Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

- Note: For more clarity of the code, please refer to the IPNB file first
- ***Note: In the latest version, the outputs are returned in the main function***

❖ Introduction:

Our project is in the field of making a game(pack_man) and finding a path to food by means of the function of the agent and the environment:

Now first we have input file like demofile.txt:

The project has five parts. The first part is writing the agent function and the environment

The agent function is:

```
def agent(percept):
  randomAction=0
  if percept[2]=='a':
     percept[2]='-'
  if percept[2]=='f':
    return 20  #--------------------------->this number is my random selected numer that i selecte so we can set each all number or string or character choosed for this
  elif percept[2]=='*':
    randomAction=setblockstatus(actions)
  else:
    randomAction=random.randint(1,4)
  actions.append(randomAction)
  return randomAction
```

the environment is:

```
def enviroment(action):
  setStatusPosition(action)
  current_situation=original_list[statusPos[0]][statusPos[1]]
  current_percept=[statusPos[0],statusPos[1],current_situation]
  historyOfPercepts.append(current_percept)
  return current_percept
```

Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

As you can see, a fraction of other functions have been called inside these functions, and for the purpose of brevity of the code, I have included only the main function to clear up the ambiguities

- In the second part of the project, there is a random map creation, the code of which is given as a part of the project code:

```
# ------------------------------------------------------------>create RANDOM_MAP
def generate_map(m, n):
    # ---------------------Define characters for terrain and objects
    terrain_chars = ['-', '*']
    object_chars = ['a', 'f']

    # ---------------------Generate empty map
    map_data = [['-' for _ in range(n)] for _ in range(m)]

    # ---------------------Place walls around the perimeter
    for i in range(m):
        for j in range(n):
            if i == 0 or i == m - 1 or j == 0 or j == n - 1:
                map_data[i][j] = '*'

    # ---------------------Place random walls within the houses
    for i in range(1, m - 1):
        for j in range(1, n - 1):
            if map_data[i][j] == '-':
                if random.random() < 0.2:  # Adjust the probability as needed
                    map_data[i][j] = '*'

    # ----------------------Place objects on the map
    for char in object_chars:
        while True:
            row = random.randint(1, m - 2)
            col = random.randint(1, n - 2)
            if map_data[row][col] == '-':
                map_data[row][col] = char
                break
```

Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

```
    return map_data
    # ----------------find positions of current_agent_position'
def find_character_position(map_data, character):
    for i, row in enumerate(map_data):
        for j, char in enumerate(row):
            if char == character:
                return i, j
    return None, None
f=open("demofile.txt","a")
f.truncate(0)
f.close()
def write_map_to_file(map_data, filename):
    with open(filename, "r+") as f:
        f.write(f'{len(map_data)},{len(map_data[0])}\n')
        a_row,a_col = find_character_position(map_data,'a')
        f.write(f'{a_row},{a_col}\n')
        f.write('\n')
        count=0
        for row in map_data:
            count+=1
            if count!=len(map_data):
              f.write(''.join(row) + '\n')
            else:
              f.write(''.join(row))
            # f.write(''.join(row) + '\n')

# Example usage:
print("now you can create random map by m and n that you enter them as size of map(m*n):\n")
print("--> enter the m(max_num of rows):")
m = input()
print("-->enter the n(max_num of cols):")
n = input()
m=int(m)
n=int(n)
map_data = generate_map(m, n)
write_map_to_file(map_data, "demofile.txt")
f.close()
```

All these two pages of the code that you have seen, the generatemap function is related to the creation of the map randomly, and in the next functions, the map is written to file using the default characters of the project;like(‘*’or ’a’ or ’f’….).

Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

- In the third part: My project does not have a graphic environment 😊

- In the fourth and fifth sections: we have a series of inputs and outputs, which I have brought to you with examples of given inputs and output of the code

Input 1:
10,10
2,2

****
___-*
-a---*-*
*****-****
****_____*
***-**___*
*-****-***
*__*___f-*
*__*__*__*

Output1:

Coordinates of the starting point :
(2, 2)
The coordinates of the place of food :
(7, 7)
the number of step to find path using 'bruth_forts' mehthod :
289
The list consists of the selected coordinate and motion pairs:
[[(2, 2), 4, (2, 1), 3, (3, 1), 1, (2, 1), 3, (3, 1), 1, (2, 1), 2, (2, 2), 3, (3, 2), 1, (2, 2), 3, (3, 2), 1, (2, 2), 3, (3, 2), 1, (2, 2), 2, (2, 3), 4, (2, 2), 3, (3, 2), 1, (2, 2), 4, (2, 1), 3, (3, 1), 1, (2, 1), 4, (2, 0), 2, (2, 1), 2, (2, 2), 1, (1, 2), 3, (2, 2), 4, (2, 1), 3, (3, 1), 1, (2, 1), 4, (2, 0), 2, (2, 1), 3, (3, 1), 1, (2, 1), 4, (2, 0), 2, (2, 1), 1, (1, 1), 3, (2, 1), 2, (2, 2), 1, (1, 2), 3, (2, 2), 3, (3, 2), 1, (2, 2), 1, (1, 2), 4, (1, 1), 2, (1, 2), 1, (0, 2), 3, (1, 2), 4, (1, 1), 2, (1, 2), 2, (1, 3), 4, (1, 2), 2, (1, 3), 2, (1, 4), 4, (1, 3), 2, (1, 4), 4, (1, 3), 3, (2, 3), 1, (1, 3), 1, (0, 3), 3, (1, 3), 1, (0, 3), 3, (1, 3), 4, (1, 2), 1, (0, 2), 3, (1, 2), 3, (2, 2), 3, (3, 2), 1, (2, 2), 2, (2, 3), 4, (2, 2), 1, (1, 2), 4, (1, 1), 2, (1, 2), 4, (1, 1), 2, (1, 2), 1, (0, 2), 3, (1, 2), 4, (1, 1), 2, (1, 2), 2, (1, 3), 1, (0, 3), 3, (1, 3), 2, (1, 4), 4, (1, 3), 4, (1, 2), 4, (1, 1), 2, (1, 2), 4, (1, 1), 2, (1, 2), 3, (2, 2), 3, (3, 2), 1, (2, 2), 1, (1, 2), 2, (1, 3), 4, (1, 2), 2, (1, 3), 3, (2, 3), 1, (1, 3), 3, (2, 3), 1, (1, 3), 2, (1, 4), 2, (1, 5), 2, (1, 6), 4, (1, 5), 3, (2, 5), 1, (1, 5), 3, (2, 5), 3, (3, 5), 1, (2, 5), 3, (3, 5), 2, (3, 6), 4, (3, 5), 2, (3, 6), 4, (3, 5), 4, (3, 4), 2, (3, 5), 3, (4, 5), 2, (4, 6), 4, (4, 5), 2, (4, 6), 4, (4, 5), 3, (5, 5), 1, (4, 5), 2, (4, 6), 2, (4, 7), 2, (4, 8), 2, (4, 9), 4, (4, 8), 3, (5, 8), 1, (4, 8), 3, (5, 8), 2, (5, 9), 4, (5, 8), 3, (6, 8), 1, (5, 8), 2, (5, 9), 4, (5, 8), 1, (4, 8), 1, (3, 8), 3, (4, 8), 4, (4, 7), 3, (5, 7), 1, (4, 7), 4, (4, 6), 4, (4, 5), 4, (4, 4), 4, (4, 3), 2, (4, 4), 2, (4, 5), 4, (4, 4), 1, (3, 4), 3,

Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

(4, 4), 3, (5, 4), 1, (4, 4), 2, (4, 5), 2, (4, 6), 1, (3, 6), 3, (4, 6), 2, (4, 7), 4, (4, 6), 3, (5, 6), 1, (4, 6), 3, (5, 6), 1, (4, 6), 2, (4, 7), 2, (4, 8), 1, (3, 8), 3, (4, 8), 2, (4, 9), 4, (4, 8), 1, (3, 8), 3, (4, 8), 3, (5, 8), 1, (4, 8), 3, (5, 8), 1, (4, 8), 3, (5, 8), 1, (4, 8), 1, (3, 8), 3, (4, 8), 3, (5, 8), 1, (4, 8), 3, (5, 8), 1, (4, 8), 2, (4, 9), 4, (4, 8), 3, (5, 8), 2, (5, 9), 4, (5, 8), 1, (4, 8), 2, (4, 9), 4, (4, 8), 1, (3, 8), 3, (4, 8), 2, (4, 9), 4, (4, 8), 4, (4, 7), 1, (3, 7), 3, (4, 7), 1, (3, 7), 3, (4, 7), 3, (5, 7), 4, (5, 6), 2, (5, 7), 1, (4, 7), 3, (5, 7), 2, (5, 8), 2, (5, 9), 4, (5, 8), 2, (5, 9), 4, (5, 8), 4, (5, 7), 4, (5, 6), 4, (5, 5), 2, (5, 6), 1, (4, 6), 1, (3, 6), 3, (4, 6), 3, (5, 6), 2, (5, 7), 4, (5, 6), 2, (5, 7), 2, (5, 8), 2, (5, 9), 4, (5, 8), 4, (5, 7), 2, (5, 8), 1, (4, 8), 2, (4, 9), 4, (4, 8), 1, (3, 8), 3, (4, 8), 3, (5, 8), 3, (6, 8), 1, (5, 8), 2, (5, 9), 4, (5, 8), 1, (4, 8), 3, (5, 8), 1, (4, 8), 4, (4, 7), 3, (5, 7), 4, (5, 6), 1, (4, 6), 1, (3, 6), 3, (4, 6), 2, (4, 7), 1, (3, 7), 3, (4, 7), 3, (5, 7), 1, (4, 7), 2, (4, 8), 2, (4, 9), 4, (4, 8), 2, (4, 9), 4, (4, 8), 4, (4, 7), 1, (3, 7), 3, (4, 7), 2, (4, 8), 3, (5, 8), 2, (5, 9), 4, (5, 8), 1, (4, 8), 4, (4, 7), 2, (4, 8), 3, (5, 8), 1, (4, 8), 1, (3, 8), 3, (4, 8), 1, (3, 8), 3, (4, 8), 2, (4, 9), 4, (4, 8), 3, (5, 8), 4, (5, 7), 4, (5, 6), 3, (6, 6), 3, (7, 6), 1, (6, 6), 2, (6, 7), 4, (6, 6), 2, (6, 7), 4, (6, 6), 2, (6, 7), 4, (6, 6), 4, (6, 5), 2, (6, 6), 3, (7, 6), 4, (7, 5), 2, (7, 6), 2, (7, 7)]] :

****

**----*??*

--?-???*

?*-*

??-----*

*???**---*

????-***

????--f?

?????????

****

the number of step to find shortest path :

12

the short path is :

[(2, 2), 1, (1, 2), 2, (1, 3), 2, (1, 4), 2, (1, 5), 3, (2, 5), 3, (3, 5), 3, (4, 5), 2, (4, 6), 3, (5, 6), 3, (6, 6), 3, (7, 6), 2, (7, 7)]

- First point: According to the description of the project section of the project: (Write programs that play the game can find the food), so according to my understanding of it, in the first part of the project, the goal is to write a program using the agent function and the environment that can only find a path,actually that is, the goal is not to find the optimal path in the first part.

