Agents and Search

In this homework, you are required to answer questions Q1 and Q2, and implement a solution to a described search problem (Q3). To do so, you are going to implement some of the search algorithms that have been covered in the class. Example mazes for problem in Q3 will be provided to you, and you are required to print an output with correct answer and correct format. Details about the problem and input/output formats are given in the next sections.

Problem 1 - PEAS Description of Agents (15 points)

For each of the following agents, develop a PEAS description of the task environment:

(a) A drone that carries a packet from one location to another

(b) A robot that sings songs until baby sleeps

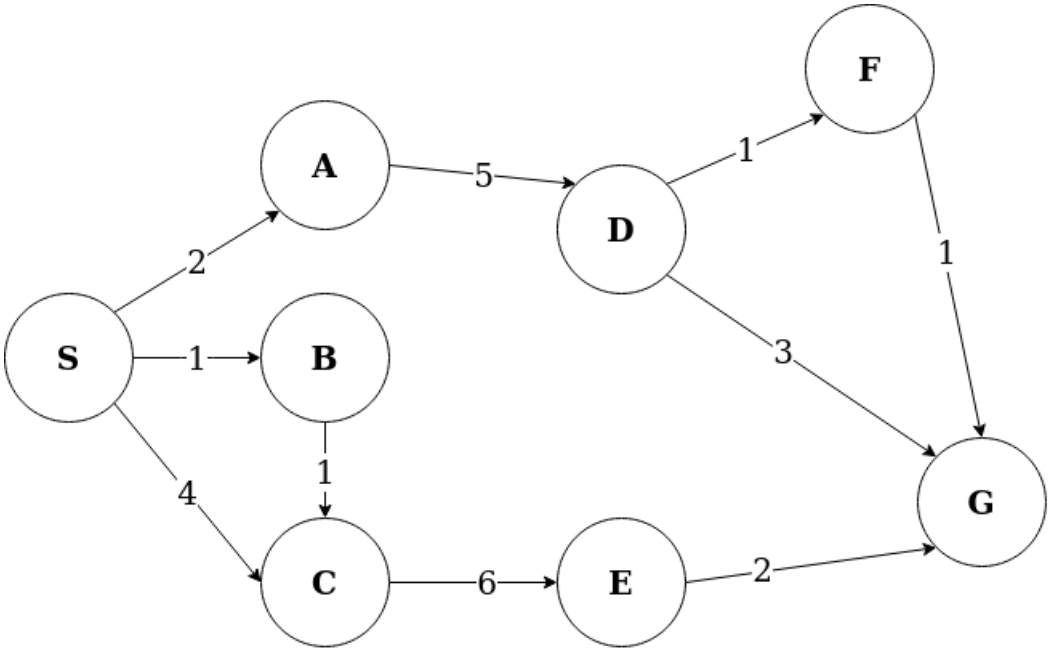
(c) Activity recognition and anomaly detection software agent in an airport

(d) An agent that classifies emails as spam or not

For each of these agent types characterize the environment according to the properties of the environment (observability, dynamism, etc.), and determine the appropriate type of the agent architecture with reasonable arguments.

Problem 2 - Admissible but Inconsistent Heuristics (25 points)

In the graph below, S is the start node and G is the goal node. There is also a heuristic function h whose values for each state is shown in the table, except h(B). Values along edges are costs of moving from one node to another.



n	h(n)
S	6
A	5
B	?
C	5
D	2
E	1
F	1
G	0

Figure 1

(**a**) What values of h(B) make h an admissible heuristic function?

(**b**) What values of h(B) make h a consistent heuristic function?

(**c**) What values of h(B) make it so that when you perform graph version of A* on this graph, you first expand node S, then node A, then node B in order.

