Problem 1 (6 points)

Read the following article on *Nature* (click "Access through your institution", type and select "Columbia University", and then log in using your UNI):

- ChatGPT has entered the classroom: how LLMs could transform education

Respond to the following questions with about three or four sentences each. There are no correct or incorrect answers, but try to think about what you've read and use any other knowledge or experiences you have had to formulate thoughtful responses.

1. Do you think that there are legitimate use cases for ChatGPT in every university class, either now or in the near future? Or are there certain subjects that should not incorporate any AI tools whatsover? Give a couple examples to support your response.
2. Suppose you used ChatGPT for homework in this class. What negative effects, if any, do you think it may have on your learning? You may refer to examples raised in the article or otherwise. Please try to reflect on your **individual** learning style and tendencies.
3. Suppose you used ChatGPT as a general learning supplement in this class. What positive effects, if any, do you think it may have on your learning? You may refer to examples raised in the article or otherwise. Please try to reflect on your **individual** learning style and tendencies.

1. Legitimate use cases for ChatGPT in every university class: providing personalized learning experience. For example, when there is a concept that I don't understand during class, I can immediately consult ChatGPT about whatever question I would like to ask. I believe this is applicable to any subject. However, during some classes that rely on empirical results, such as physics and chemistry labs, those results cannot be generated by ChatGPT.

2. Negative effects: I might just copy down what ChatGPT told me without thinking through the problem. Moreover, over reliance on ChatGPT might hinder the cultivation of critical thinking and problem-solving skills that are essential for academic and professional success. The thinking process is especially important when we encounter challenging problems.

3. Sometimes the slides that professors use contain only formulas and not many explanations. It's helpful to use ChatGPT to explain those slides for me, for example, what the notations mean. Moreover, although ChatGPT is not always correct, it can provide immediate hints for the problems that confuse me so that I don't have to wait till office hours.

Problem 2 (12 points)

AlphaGeometry was a recently announced AI system that can solve geometry problems at the "gold-medal" level of the International Mathematical Olympiad. (You should read the linked blog post to complete this problem, but you do not have to read the scientific article on *Nature*.)

1. (4 pts) Give a complete PEAS description of the task environment to which AlphaGeometry is a solution.
2. (6 pts) Classify this task environment according to the six properties discussed in class, and include a one- or two-sentence justification for each. For some of these properties, your reasoning may determine the correctness of your choice.
3. (2 pts) Would AlphaGeometry best be classified as a simple reflex agent, model-based reflex agent, goal-based agent, learning agent, or some combination of these? Briefly explain your answer.

1. PEAS: Performance measure, environment, actuators, sensors

- P:
  - In the benchmarking test: number of Olympiad geometry problems solved (out of 30) under competition time limits.
  - Complexity/Difficulty of the problems.
  - Correctness of AlphaGeometry's solution.
  - Quality of the output: verifiable and clean.
- E:
  - Olympiad geometry problems.
  - synthetic training data.
- A:
  - symbolic deduction engine, are based on formal logic and use clear rules to arrive at conclusions.
  - neural language model, make good suggestions for new construct.
  - parallelized computing.
- S:
  - neural language model, identifying general patterns and relationships in data.

2. Task Environment Properties

- Fully observable vs partially observable vs unobservable
  Can agent sense all relevant information? Is internal state (memory) required?
  - **Fully Observable**: The system receives the entire geometry problem and theorem premises.
- Single-agent vs multi-agent
  Does maximization of performance measure depend on other agents' behaviors?
  - **Single-Agent**: AlphaGeometry solves geometry problems independently. Maximization of performance measure only depends on AlphaGeometry's ability to solve the problems.
- Deterministic vs stochastic
  Can we completely predict the next state of the environment?
  - **Deterministic**: the environment is deterministic and no contingencies. After adding new geometric constructs, the result is certain and can be predicted. Similarly, after using a particular theorem on the problem, the result is certain.
- Episodic vs sequential
  Do current decisions depend on past ones? Do current decisions affect future ones?
  - **Sequential**: current decisions depend on past ones and current decisions affect future ones. All previous constructions and deductions are used to make decisions. And adding one potentially useful construct opens new paths of deduction for the symbolic engine, which affects what the next decision is.
- Static vs dynamic
  Does the environment change while the agent is thinking?
  - **Static**: The environment does not change while the agent is thinking. The agent does not need to maintain model the world while it works towards a solution.
- Discrete vs continuous
  Is number of states, actions, percepts, time, etc. finite?
  - **Discrete**:

each individual problem has a finite number of possible states
solving a particular problem requires finite actions.
the percepts are finite as there is finite amount of information input in the problem statement.
solving a particular problem requires finite amount of time.