- Second point: As you can see in the first part, all the paths traveled by the agent function are printed. The reason is that the number of steps is high and the path is long. Due to the irrationality of the agent and the lack of recognition of the traveled path in each step, my algorithm It chooses randomly and finally acts in the same way until it reaches the food, so our method is based on brute-force.
- The third point, considering that the agent is irrational and moves randomly and irrationally every time, so it is possible to take very high steps so that the computer is running for several minutes, because our algorithm is not optimal and in If this problem occurs, run the code again.

Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

- 4th point: according to the request of the project, according to the table below, the project asked us all the routes traveled from start to reaching the food, which is in the form of a list, that is, the same as the first list.

4. دنباله تمام حرکات انجام شده تا رسیدن به غذا(به صورت یک سری از محل خانه ها و حرکت انجام شده)

```
Number Of Moves : 38
Moves Steps :
[8,3]-->UP-->[7,3]-->DOWN-->[8,3]-->LEFT-->[8,2]-->RIGHT-->[8,3]-->UP-->[7,3]-->RIGHT-->[7,4]-->LEFT-->[7,3]-->LEFT-->[7,2]-->RIGHT-->
[7,3]-->RIGHT-->[7,4]-->UP-->[6,4]-->DOWN-->[7,4]-->RIGHT-->[7,5]-->LEFT-->[7,4]-->UP-->[6,4]-->RIGHT-->[6,5]-->LEFT-->[6,4]-->RIGHT-->[6,5]-->UP-->
[5,5]-->DOWN-->[6,5]-->RIGHT-->[6,6]-->LEFT-->[6,5]-->DOWN-->[7,5]-->UP-->[6,5]-->UP-->[5,5]-->RIGHT-->[5,6]-->LEFT-->[5,5]-->RIGHT-->[5,6]-->UP-->
[4,6]-->DOWN-->[5,6]-->RIGHT-->[5,7]-->LEFT-->[5,6]-->DOWN-->[6,6]-->UP-->[5,6]-->UP-->[4,6]-->UP-->[3,6]-->DOWN-->[4,6]-->RIGHT-->[4,6]
```

تصویر 2: نمونه دنباله حرکات انجام شده

- The fifth point: In part number five, it is considered that it has a memory input, so in this part it is asked that it asks us for the shortest path considering the memory, so we can choose one of the tree or graph search methods considering the optimality and According to the analysis search, I have come to this result to use the IDS search that I I have given its code below:

```
def get_neighbors(node, map):
    neighbors = []
    row, col = node
    for dr, dc in [(0, 1), (1, 0), (0, -1), (-1, 0)]:
        new_row, new_col = row + dr, col + dc
        if 0 <= new_row < len(map) and 0 <= new_col < len(map[0]) and map[new_row][new_col] != '*':
            neighbors.append((new_row, new_col))
    return neighbors
def dls_shortest_path(map, start, goal, depth):
    if start == goal:
        return [start]
    if depth == 0:
        return None
    for neighbor in get_neighbors(start, map):
        result = dls_shortest_path(map, neighbor, goal, depth - 1)
        if result is not None:
```

```
            return [start] + result
    return None

def ids_shortest_path(map, start, goal):
    depth = 0
    while True:
        result = dls_shortest_path(map, start, goal, depth)
        if result is not None:
            return result
        depth += 1
```

- this code have tree function that using into each.

- Sixth point:As you can see in the output of the first example, I have found the optimal route using IDS and the number of steps of the optimal route along with the list of the traveled route, similar to the first list, and printed it.
- Seventh point: As you can see, I printed my outputs as above to avoid some ambiguities.
- Input2:

  10,10
  8,8

  ****
  ___-*
  *__*__*-*
  *-***-****
  *-f*____*
  ***-**__*
  *-*___***
  *__*____*
  *__*__*-a
  ****

10,10
9,9
Coordinates of the starting point :
(8, 8)
The coordinates of the place of food :
(4, 2)
the number of step to find path using 'bruth_forts' mehthod :
549
The list consists of the selected coordinate and motion pairs:

[[(8, 8), 4, (8, 7), 1, (7, 7), 2, (7, 8), 1, (6, 8), 3, (7, 8), 4, (7, 7), 2, (7, 8), 3, (8, 8), 4, (8, 7), 4, (8, 6), 2, (8, 7), 2, (8, 8), 1, (7, 8), 3, (8, 8), 3, (9, 8), 1, (8, 8), 2, (8, 9), 4, (8, 8), 1, (7, 8), 2, (7, 9), 4, (7, 8), 3, (8, 8), 4, (8, 7), 4, (8, 6), 2, (8, 7), 4, (8, 6), 2, (8, 7), 4, (8, 6), 2, (8, 7), 4, (8, 6), 2, (8, 7), 3, (9, 7), 1, (8, 7), 2, (8, 8), 3, (9, 8), 1, (8, 8), 1, (7, 8), 3, (8, 8), 3, (9, 8), 1, (8, 8), 3, (9, 8), 1, (8, 8), 1, (7, 8), 2, (7, 9), 4, (7, 8), 3, (8, 8), 2, (8, 9), 4, (8, 8), 2, (8, 9), 4, (8, 8), 3, (9, 8), 1, (8, 8), 1, (7, 8), 4, (7, 7), 2, (7, 8), 2, (7, 9), 4, (7, 8), 3, (8, 8), 2, (8, 9), 4, (8, 8), 3, (9, 8), 1, (8, 8), 1, (7, 8), 4, (7, 7), 2, (7, 8), 2, (7, 9), 4, (7, 8), 1, (6, 8), 3, (7, 8), 1, (6, 8), 3, (7, 8), 1, (6, 8), 3, (7, 8), 3, (8, 8), 3, (9, 8), 1, (8, 8), 1, (7, 8), 3, (8, 8), 2, (8, 9), 4, (8, 8), 2, (8, 9), 4, (8, 8), 2, (8, 9), 4, (8, 8), 4, (8, 7), 1, (7, 7), 3, (8, 7), 1, (7, 7), 1, (6, 7), 3, (7, 7), 2, (7, 8), 3, (8, 8), 4, (8, 7), 1, (7, 7), 1, (6, 7), 3, (7, 7), 3, (8, 7), 2, (8, 8), 2, (8, 9), 4, (8, 8), 2, (8, 9), 4, (8, 8), 2, (8, 9), 4, (8, 8), 4, (8, 7), 3, (9, 7), 1, (8, 7), 4, (8, 6), 2, (8, 7), 1, (7, 7), 4, (7, 6), 3, (8, 6), 1, (7, 6), 4, (7, 5), 1, (6, 5), 3, (7, 5), 2, (7, 6), 2, (7, 7), 4, (7, 6), 3, (8, 6), 1, (7, 6), 2, (7, 7), 2, (7, 8), 2, (7, 9), 4, (7, 8), 3, (8, 8), 1, (7, 8), 1, (6, 8), 3, (7, 8), 2, (7, 9), 4, (7, 8), 1, (6, 8), 3, (7, 8), 3, (8, 8), 4, (8, 7), 3, (9, 7), 1, (8, 7), 4, (8, 6), 2, (8, 7), 2, (8, 8), 2, (8, 9), 4, (8, 8), 4, (8, 7), 3, (9, 7), 1, (8, 7), 1, (7, 7), 3, (8, 7), 4, (8, 6), 2, (8, 7), 4, (8, 6), 2, (8, 7), 2, (8, 8), 3, (9, 8), 1, (8, 8), 2, (8, 9), 4, (8, 8), 1, (7, 8), 2, (7, 9), 4, (7, 8), 2, (7, 9), 4, (7, 8), 3, (8, 8), 4, (8, 7), 3, (9, 7), 1, (8, 7), 3, (9, 7), 1, (8, 7), 3, (9, 7), 1, (8, 7), 1, (7, 7), 2, (7, 8), 4, (7, 7), 1, (6, 7), 3, (7, 7), 2, (7, 8), 1, (6, 8), 3, (7, 8), 4, (7, 7), 1, (6, 7), 3, (7, 7), 3, (8, 7), 1, (7, 7), 4, (7, 6), 