Problem 3 - Problem Solving with Search Algorithms (60 points)

In this section, you are required to implement a solution to the given problem using various search algorithms. In this problem, you control some agents whose aim is to reach their goal position **in minimum amount of time steps** in a grid world environment. Figure 2 illustrates visualization of two example test cases:

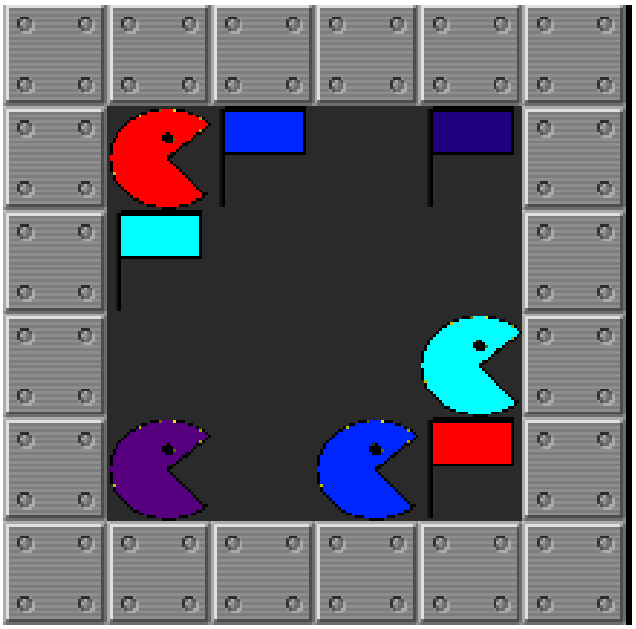


Figure 2: Visualization of an example test case for the described problem.

The game has the following rules:

- At each time step, the agents can move in 4 directions: right, up, left, down. They also can pass their turn to stay at the same position. They can not walk into walls.
- An agent can only move from 1 grid rectangle to another at a single time step. There is not a move such as "go right until hitting a wall".
- An agent may continue to move after it reached its goal position.
- More than one agent can not be at the same position at a time step, and agents can not move through each other while moving. See below section for examples.
- A test case is completed/solved when all agents at their respective goal position at a time step.

You need to:

(a) Formulate this problem in a well-defined form. Explain your state and action representations in detail. If you have used different representations in your implementation, explain them all.

(b) Run appropriate versions of Breadth-First Search (BFS) and Depth-First Search (DFS). Analyze the results in terms of:

- Number of nodes generated
- Number of nodes expanded
- Maximum number of nodes kept in the memory
- Running time

If any of the algorithms does not last, please specify the reason.

(c) Run A* Algorithm with an admissible and consistent heuristic function. Experiment with at least two different tie breaking (deciding which node to expand when f values of the nodes are same) approach, and give a detailed analysis of the results in your report in terms of:

- Number of nodes generated
- Number of nodes expanded
- Maximum number of nodes kept in the memory
- Running time

Some important notes about your solution:

- Multiple mazes (maze 0 to maze 4) are given to you so that you can test correctness of your implementation, but you do not need to perform asked analyses on all mazes. Choose one of the mazes to compare all algorithms (BFS, DFS, A*) and perform A* analysis on maze 3 only.
- Your program **must** accept two command line arguments. First argument is input file path to read initial game state and second is output file name. If second argument is not specified, use default name "output.txt".
- Maze 3 is given to solve with A* only, you do not need to solve it with the uninformed search algorithms. Also maze 4 is given if you want to keep working on the problem, but it is not required to solve it for completing the assignment. For the other mazes, if any of the algorithms can not find the result, please specify the reason.

Input File Format

First line contains 3 tab separated integers: Number of rows in the grid (R), number of columns in the grid (C), and number of agents (N). Remaining R lines contain information about what is in the each grid cell. Each of these lines contain C tab separated strings:

- W represents wall
- AX represents agent with number X (first agent index is 1)
- GX represents goal position with number X (first goal index is 1)
- E represents empty cell.

Output Format

First line of the output contains two integer: Number of time steps to reach final state (M) and number of agents (N). Remaining M lines contains move of each agent in time steps. Each move is represented by a single letter:

- R - move right
- U - move up
- L - move left
- D - move down
- P - pass

Constraints

$3 \leq$ R ≤ 20
$3 \leq$ C ≤ 20
$1 \leq$ #agents ≤ 9

Borders of each maze is enclosed by walls, so you do not need to perform additional bounds checking.