3. Combination of a learning agent and a goal-based agent

Goal-based agents try to achieve particular states
Problems often solved using search and planning

AlphaGeometry try to solve (a correct solution or proof) the geometry problems by deducing (searching) statements that can be used and planing for new constructs.

Learning agent: a tool in AI that is capable of learning from its experiences.
AlphaGeometry learns from both the training dataset and the experiences it get from solving the problems, and improve its performance over time.

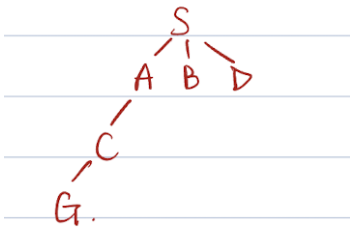
Problem 3 (16 points)
In the state space graph below, S is the start state and G is the goal state. Costs are shown along edges and heuristic values are shown adjacent to each node. All edges are undirected (or bidi- rectional). Assume that node expansion returns nodes by alphabetical order of the corresponding states, and that search algorithms expand states in alphabetical order when ties are present.

Here, the early goal test refers to checking that a state is the goal right before insertion into the frontier. The late goal test refers to checking it right after popping from the frontier. Usage of a reached table allows for multiple path pruning.

1. (3 pts) Suppose we run DFS. Assuming that it finishes, list the sequence of states that are expanded (not including the goal), as well as the state sequence solution, for each of the following scenarios:

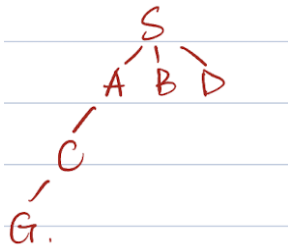
a) early goal test and reached table,

Expanded: S, A, C
Solution: S, A, C, G



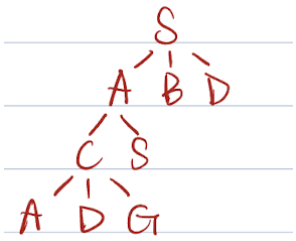
b) late goal test and reached table,

Expanded: S, A, C
Solution: S, A, C, G



c) late goal test and no reached table. Otherwise, indicate that DFS does not finish.

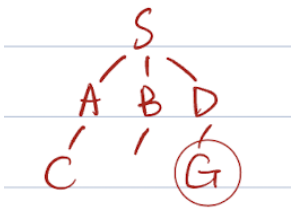
Expanded: S, A, C, A, C, A, C, …
DFS does not finish



2. (3 pts) Repeat part 1 above for BFS.

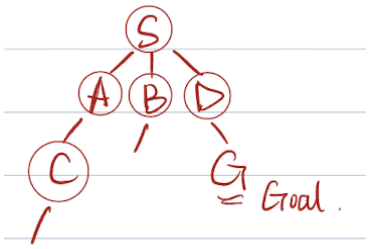
a) early goal test and reached table,

Expanded: S, A, B, D
Solution: S, D, G



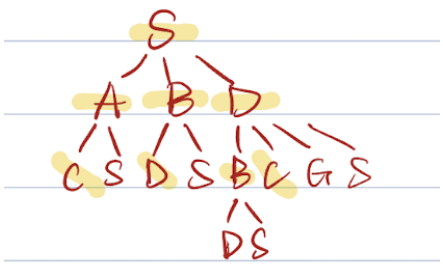
b) late goal test and reached table,

Expanded: S, A, B, D, C
Solution: S, D, G



c) late goal test and no reached table. Otherwise, indicate that DFS does not finish.

