Programming Assignment 1 - Park Restroom Planning

Deadline: 3/24

General Instructions

Please modify the code in **submission.py** between

 # BEGIN_YOUR_CODE

and

 # END_YOUR_CODE

and you could add other functions outside the block if you want. Do not make changes to files other than submission.py.

Your code will be evaluated on two types of test cases, basic and hidden. The input files of the tasks are in the directory task_in, and the directory task_in should be in the same directory as your code. For basic tests, such as task-0-0, task-0-1, task-1-0, and task-1-1, the input files and answers are fully provided to you. For hidden tests, your code will be evaluated on line even though the input file of the task-2 is provided to you but the answer is not.

To run the tests, you should have **grader.py** and **graderUtil.py** in the same directory as your code. You can run single task by typing

python3 submission.py task_0_0.txt

to get the output of the final result. You also can run all the tasks by typing

python3 grader.py

to derive the score of your code.

You should submit your submission.py to moodle.

Problem

Where you place a restroom is key to the park layout. Because kids wait until the last minute to warn they need to go to the restroom, it is best to locate the restroom near the playground. Given a park, which is represented by $n \times m$ grids, and l playgrounds, please locate k restrooms with the minimum cost. The cost is defined as follows:

$$\sum_{i=1}^{l} min\{dist(p_i, r_1), \ldots, dist(p_i, r_k)\} ,$$

where $dist(p_i, r_j)$ is the Manhattan distance between playground p_i and restroom r_j.

For example, in Figure 1, given a park, which is 4×4 grids, four playground p_1, p_2, p_3, and p_4, and two restrooms r_1 and r_2, the cost is 7, i.e., 1+3+2+3=9.

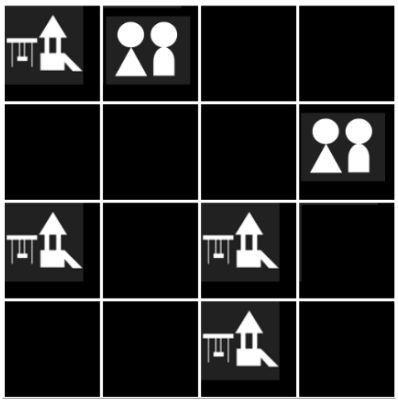

Figure 1

Task 0-0: (5%,5%,5%) Given an input task_0_0.txt, please use a **hill-climbing search** algorithm for locating the restroom. In the task_0_0.txt, as shown below, the integer in the first line indicates the search algorithm type, i.e., 0 for hill climbing search, the park is 4 x 4 grids, four playgrounds located in (0,0), (2,1), (2,2), (3,2), and one restroom with the initial location (0,3).

```
0
4,4
4|0,0|2,1|2,2|3,2
1|0,3
```

Your program should output a dictionary consisting of the initial cost, the minimal cost, the locations of restrooms. For example,

```
{“ini_cost”: 15, “best_cost”: 9, “locations”: [[1,2]]}
```

Task-0-1: (10%,10%) Similarly, Given an input as task_0_1.txt, please use a **random-restart hill climbing search** for locating the restroom. In the task_0_1.txt, as shown below, the integer in the first line indicates the search algorithm type, i.e., 1 for random-restart hill climbing search, the park is 4 x 4 grids, four playgrounds locate in (0,0), (2,0), (2,2), (3,2), the number of restrooms should be one, and run the hill climbing search with 10 restarts to get the minimum cost.

```
1
4,4
4|0,0|2,0|2,2|3,2
1
10
```

Your program should output a dictionary consisting of the minimal cost and the location of the restroom. For example,

```
{“best_cost”: 7, “locations”: [[2,1]]}
```

Task-1-0: (5%,10%,10%) Given an input as task_1_0.txt, please use a **hill-climbing search** algorithm for locating the restrooms. Your program should output a dictionary consisting of the initial cost, the minimal cost, the locations of restrooms. For example,

```
{“ini_cost”: 9, “best_cost”: 7, “locations”: [[1,0],[1,2]]}
```

Task-1-1: (20%) Similarly, Given an input as task_1_1.txt, please use a **random-restart hill climbing search** for locating the restrooms. Your program should output a dictionary consisting of the minimal cost and the locations of restrooms. For example,

```
{“best_cost”: 5, “locations”: [[1,0],[2,1]]}
```

Task-2: (10%) Similarly, Given an input as task_2_1.txt, please use a **random-restart hill climbing search** for locating the restrooms. Your program should output a dictionary consisting of the minimal cost and the locations of restrooms.

```
{“best_cost”: ?, “locations”: [?]}
```

Task-3(On-line): (10%) Given an on-line input, task_3_1.txt, it would test your code of a **random-restart hill climbing search** for locating the restrooms. Your program should output consisting of the minimal cost and the locations of restrooms.

{“best_cost”: ?, “locations”: [?]}