Problem assignment 1

Due: Wednesday, February 5, 2025

Problem 1. Search Method Review

Consider the following graph that represents road connections between different cities. The weights on links represent driving distances between connected cities. Let S be the initial city and G the destination.

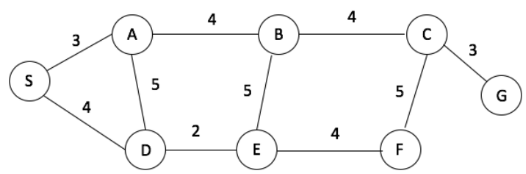

Figure 1:

Part a. Show how the uniform cost search works by giving the **order in which the nodes of the search tree are expanded**. Is the path found by the algorithm optimal?

Part b. Assume the following set of straight line distances between G and other cities:

S	A	B	C	D	E	F
10	10	7	2	9	6	2

Show how the greedy search algorithm with the straight-line distance heuristic works. Is the path the algorithm finds optimal?

Part c. Show how the A^* algorithm with straight-line distance heuristic works. Is the path found optimal?

Problem 2. Search for the Puzzle 8 problem.

In this problem we continue our exploration of search algorithms for the Puzzle 8 problem. We will use the evaluation-function driven search procedure to incorporate various exploration strategies. The procedure searches the space by expanding the nodes with the minimum evaluation function value first. You are given three files:

- `Puzzle8.py` which gives the definition of the Puzzle 8 problem, and TreeNode, HashTable, and Priority queue structures implemented as classes. Please note this file is slightly different from `Puzzle8.py` file you were given for homework assignment 1!
- `f_driven_search.py` which implements an evaluation function driven search algorithm. Briefly, the procedure searches the space by expanding the nodes in the exploration fringe with the minimum `f_value`. These nodes are kept in the priority queue.
- `heuristic.py` that calculates the h function for the uniform cost search.

Part a. Uniform cost search

The `f_driven_search.py` code we gave you allows you to modify/update the evaluation function driven search, as well as, use your own heuristic function by importing a new definition of the h function. This function together with the g-value for the node (automatically calculated) defines the f-value of the node. The files given to you implement the uniform cost search where $h(n) = 0$ and hence $f(n) = g(n)$.

Remark: The uniform cost search algorithm for the Puzzle-8 problem in fact implements the breadth-first search since all operator costs are one. The difference is that we simulate the breadth-first search through a more flexible evaluation-function representation and priority queue operations.

The `f_driven_search.py` currently does not calculate any search statistics similarly to the initial code you were initially given in homework assignment 1. Please define a new version of the `eval_function_driven_search(problem)` such that it calculates the following stats:

- the total number of nodes generated
- the total number of nodes expanded
- the maximum length of the queue
- the length of the solution

Include the new function in file `main2a.py`. Run it on at least first three initial game configurations and report statistics.

Part b. Uniform cost search with elimination of state repeats.

Modify the function `eval_function_driven_search(problem)` in the `main2a.py` file to include the check and elimination of all state repeats. Call the new function: `eval_function_driven_search_repeats(problem)` and include it in file `main2b.py`. Your program should be able to solve all 5 example configurations.

Part c. A algorithm with the misplaced tile heuristic*

Our next step is to implement the A* search procedure with the misplaced tiles heuristic. In order to do so you will need to write a new `h_function` definition and import it to the `Puzzle8.py` file. Please write `heuristic1.py` file that implements the `h_function` using the misplaced tile heuristic. Run `main2b.py`. with `Puzzle8.py` importing the `h_function` from `heuristic1.py` instead of the current `heuristic.py`. The program should run on all five test examples and collect the same set of statistics as above.

Part d. A algorithm with the Manhattan distance heuristic*

Similarly to Part c, write heuristic2.py that implements the Manhattan distance heuristic. Run main2b.py with `Puzzle8.py` importing the h function from heuristic2.py.

Part e. Analysis of results

Analyze the performance of all methods (parts a through d) in terms of the collected statistics and include the analysis in the report. You should:

- Summarize the results of the methods in different tables, one table for every configuration tested: Uniform cost search, Uniform cost search with elimination of repeats, A* with misplaced tile heuristic, A* with Manhattan distance heuristics.
- Which method is the best in terms of the respective statistics? Explain why.
- State which heuristic you would suggest to use and explain why.

In addition, answer the following questions.

- Would A* work without state repeats elimination? Why or why not?
- Assume we create a heuristic function h_3 such that it averages the values of the misplaced tile heuristic (h_1) and the Manhattan distance heuristic (h_2):

$$h_3(n) = \frac{1}{2}\left[h_1(n) + h_2(n)\right]$$

  Is h_3 an admissible heuristic? You must demonstrate why or why not.

Code to be submitted for Problem 2:

1. `main2a.py`
2. `main2b.py`
3. `heuristic1.py`
4. `heuristic2.py`

Please note the TA for the course will run your code to check if the code is consistent with the reported results.