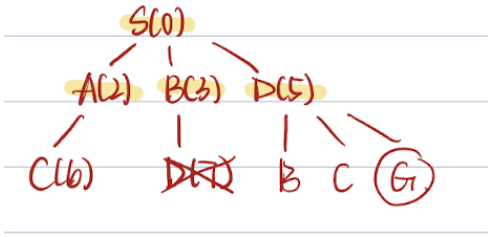
Expanded: S, A, B, D, B, C, C, D
Solution: S, D, G



3. (2 pts) Suppose we run UCS with a reached table. List the sequence of expanded states as well as the solution for each of the following scenarios:

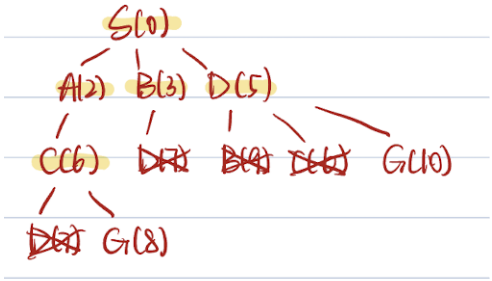
a) early goal test,

Expanded: S, A, B, D
Solution: S, D, G (10)



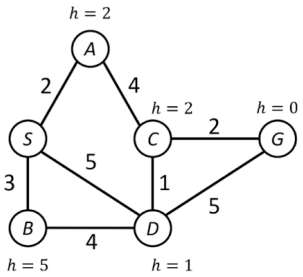
b) late goal test.

Expanded: S, A, B, D, C
Solution: S, A, C, G (8)

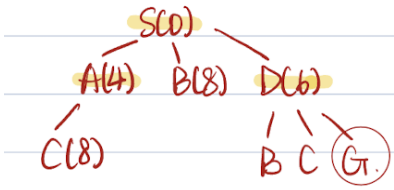


4. (2 pts) Repeat part 3 above for A* search.

a) early goal test,

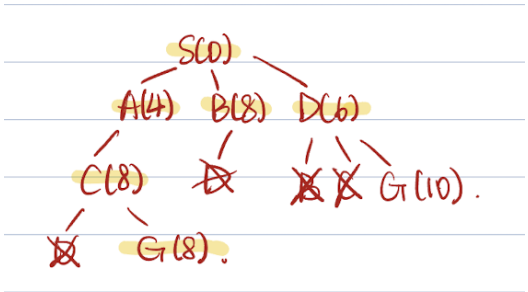


Expanded: S, A, D
Solution: S, D, G; Cost: 10



b) late goal test.

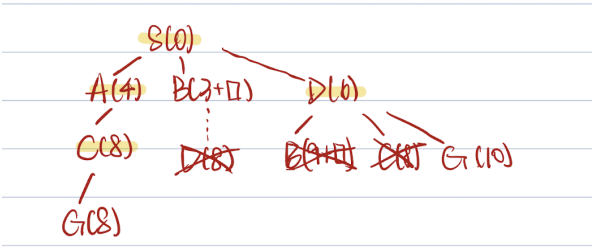
Expanded: S, A, D, B, C
Solution: S, A, C, G; Cost: 8



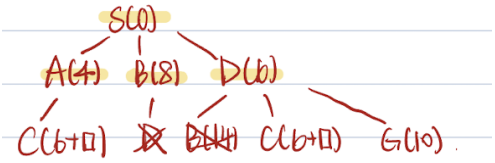
5. (4 pts) Suppose we change the heuristic function. Find the range of nonnegative heuristic values for h(A) that would cause A* (utilizing a late goal test and reached table) to return an suboptimal solution, or indicate if A* would behave optimally regardless of h(A). Repeat for h(B), h(C), and h(D).

1. A* behave optimally regardless of h(A)
   No matter what h(A) chooses
   The solution is either S->A->C->G (8) or S->D->C->G (8)
   The costs are both 8 and are both optimal.

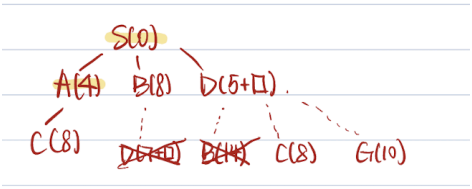
2. A* behave optimally regardless of h(B)
   No matter what h(B) chooses, the path from S to D will be S->D and never be S->B>D
   The solution will always be the optimal solution S->A->C->G (8)



3. Suboptimal solution S->D->G (10) if h(C)>4
   This is because f(C)>f(G)=10, we will choose the goal G before choosing C
   Otherwise the solution is always S->A->C->G (8)