2, (7, 7), 2, (7, 8), 3, (8, 8), 3, (9, 8), 1, (8, 8), 3, (9, 8), 1, (8, 8), 1, (7, 8), 1, (6, 8), 3, (7, 8), 2, (7, 9), 4, (7, 8), 1, (6, 8), 3, (7, 8), 3, (8, 8), 2, (8, 9), 4, (8, 8), 1, (7, 8), 3, (8, 8), 4, (8, 7), 1, (7, 7), 4, (7, 6), 3, (8, 6), 1, (7, 6), 2, (7, 7), 4, (7, 6), 4, (7, 5), 4, (7, 4), 4, (7, 3), 2, (7, 4), 2, (7, 5), 1, (6, 5), 2, (6, 6), 1, (5, 6), 2, (5, 7), 3, (6, 7), 1, (5, 7), 1, (4, 7), 4, (4, 6), 2, (4, 7), 3, (5, 7), 3, (6, 7), 1, (5, 7), 3, (6, 7), 1, (5, 7), 2, (5, 8), 3, (6, 8), 1, (5, 8), 1, (4, 8), 3, (5, 8), 3, (6, 8), 1, (5, 8), 2, (5, 9), 4, (5, 8), 3, (6, 8), 1, (5, 8), 1, (4, 8), 4, (4, 7), 1, (3, 7), 3, (4, 7), 4, (4, 6), 4, (4, 5), 4, (4, 4), 2, (4, 5), 4, (4, 4), 4, (4, 3), 2, (4, 4), 1, (3, 4), 3, (4, 4), 2, (4, 5), 1, (3, 5), 3, (4, 5), 3, (5, 5), 1, (4, 5), 1, (3, 5), 3, (4, 5), 1, (3, 5), 4, (3, 4), 2, (3, 5), 3, (4, 5), 2, (4, 6), 1, (3, 6), 3, (4, 6), 1, (3, 6), 3, (4, 6), 1, (3, 6), 3, (4, 6), 3, (5, 6), 3, (6, 6), 1, (5, 6), 4, (5, 5), 2, (5, 6), 1, (4, 6), 3, (5, 6), 3, (6, 6), 1, (5, 6), 3, (6, 6), 2, (6, 7), 4, (6, 6), 1, (5, 6), 2, (5, 7), 4, (5, 6), 1, (4, 6), 4, (4, 5), 4, (4, 4), 1, (3, 4), 3, (4, 4), 4, (4, 3), 2, (4, 4), 2, (4, 5), 2, (4, 6), 4, (4, 5), 4, (4, 4), 3, (5, 4), 1, (4, 4), 4, (4, 3), 2, (4, 4), 3, (5, 4), 1, (4, 4), 4, (4, 3), 2, (4, 4), 2, (4, 5), 3, (5, 5), 1, (4, 5), 1, (3, 5), 4, (3, 4), 2, (3, 5), 3, (4, 5), 3, (5, 5), 1, (4, 5), 1, (3, 5), 2, (3, 6), 4, (3, 5), 4, (3, 4), 2, (3, 5), 2, (3, 6), 4, (3, 5), 4, (3, 4), 2, (3, 5), 2, (3, 6), 4, (3, 5), 3, (4, 5), 4, (4, 4), 3, (5, 4), 1, (4, 4), 4, (4, 3), 2, (4, 4), 4, (4, 3), 2, (4, 4), 2, (4, 5), 4, (4, 4), 4, (4, 3), 2, (4, 4), 4, (4, 3), 2, (4, 4), 2, (4, 5), 1, (3, 5), 2, (3, 6), 4, (3, 5), 1, (2, 5), 1, (1, 5), 2, (1, 6), 4, (1, 5), 2, (1, 6), 4, (1, 5), 4, (1, 4), 2, (1, 5), 3, (2, 5), 3, (3, 5), 2, (3, 6), 4, (3, 5), 3, (4, 5), 3, (5, 5), 1, (4, 5), 2, (4, 6), 1, (3, 6), 3, (4, 6), 1, (3, 6), 3, (4, 6), 2, (4, 7), 3, (5, 7), 2, (5, 8), 4, (5, 7), 2, (5, 