Introduction to Artificial Intelligence – 236501

**Assignment 1 – State space search**

**Assignment objectives:**

1. Exercise material until Local Search (excluded).
2. Implement algorithms learned in class in Python.
3. Have fun!

**General guidelines:**

1. Submission deadline: 24.5, 23:59.
2. Submission in pairs only.
3. The submission should be typed either in Hebrew or English. Handwritten submission will not be checked.
4. You are welcome and encouraged to ask questions at the assignment [piazza page](https://piazza.com/technion.ac.il/spring2023/236501).
5. TA in charge: **Ron Benchetrit**.
6. Valid requests for extension (e.g., reserve service, hospitalization, etc.) should be sent to the responsible TA (**Sapir Tubul**) only.
7. Updates may be posted to this document throughout the assignment. If so, a notification will be sent, and updates will be highlighted. Updates will be mandatory, and you must stay updated until the submission deadline of the exercise.
8. Please note that the exercise constitutes 15% of the final grade in the course, and therefore, plagiarism will be treated severely!
9. The assignment grade is composed for the following components:
   1. 65% for the theoretic (dry) part.
   2. 35% for the practical (wet) part.

**Theoretic part guidelines:**

1. As a rule of thumb, given a question. First provide the immediate answer and then explain and elaborate. For example, if asked “what is the space complexity of BFS?” an answer would be “. Because in the worst case we store the entire search tree of the problem in the CLOSE list”. Please stick to this format throughout the assignment.

**Practical part guidelines:**

1. We encourage you to first go over the provided files to understand how the environment works and which methods you can use in your implementation.
2. Your code will be thoroughly checked by tests. The tests will check your results against the results obtained by our implementation over different maps. We expect you to receive the same values exactly. We will check, among other things, the received path, its cost, and the number of expansions. Therefore, you must adhere to the instructions in this exercise. The tests will, of course, be time limited. You will be given ample time to run each test.
3. Submit a clear code that can be checked manually.

**Introduction and background:**

The assignment spans over this document and the provided notebook. It is recommended to answer the questions by the order.

In this task, we will deal with the implementation of search algorithms on state spaces for navigation problems. It is recommended to review the relevant lecture and exercise materials before starting the exercise and revisit throughout.

During the exercise, you will be asked to run several experiments and report on their results. You are required to analyze the results, as explained below.

Rick and Morty went on another adventure and this time Rick took Morty on a tour of the Gazorpazorp Bar in the 9th planet of the Alk'taob star. After Rick turns into a sour pickle and gets into a scuffle with a Blarf species creature, they escape from the bar. Rick plans to use his portal gun to get back home (a gun that opens a green gate through which one can teleport to different places), but he discovers that he has run out of portal gun fuel. Morty remembers that there is a fuel depot located at the edge of the frozen lake, but the problem is that they have to cross the lake, which is full of holes.

Luckily for Rick and Morty, you are taking Intro to AI. They ask you to help them plan a path to the fuel depot.





**Question 1 – Introduction (8 pt):**

The theoretical questions in this part are based on the provided board and the frozen lake environment as detailed in the provided notebook unless mentioned otherwise.

1. Wet: go over the notebook until you reach BFS-G part.
2. Dry (1 pt): First we will want to define a Search Space like learned in class. Formally define for the given environment where is the state space, is the operator space, is the initial state and is the set of goal states. What is the size of the state space?
3. Dry (1 pt): What is the output of .
4. Dry (1 pt): What are the successors states of the initial state, .
5. Dry (1 pt): Are there cycles in our search space?
6. Dry (1 pt): What is the branching factor ?
7. Dry (1 pt): In the worst case, how many actions will be required for a general agent to reach a goal?
8. Dry (1 pt): In the best case, how many actions will be required for a general agent to reach a goal?
9. Dry (1 pt): In a general board in the frozen lake environment, the **least cost** path will necessary be a one that reaches a goal state that is closest to the initial state (in terms of ? If so, prove. If not, provide an example.

**Question 2 – Breadth First Search (BFS-G) (7 pt):**

1. Wet: implement BFS-G (BFS graph search) in code.
2. Dry (1 pt): What should be the condition on the search graph (not necessarily in the frozen lake problem) so that BFS on a graph and BFS on a tree generate and expand identical nodes in the same order?
3. Dry (2 pt): For the "4x4" board shown in the notebook, draw the state graph.
4. Dry (2 pt): Given an NxN board without portals, suggest a way to use the BFS-G algorithm to find an optimal (minimum cost) solution and explain.