4. A* behave optimally regardless of h(D)
   The solution will always be the optimal solution S->A->C->G (8)



6. (2 pts) A heuristic value has not been assigned to S. Give the upper bound on the value of h(S) so that a) h is admissible (but possibly inconsistent), and b) h is consistent (and admissible).

a) Upper bound h(S) = 8
b) Upper bound h(S) = 4

Problem 4 (16 points) We are trying to schedule five activities (A, B, C, D, E) into four timeslots (1, 2, 3, 4). Each activity may be assigned to any of the timeslots, and a timeslot may contain no or multiple activities, but the following constraints must be satisfied:

- $A > D$
- $B \geq A$
- $C > E$
- $D > E$
- $A \neq C$
- $B \neq C$
- $C \neq D$
- $C \neq D + 1$

1. (5 pts) We set up this problem as a constraint satisfaction problem and run arc consistency. Find the domain of A if it is the first variable to be made arc-consistent with the full domains of all other variables. Repeat for each of the other variables, indicating their domains in the scenario that each is the first variable to be made arc-consistent.

A is the first variable:
Make A consistent with D: A = {2,3,4}
A is consistent with C
A is consistent with B
A = {2,3,4}

B is the first variable:
B is consistent with A
B is consistent with C
B = {1,2,3,4}

C is the first variable:
Make C consistent with E: C = {2,3,4}
C is consistent with A
C is consistent with D
C is consistent with B
C = {2,3,4}

D is the first variable:
Make D consistent with A: D = {1,2,3}
Make D consistent with E: D = {2,3}
D is consistent with C
D = {2,3}

E is the first variable:
Make E consistent with C: E = {1,2,3}
E is consistent with D
E = {1,2,3}

2. (5 pts) Now continue from the domains that you found by repeatedly making all variables arc-consistent with all other variables (i.e., run the AC-3 algorithm). List the resultant domains when the process finishes.

- $A > D$
- $B \geq A$
- $C > E$
- $D > E$
- $A \neq C$
- $B \neq C$
- $C \neq D$
- $C \neq D + 1$

$A > D$
A = {2,3,4}
B = {1,2,3,4}
C = {1,2,3,4}
D = {1,2,3}
E = {1,2,3,4}
$C > E$
A = {2,3,4}
B = {1,2,3,4}
C = {2,3,4}
D = {1,2,3}
E = {1,2,3}
$A \neq C$
$C \neq D$
$B \geq A$
A = {2,3,4}
B = {2,3,4}
C = {2,3,4}
D = {1,2,3}
E = {1,2,3}
$D > E$
A = {2,3,4}
B = {2,3,4}
C = {2,3,4}
D = {2,3}
E = {1,2}
$B \neq C$
A = {2,3,4}
B = {2,3,4}
C = {2,3,4}
D = {2,3}
E = {1,2}
$C \neq D + 1$
A = {2,3,4}
B = {2,3,4}
C = {2,3,4}

D = {2,3}
E = {1,2}

$A > D$
A = {3,4}
B = {2,3,4}
C = {2,3,4}
D = {2,3}
E = {1,2}
$C > E$
$A \neq C$
$C \neq D$
$B \geq A$
A = {3,4}
B = {3,4}
C = {2,3,4}
D = {2,3}
E = {1,2}
$D > E$
$B \neq C$
$C \neq D + 1$

$A > D$
$C > E$
$A \neq C$
$C \neq D$
$B \geq A$
$D > E$
$B \neq C$
$C \neq D + 1$

Resultant Domain:
A = {3,4}
B = {3,4}
C = {2,3,4}
D = {2,3}
E = {1,2}

3. (2 pts) Is arc consistency by itself sufficient to yield a unique solution to the scheduling problem? Briefly explain why or why not.

Arc consistency by itself is not sufficient to yield a unique solution to the scheduling problem. Because there are multiple values in the resultant domain for each variable.

4. (2 pts) Find all valid solutions to the scheduling problem.