8), 3, (6, 8), 1, (5, 8), 1, (4, 8), 3, (5, 8), 4, (5, 7), 4, (5, 6), 3, (6, 6), 1, (5, 6), 4, (5, 5), 2, (5, 6), 4, (5, 5), 2, (5, 6), 1, (4, 6), 1, (3, 6), 3, (4, 6), 3, (5, 6), 2, (5, 7), 3, (6, 7), 1, (5, 7), 1, (4, 7), 3, (5, 7), 3, (6, 7), 1, (5, 7), 3, (6, 7), 1, (5, 7), 3, (6, 7), 1, (5, 7), 4, (5, 6), 2, (5, 7), 1, (4, 7), 1, (3, 7), 3, (4, 7), 1, (3, 7), 3, (4, 7), 1, (3, 7), 3, (4, 7), 1, (3, 7), 3, (4, 7), 1, (3, 7), 3, (4, 7), 4, (4, 6), 3, (5, 6), 4, (5, 5), 2, (5, 6), 3, (6, 6), 3, (7, 6), 1, (6, 6), 3, (7, 6), 4, (7, 5), 4, (7, 4), 2, (7, 5), 2, (7, 6), 3, (8, 6), 1, (7, 6), 2, (7, 7), 4, (7, 6), 2, (7, 7), 4, (7, 6), 1, (6, 6), 2, (6, 7), 4, (6, 6), 4, (6, 5), 2, (6, 6), 2, (6, 7), 4, (6, 6), 1, (5, 6), 1, (4, 6), 4, (4, 5), 2, (4, 6), 1, (3, 6), 3, (4, 6), 3, (5, 6), 1, (4, 6), 3, (5, 6), 4, (5, 5), 2, (5, 6), 4, (5, 5), 2, (5, 6), 1, (4, 6), 4, (4, 5), 4, (4, 4), 4, (4, 3), 2, (4, 4), 2, (4, 5), 2, (4, 6), 3, (5, 6), 1, (4, 6), 4, (4, 5), 4, (4, 4), 4, (4, 3), 2, (4, 4), 1, (3, 4), 3, (4, 4), 2, (4, 5), 4, (4, 4), 4, (4, 3), 2, (4, 4), 2, (4, 5), 3, (5, 5), 1, (4, 5), 1, (3, 5), 4, (3, 4), 2, (3, 5), 1, (2, 5), 3, (3, 5), 4, (3, 4), 2, (3, 5), 3, (4, 5), 4, (4, 4), 3, (5, 4), 1, (4, 4), 3, (5, 4), 1, (4, 4), 1, (3, 4), 3, (4, 4), 1, (3, 4), 3, (4, 4), 4, (4, 3), 2, (4, 4), 1, (3, 4), 3, (4, 4), 2, (4, 5), 4, (4, 4), 2, (4, 5), 4, (4, 4), 1, (3, 4), 3, (4, 4), 1, (3, 4), 3, (4, 4), 3, (5, 4), 1, (4, 4), 1, (3, 4), 3, (4, 4), 1, (3, 4), 3, (4, 4), 2, (4, 5), 1, (3, 5), 4, (3, 4), 2, (3, 5), 3, (4, 5), 1, (3, 5), 1, (2, 5), 1, (1, 5), 3, (2, 5), 1, (1, 5), 1, (0, 5), 3, (1, 5), 2, (1, 6), 4, (1, 5), 2, (1, 6), 4, (1, 5), 2, (1, 6), 4, (1, 5), 4, (1, 4), 4, (1, 3), 1, (0, 3), 3, (1, 3), 4, (1, 2), 3, (2, 2), 3, (3, 2), 1, (2, 2), 3, (3, 2), 1, (2, 2), 2, (2, 3), 4, (2, 2), 4, (2, 1), 3, (3, 1), 2, (3, 2), 4, (3, 1), 3, (4, 1), 4, (4, 0), 2, (4, 1), 1, (3, 1), 2, (3, 2), 4, (3, 1), 4, (3, 0), 2, (3, 1), 4, (3, 0), 2, (3, 1), 4, (3, 0), 2, (3, 1), 3, (4, 1), 2, (4, 2)]] :