Hint: You should provide a function T:G→G' that takes the state graph G and creates a new graph G', and use it to find the optimal path in G.

1. Dry (2 pt) Given an NxN board without holes, portals, with regular cells (F, T, A, L), an initial state at the upper-left corner and a goal state at the bottom-right corner, how many nodes will be opened and expanded during the BFS-G search? Explain.

**Question 3 – Depth First Search (DFS-G) (6 pt)**

1. Wet: Implement the DFS-G algorithm.
2. Dry (1 pt): For the frozen lake problem with an NxN board, is the algorithm complete? Is it optimal?
3. Dry (1 pt): Would a DFS algorithm (on a tree) for the frozen lake problem on an NxN board find any solution? If so, what would be the resulting path? If not, how would the algorithm behave?
4. Dry (2 pt) Given an NxN board without holes, portals, with regular cells (F, T, A, L), an initial state at the upper-left corner and a goal state at the bottom-right corner, how many nodes will be opened and expanded during the DFS-G search? Explain.
5. Dry (2 pt) Given an NxN board without holes, portals, with regular cells (F, T, A, L), an initial state at the upper-left corner and a goal state at the bottom-right corner, how many nodes will be opened and expanded during the DFS-G with backtracking (lazy expansion) search? Explain.

**Question 4 – DFS-L (6 pt):**

1. Dry (6 pt): Jerry wants to find a path in the frozen lake using DFS-L. It is given that the path length to the closest goal is . However, Rick limits the search to a depth of .
   1. (2 pt) Propose a modification to the search space such that Jerry will be able to find a path without exceeding the limit imposed by Rick.
   2. (1 pt) Has the branching factor changed? What is the new branching factor ?
   3. (1 pt) What is the time and space complexity? Answer in terms of and . compare it to a regular DFS-L with depth . How would your answer change if we used DFS-L with backtracking?
   4. (2 pt) Provide one example where your modified search performs better than DFS-L with depth and one example where DFS-L with depth performs better. In your answers address the number of nodes expanded. (Examples can be general and not from the frozen lake environment).

**Question 5 – ReverseDFS (6 pt):**

Assume we have some prior knowledge about an upper bound for the depth of a goal state, denoted D. Beth proposed the following search algorithm:

**function ReverseDFS (*problem, D*):**

L D

result failure

**While** Not Interrupted:

*new\_result DFS-L (problem, L)*

*if new\_result = failure:*

*break*

L L - 1

result new\_result

**return** result

In the question below assume there is enough time to complete the first iteration.

* 1. (1 pt) Is it complete? If yes prove. If not, provide an example.
  2. (1 pt) Is it optimal? If yes prove. If not, provide an example.
  3. (2 pt) Provide one example where **ReverseDFS** is better than **ID-DFS** and one example where **ID-DFS** is better than **ReverseDFS**. (Examples can be general and not from the frozen lake environment).
  4. (2 pt) Explain how we can make the algorithm more efficient. Hint: can you think of a better way to update L?

**Question 6 - UCS ( 4pt)**

The theoretical questions in this section are based on the "8x8" board that appears in the notebook, unless specified otherwise.

1. Wet: Complete the missing parts of the UCS algorithm in the file according to the instructions in the notebook.
2. Dry (1 pt): For which search problems will the UCS algorithm and the BFS algorithm behave the same? Explain.
3. Dry (1 pt): Is the algorithm complete for our search problem, for an NxN board? Is it optimal?
4. Dry (2 pt): Dan made a mistake in the implementation of the UCS algorithm and erroneously checked if a node is a goal node instead of expanding it. Provide an example of a search graph for which Dan would still return the optimal path, and an example of a search graph for which Dan would not return the optimal path. For each example, explain what path and cost UCS would return incorrectly, and what path and cost the correct algorithm would return. Note that the search graph does not necessarily have to represent the frozen lake problem. You can provide an example of a different search problem. The graph should contain directed edges and the cost of each edge.