Resultant Domain:
A = {3,4}
B = {3,4}
C = {2,3,4}
D = {2,3}
E = {1,2}

$A > D$
$C > E$
$A \neq C$
$C \neq D$
$B \geq A$
$D > E$
$B \neq C$
$C \neq D + 1$

$B \geq A$
$A > D > E$
$C > E$
$A \neq C$
$B \neq C$
$C \neq D$
$C \neq D + 1$

All Valid Solutions:
A = 3, B = 3, C = 4, D = 2, E = 1
A = 4, B = 4, C = 2, D = 3, E = 1

5. (2 pts) Going back to the arc-consistent, but unsolved, CSP in part 2, make a variable assignment that does not appear in a valid solution. Trace the subsequent constraint propagation steps, and indicate which constraint is ultimately violated, leading to backtracking.

Let C = {3}

A = {3,4}
B = {3,4}
C = {3}
D = {2,3}
E = {1,2}

$A > D$ ✓
$C > E$ ✓
$A \neq C$ ✓
A = {4}
B = {3,4}
C = {3}
D = {2,3}
E = {1,2}
$C \neq D$ ✓
A = {4}
B = {3,4}
C = {3}
D = {2}
E = {1,2}
$B \geq A$ ✓
A = {4}
B = {4}
C = {3}
D = {2}
E = {1,2}
$D > E$ ✓
A = {4}
B = {4}
C = {3}
D = {2}
E = {1}
$B \neq C$ ✓
$C \neq D + 1$ **Violated**

Problem 5

1. Run uninformed search on each of the four worlds. Show the DFS and BFS output figures for one of the worlds (your choice) in your document. What do you notice about the empirical time and space complexities of DFS and BFS as indicated by the number of expanded nodes and maximum frontier size? Explain any discrepancies as compared to the theoretical complexities of the two algorithms.

```
python main.py worlds 1
========================================
Testing Grid World Path Planning...
Loading grid world file from path: worlds
Modes: 1. DFS, 2. BFS
Testing world 1

Mode: DFS
Path length: 2583
Path cost: 7896
Number of expanded states: 3665
Max frontier size: 5082

Mode: BFS
Path length: 138
Path cost: 414
Number of expanded states: 8907
Max frontier size: 147

Done
========================================
```

Theoretically:
Depth-First Search:
Suppose search tree has branching factor b and max depth m
Time complexity: (Worst case) number of expanded nodes $O(b\text{^}m)$
Space complexity: (Worst case) number of frontier nodes in memory $O(bm)$

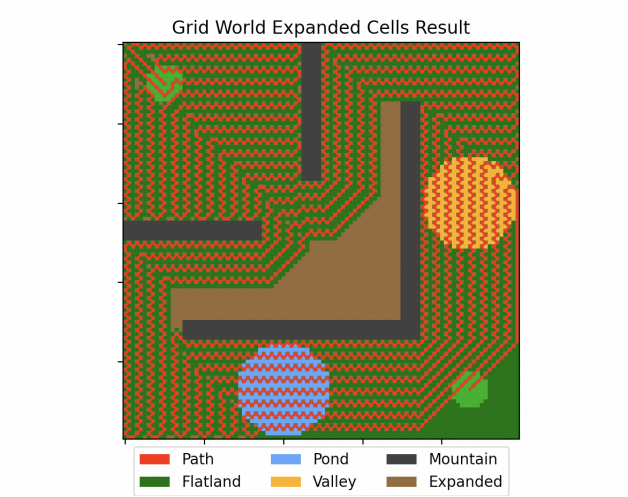
- One node in each of m tiers with b successors each

Breadth-First Search: Worst-case time and space complexities are $O(b\text{^}d)$, depth d

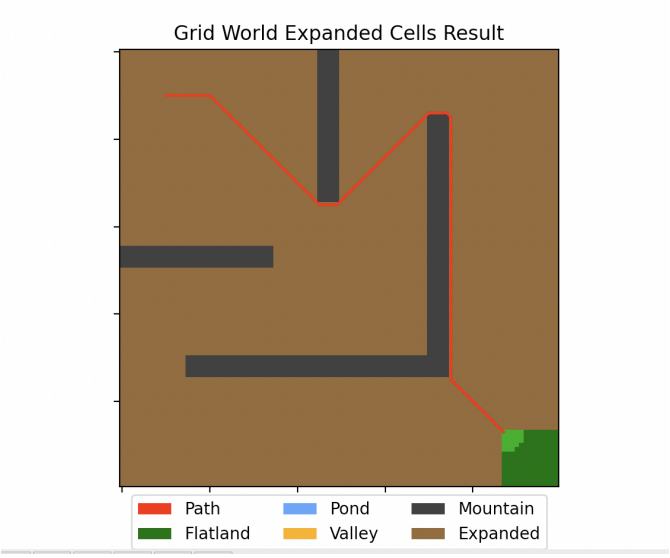
Empirical:
Depth-First Search:
Maximum frontier size (representing space) is greater than number of expanded states (representing time). Space complexity is higher than expected. There might be many branching paths, and DFS needs to keep track of all the paths in memory. It is possible that DFS finds a path that requires fewer expansions but is not optimal.

