Lab 1

姓名：游灏溢
班级：F1903302
时间：23/9/2021

Abstract

This lab mainly helps us to be familiar with the programming using **python**. After that, we use the python to realize the search algorithm

Content

Data structure

Stack

- **push()**: list adds the item in the end. Here I use the function append().
- **pop()**: list pop the last item. Here I use the list function pop().
- **is_empty()**: judge whether the length of the list is zero.

Queue

- **push()**: list adds the item in the end. Here I also use the function append().
- **pop()**: list pop the first item. Here I use the list function pop(0).
- **is_empty()**: judge whether the length of the list is zero.

For more details please refer to the file **util.py**

Search Algorithm

Depth First Algorithm(DFS)

- 1.Add the start position into the stack.
- 2.Pop the top item of the stack, search all the possible actions and destinations.
- 3.Judge the new destinations whether to be the goal.
- 4.If the destination is the goal, return the actions from the start position to here.
- 5.Otherwise, push the new destinations and its corresponding actions to there. Repeat the 2^{th}action
- 6.If the stack is empty, the algorithm fail and return an empty list.

Breadth First Algorithm(BFS)

- 1.Add the start position into the queue.
- 2.Pop the first item of the queue, search all the possible actions and destinations.
- 3.Judge the new destinations whether to be the goal.
- 4.If the destination is the goal, return the actions from the start position to here.
- 5.Otherwise, push the new destinations and its corresponding actions to there. Repeat the 2^{th}action
- 6.If the queue is empty, the algorithm fail and return an empty list.

For more details please refer to the file **algorithm.py**

Algorithm improvement

So it is necessary to improve the DFS algorithm. When analyzing the reason, it is the cycle causing DFS not finishing. So we only need to record the positions the search reaches. When the search process reaches a position not for the first time, it will ignore it. Then the algorithm will end.

Here I use the set data structure(hash table) to record the positions the search has reached.

For more details, please refer to the **algorithm_modify.py**

For the case that the DFS algorithm, we only need to build the graph that has cycle in the actions with higher priority. Since the West action is the last action in the function **get_successor()**, and using stack causes that it will choose the last action firstly. So it only needs to construct the graph based on this case. Here I construct a graph in the file **testMaze1.lay**

Problem answers

1. Please refer to the data sturcture part.
2. Here I provide 2 algorithm, one in **algorithm.py** and another in **algorithm_modify.py**, and the introduction is in DFS part.
3. Here I provide 2 algorithm, one in **algorithm.py** and another in **algorithm_modify.py**, and the introduction is in BFS part.
4. The maze world is in **testMaze1.lay**, which is the situation that DFS will be not able to stop and get the result, more explanation please refer to the Algorithm improvement parts

   Also, the DFS algorithm may not get the optimal solution as long as the first path it searches is not the shortest ones. Here I alsoo propose another situation that DFS will get a solution not optimal in **testMaze2.lay** .