overall number of expansions required.

  Drawback (2.5 marks):
  A notable drawback of informed search is that it may incur high memory usage. Algorithms like A*, which maintain a priority-based open set (frontier), can result in increased memory overhead, particularly in expansive search spaces. This is due to the need to store and prioritize nodes based on the estimated cost, which may not scale efficiently in certain cases.
  (Other Benefits and Drawbacks are acceptable as well)