Output2:

```
**********
*?----*??*
*--*--???*
*-*?*-**?*
*-f*-----*
*???**---*
*????--***
*??*-----*
*?????*--*
```

Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

the number of step to find shortest path :
18
the short path is :
[(8, 8), 4, (8, 7), 1, (7, 7), 4, (7, 6), 1, (6, 6), 1, (5, 6), 1, (4, 6), 4, (4, 5), 1, (3, 5), 1, (2, 5), 4, (2, 4), 1, (1, 4), 4, (1, 3), 4, (1, 2), 3, (2, 2), 4, (2, 1), 3, (3, 1), 3, (4, 1), 2, (4, 2)]**the number of step to find shortest path :**
18
the short path is :
[(8, 8), 4, (8, 7), 1, (7, 7), 4, (7, 6), 1, (6, 6), 1, (5, 6), 1, (4, 6), 4, (4, 5), 1, (3, 5), 1, (2, 5), 4, (2, 4), 1, (1, 4), 4, (1, 3), 4, (1, 2), 3, (2, 2), 4, (2, 1), 3, (3, 1), 3, (4, 1), 2, (4, 2)]
****
----*
----*-*
*-***-****
-f-----*
***-**---*
----***
-------*
----*-a**
****

- Multiple return capability According to the project:

```
def main():                                                    #multi return
    # return get_start_or_end_location('a'),get_start_or_end_location('f'),count_number_of_Step(actions),find_food,count_short[0],combine  ->this line is completer return
    return get_start_or_end_location('a'),get_start_or_end_location('f'),count_short[0],combine

# write into file that we suppose an agent have memory(امتیازی بخش)
write_out_put_Map_two_list(update_out_put_map[0])

if __name__ == "__main__":
    print(main())
```

- With multi-return capability, the output of Input samples 1 and 2 are :
- Out_put_sample_1:
- ((2, 2), (7, 7), 12, [(2, 2), 1, (1, 2), 2, (1, 3), 2, (1, 4), 2, (1, 5), 3, (2, 5), 3, (3, 5), 3, (4, 5), 2, (4, 6), 3, (5, 6), 3, (6, 6), 3, (7, 6), 2, (7, 7)])

Report Of first Project

AI course(packmnan_game_crafting)

(Ehsan_Khademi_40028733)

Under the supervision of the Dr.Derhami

- Out_put_sample_2:
- ((8, 8), (4, 2), 18, [(8, 8), 4, (8, 7), 1, (7, 7), 4, (7, 6), 1, (6, 6), 1, (5, 6), 1, (4, 6), 4, (4, 5), 1, (3, 5), 1, (2, 5), 4, (2, 4), 1, (1, 4), 4, (1, 3), 4, (1, 2), 3, (2, 2), 4, (2, 1), 3, (3, 1), 3, (4, 1), 2, (4, 2)])

- first point: considering that the output test is software, I added the mine function
- second point:
- The second point is the output in the form of a list including:
- The first element of the operator coordinate list as a tuple
- The second element of the coordinates of food in the form of a tuple
- The third element is the number of steps as the number
- the fourth element of the traveled path list