Example Input Outputs

Example input 0:

7 8 2
W W W W W W W W
W E E E W E E W
W G2 W E E E E W
W W W A1 W W W W
W E W E E E G1 W
W E E E W A2 E W
W W W W W W W W

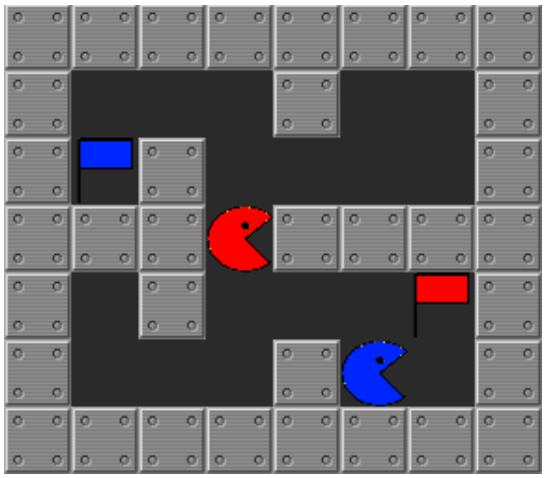


Figure 3: Visualization of maze from example input 0.

Example output 0:
9 2
DU
PL
DL
UU
RU
RU
RL
DL
UD

Explanation of example 0:
In input file, we see 7 rows, 8 columns and 2 agents. In output file, there are 9 moves and 2 agents. Letters in the left are moves of agent 1 and letters in the right are moves of agent 2. If these moves are performed sequentially given problem is solved in 9 steps.

Example Legal and Illegal Moves

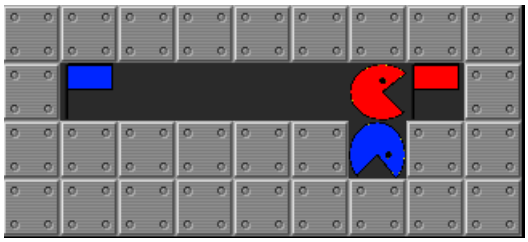


Figure 4: If red agent moves to left or right, blue agent can move to up in the same time step.

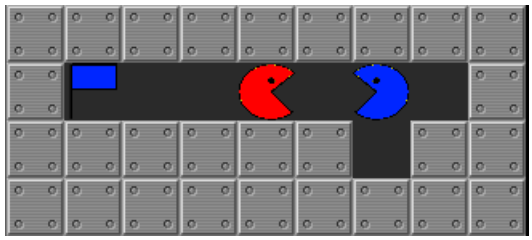


Figure 5: If red agent (agent 1) moves to the right, blue agent (agent 2) can not move to the left and vice versa. So "RL" is an invalid move combination for this time step.

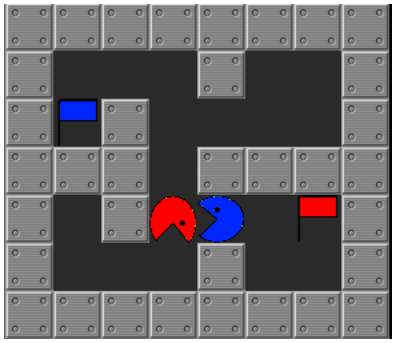


Figure 6: In same time step, red agent can not move to right and blue agent can not move to left because they have to walk through each other.

Submission

Submit your homework files through Ninova. Please zip and upload all your files using filename BLG435E_HW_1_STUDENTID.zip. You are going to submit:

1. All your code files for implementation of Q3.
2. A pdf file containing answers of the first two questions of the homework, required explanations/analyses about the third question of the homework (also specify which mazes you performed the analyzes) and instructions to compile/run your code.

There is no restriction on the programming language.

Your code must be able to compile and run on ITU's Linux Servers without any errors, otherwise your code may not be evaluated (If you can not run your program on ITU servers due to some requirement for your programming language, like JDK for java, just make sure in can run on a Linux machine and explain the situation in your report).

Pay attention to deadline of the homework, including hour.

Important: In Q3, your solution can rely on existing BFS, DFS or A* algorithms. However, you need to explain how the algorithms work in this problem and perform the requested analyses above with sufficient explanations in your report. **Code usage without relevant references will be considered as plagiarism.**

Note that submitted homeworks may also be tested with different initial grids other than those that are given to you.

In case of any questions, feel free to send an e-mail to unlut@itu.edu.tr.