**Question 7 - Heuristics (8 points):**

1. Dry (1 pt): Given two admissible heuristics . is admissible? If yes, prove. If not, provide a counterexample.
2. Dry (1 pt): Given two admissible heuristics . is admissible? If yes, prove. If not, provide a counterexample.
3. Dry (1 pt): Given two consistent heuristics . is consistent? If yes, prove. If not, provide a counterexample.
4. Dry (1 pt): Given two consistent heuristics .. is consistent? If yes, prove. If not, provide a counterexample.

We define a new heuristic for a state space with a single goal state. i.e.,:

where the first term is the Manhattan distance from the current state to the goal state, and the second term is the cost of the edge leading to the portal.

1. Dry (1 pt): Is admissible for all boards? If yes, explain. If not, provide a counterexample.
2. Dry (1 pt): Is consistent for all boards? If yes, explain. If not, provide a counterexample.

We extend to a state space with numerous goal states i.e., by defining:

Note that in this case we take the minimum over all the goal states.

1. Dry (1 pt): Is admissible for all boards? If yes, explain. If not, provide a counterexample.
2. Dry (1 pt): Is consistent for all boards? If yes, explain. If not, provide a counterexample.

**Question 8 - Greedy Best First Search (3 points):**

The theoretical questions in this section are based on an 8x8 grid as shown in the notebook, unless stated otherwise.

1. Wet: Implement the missing parts in the Greedy Best First Search algorithm according to the instructions in the notebook. You should use the heuristic function
2. Dry (1 pt): Is the algorithm complete? Is it optimal?
3. Dry (1 pt): Provide one advantage and one disadvantage of Greedy Best First Search compared to UCS (Uniform Cost Search).
4. Dry (1 pt): Provide one advantage and one disadvantage of Greedy Best First Search compared to Beam search.

**Question 9 - W-A\* (2 points):**

1. Wet: Implement the missing parts in the Greedy Best First Search algorithm according to the instructions in the notebook. You should use the heuristic function
2. Dry (2 pt) For , denote the paths from the initial state to the goal state, returned by W-A\* under the formulation, for and by and (respectively). Then , for:
   1. An admissible heuristic ? If yes, explain. If not, provide a counterexample.
   2. A general (not necessarily admissible) heuristic ? If yes, explain. If not, provide a counterexample.

**Question 10 –IDA\* (2 points):**

1. Wet: Implement the missing parts in the IDA\* algorithm in the file according to the instructions provided there. You should use the heuristic function
2. Dry (2 pt): Provide one advantage and one disadvantage of IDA\* compared to A\*. In which cases you would prefer the former and which the latter?

**Question 11 - A\*-epsilon (6 points):**

1. Dry (2 pt): Provide an advantage and a disadvantage of A\*-epsilon compared to A\*.
2. Dry (4 pt): Propose a heuristic to select the next node to expand from FOCAL. Describe the heuristic and compare it to using in terms of number of expanded nodes, returned path and path cost.

**Question 12 – Benchmarking (2 points):**

In this question we would like to compare the implemented algorithms on different problems. Run the relevant section in the notebook. You should see a csv file (can be opened with excel) showing the results of each algorithm.

1. Wet: run the experiments in the notebook.
2. Dry (2 pt): Explain the results. Do they match your expectation? How would you think the results would change with a more informed heuristic? Analyze and explain the results in terms of solution cost and number of expansions. Note that in this section there is no right or wrong answer, but you are required to provide detailed answers.

**Question 13 – Local Search (5 points)**

Given the following state space, where is the initial state and a utility function where the value of each state is indicated in each state.

Our goal is to find a state that maximizes value.

We will use the Stochastic Hill Climbing algorithm. **.**

1. What are the probabilities to transition from the initial state to each of the states ? Write down and .
2. What is the maximal number of steps the algorithms can do, where a step is a transition between states?
3. Given that the first transition if from the initial state to state . Will the algorithm reach an optimal (global) solution?
4. What is the probability that the algorithm reaches a suboptimal solution? Write your answer as and explain.
5. What is the range of values of such that the probability of reaching an optimal (global) solution in exactly 3 steps is bigger than ?

**Submission:**

You should submit a single zip file named AI1\_<id1>\_<id2>.zip (without the brackets) that contains:

1. A file named AI1\_<id1>\_<id2>.pdf which contains your answers for the theoretical questions.
2. Algorithms.py which contains your python implementation of the practical part.