1. a1-forrelease

2.

CSCI B551

2.1. _

Assignment-a1 : Searching

Submitted by Durga Sai Sailesh Chodabattula, Manideep Varma Penumatsa, Sai Jagan Lakku

Submitted on 8th October

2.2. **1.1. Part 1 : The 2021 Puzzle**

- Q1) In this problem, what is the branching factor of the search tree?

2.2.0.1. Ans: There are 24 possible successor moves for each move, so the branching factor of the search tree is 24.

- Q2) If the solution can be reached in 7 moves, about how many states would we need to explore before we found it if we used BFS instead of A* search? A rough answer is fine.

2.2.0.2. Ans: If we use BFS instead of A*, we must search a maximum of 24*7 states.

Initial state: The initial board.

2.2.1. Goal state:

```
1  2  3  4   5

6  7  8  9  10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 25
```

2.2.2. Successors:

Each state has a total of 24 successors. Sliding the rows L(left) and R(right) a total of 10 successors, sliding the columns U(up) and D(down) a total of 10 successors, rotating 0c(outer ring clockwise) and 0cc(anticlockwise) a total of 2 successors, and rotating Ic(rotating inner ring clockwise) & Icc(anticlockwise) a total of 2 successors are all used to create successors.

2.2.3. Cost function:

The cost function is the total number of moves required to attain the desired state. The cost function is one for each move.

2.2.4. Heuristic function:

Sum of Manhattan distance divided by 5

2.2.4.1. Ans:

- As a heuristic function, we used the sum of Manhattan distance divided by 5.
- Because five items change positions along the row or column with each move.

2.2.5. Initial State:

The initial state here is a sliding puzzle with misplaced tiles.

2.3. Part 2 : Road trip!

The initial step was to read and parse both the city-gps and road-segment information.

2.4. Goal State:

The goal was to identify the shortest path between two points .

2.5. Cost Function:

Distance, time, delivery time, and segments make up the cost function.

2.6. SuccessorFunction:

-We utilized heapq to identify the best route between the two locations.

- A key will be assigned to each item in the queue, as well as the full path taken up until that node.
- For each nodes, all successors were tested for the specified cost function, except for the one that was already on the path.

Part 3 : Choosing teams

3.1. Cost Fucntion:

Varying Cost fuction based on the criteria given in the question.

```
def costFn(input, teamList):

        cost = 0

        #teams
        for i in range(0,len(teamList)):cost += 5

        #unwanted cases
        for l in input:
            check = l.split(" ")[0]
            not_list = l.split(" ")[-1].split(",")
            for t in teamList:
                t = t.split("-")
                if check in t:
                    for user in not_list:
                        if user in t:cost += 10
                    break

        #redunduncy check condition
        for l in input:
            check = l.split(" ")[0]
            yes_list = l.split(" ")[1].split("-")
            for t in teamList:
                t = t.split("-")
                if check in t:
                    teamChk = t
                    break
            for user in yes_list:
                if user != 'xxx':
                    if user not in teamChk:cost += 3


        #Wrong group assignment
        for l in input:
            check = l.split(" ")[0]
            for t in teamList:
```

```
                t = t.split("-")
                if check in t:
                    teamChk = t
                    if len(teamChk) != len(l.split(" ")[1].split("-")):
                        cost += 2
                        break
        return cost
```

3.1.1. Goal State:

- The optimal team combinations based on the survey input file.
- The program yeilds combinations until the optimal team combination is achieved
- Below is the user genaration list function

```
def genFn(userList):
        team = []
        while userList:
            teamSize = min(len(userList),random.randint(1,3))
            pop_list = random.sample(userList, teamSize)
            team.append("-".join(pop_list))
            userList = list(set(userList)-set(pop_list))
        return team
```