Breadth-First Search: time complexity significantly higher than expected and space complexity. The large number of expanded states shows that BFS exhaustively explore layer by layer up to the depth where the solution is found.

DFS:


Grid World Expanded Cells Result
Path
Flatland
Pond
Valley
Mountain
Expanded

BFS:


Grid World Expanded Cells Result
Path
Flatland
Pond
Valley
Mountain
Expanded

2. Run A* and beam search (using the default width) on each of the four worlds. You should verify that both heuristics produce similar results. Compare the resultant space complexities of the two algorithms, as well as the optimality of the solutions returned. Show the output figures of the two algorithms for a world in which the solutions for A* and beam search are not identical, and explain why that may occur.

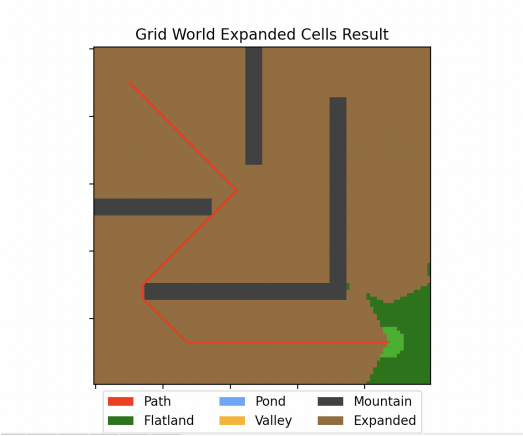
```
python3 main.py worlds 1 -e 1
========================================
Testing Grid World Path Planning...
Loading grid world file from path: worlds
Modes: 1. A_STAR, 2. BEAM_A_STAR
Using Manhattan heuristic
Testing world 1

Mode: A_STAR
Path length: 138
Path cost: 391
Number of expanded states: 8677
Max frontier size: 256

Mode: BEAM_A_STAR
Path length: 140
Path cost: 397
Number of expanded states: 7106
Max frontier size: 100
Done
========================================
```

For both methods:
The expanded states(time complexity) is significantly larger than the maximum frontier size(space complexity).

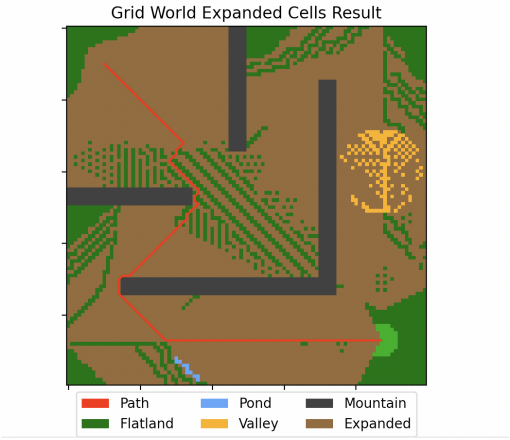
Optimality:
A* is guaranteed to find the optimal solution if one exists.
BEAM_A_STAR search does not guarantee an optimal solution. This is because BEAM_A_STAR expands only a limited number of nodes at each level of the tree, and therefore not guaranteed to find the shortest path. As in our case,path length and path cost for BEAM_A_STAR is slightly higher than the A* method.

From the graph: BEAM_A_STAR is not following straight lines and not avoiding obstacles in an optimal way.

A_STAR:



BEAM_A_STAR:



3. Tuning the beam width in beam search can be a delicate process. Let's focus on world 3. Try some values of width smaller than 100 and observe what changes, if any, occur to the beam search results (using either heuristic). Around what width value do we start seeing a suboptimal solution? Around what width value do we fail to find a solution? Report the two values that you find, and show the output figures for beam search in each case (the latter should show no path).

Using Euclidean heuristic

```
python main.py worlds 3 -e 2
========================================
Testing Grid World Path Planning...
Loading grid world file from path: worlds
Modes: 1. A_STAR, 2. BEAM_A_STAR
Using Euclidean heuristic
Testing world 3
Mode: A_STAR
Path length: 101
Path cost: 303
Number of expanded states: 8184
Max frontier size: 209
Mode: BEAM_A_STAR
Path length: 101
Path cost: 303
Number of expanded states: 6770
Max frontier size: 100
Done
========================================
```

Suboptimal: 35

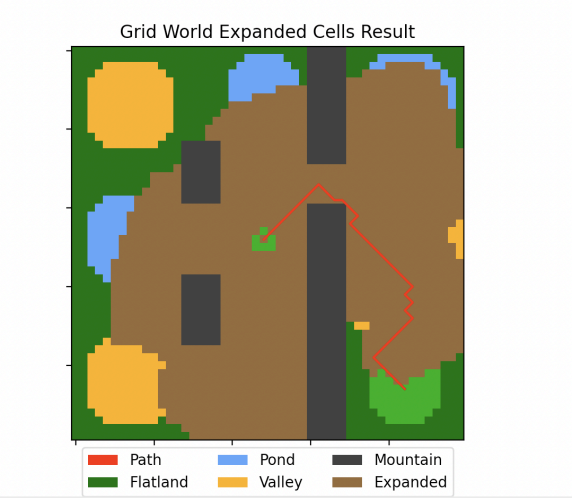
```
python3 main.py worlds 3 -e 2 -b 35
========================================
Testing Grid World Path Planning...
Loading grid world file from path: worlds
Modes: 1. A_STAR, 2. BEAM_A_STAR
Using Euclidean heuristic
Testing world 3
Mode: A_STAR
Path length: 101
Path cost: 303
Number of expanded states: 8184
Max frontier size: 209

Mode: BEAM_A_STAR
Path length: 104
Path cost: 312
Number of expanded states: 3738
Max frontier size: 35

Done
========================================
```

fail to find a solution: <30

```
python3 main.py worlds 3 -e 2 -b 30
========================================
Testing Grid World Path Planning...
Loading grid world file from path: worlds
Modes: 1. A_STAR, 2. BEAM_A_STAR
Using Euclidean heuristic
Testing world 3
Mode: A_STAR
Path length: 101
Path cost: 303
Number of expanded states: 8184
Max frontier size: 209
Mode: BEAM_A_STAR
Path length: 0
Path cost: 0
Number of expanded states: 2601
Max frontier size: 30
Done
========================================
```

4. Run IDA* on world 4 using the Manhattan heuristic. Show the output figure for IDA*.
What do you notice about the optimality of the solution, as well as time and space complexities? In what situations would IDA* be a good use case (compared to regular A*), and when might it not be?

```
python3 main.py worlds 4 -e 3
========================================
Testing Grid World Path Planning...
Loading grid world file from path: worlds
Mode: ID_A_STAR
Using Manhattan heuristic
Testing world 4

Mode: IDA_STAR
Path length: 35
Path cost: 105
Number of expanded states: 247978
Max frontier size: 82

Done
========================================
```



Optimality of the solution:
The solution is the optimal solution, comparing with A* (which is guaranteed to find the optimal solution.) because the path length and path cost is the same as A*. It also seems to be direct.
IDA* is guaranteed to find the optimal solution if one exists.

It requires relatively small frontier size and large total number of states expanded.
IDA* has an advantage in small space complexity, but high time complexity.
IDA* would be a good choice if space complexity is a big concern, where we want to significantly reduce memory.
IDA* is not ideal when we want to have quick responses.

```
python3 main.py worlds 4 -e 1
========================================
Testing Grid World Path Planning...
Loading grid world file from path: worlds
Modes: 1. A_STAR, 2. BEAM_A_STAR
Using Manhattan heuristic
Testing world 4
Mode: A_STAR
Path length: 35
Path cost: 105
Number of expanded states: 1394
Max frontier size: 153
Mode: BEAM_A_STAR
Path length: 35
Path cost: 105
Number of expanded states: 1256
Max frontier size: 100
Done
========================================
```