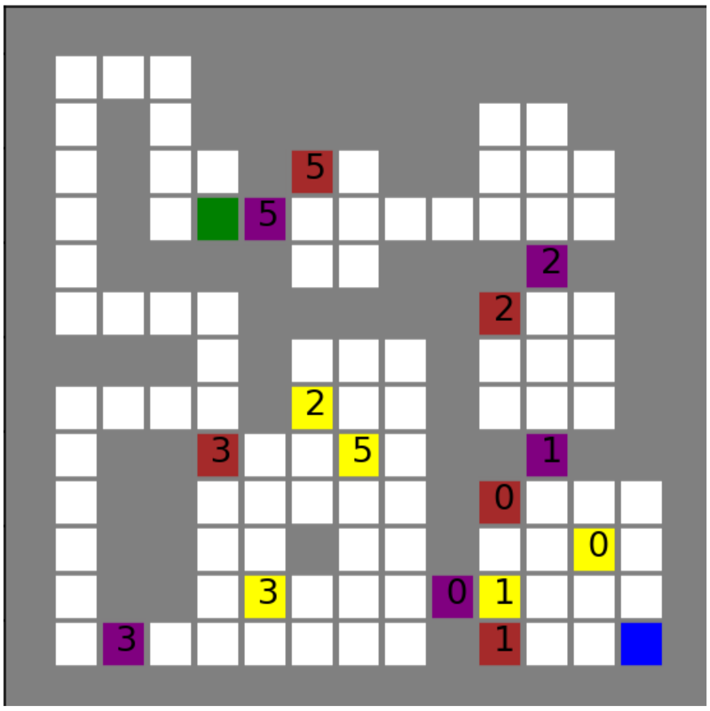
Home assignment 1: Pathfinding

Introduction

The point of the exercise is to experiment with solving a problem using some of the pathfinding algorithms we studied in the course.



The problem you will tackle is like the Japanese "sokoban" game.
The goal is to move key blocks (yellow) to their pressure plates (brown) to open their locked doors (purple) to have the agent (blue) a clear **orthogonally** path to the goal (green).

Problem:

Input: initial setup of the game matrix of size NxM represented in a tuple of tuples.

Possible actions: The agent may move orthogonally and may only push a key block if the path is clear (the agent may not push multiple blocks at once).
The agent **may not** move through pressure plates or key blocks; once key blocks are set onto their pressure plate they behave like a wall.
Once a pressure plate has been set by their key block they open their locked door, and the locked door becomes an empty space.

Success: The agent finds a path to the goal.
Failure: No possible path to the goal can be found or dead-end.

Assignment:

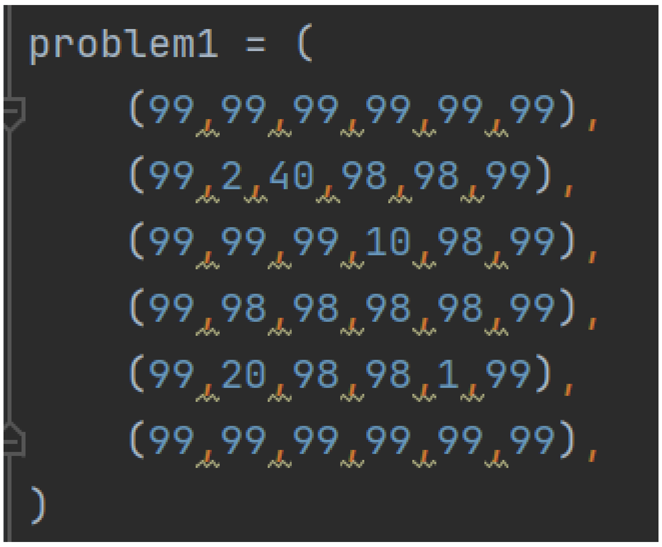
For this assignment we've attached code for you to use. Most of what's needed is implemented. What you will have to do is implement the PressurePlate class functions so the checker code can be utilized to solve the problem.
ex1.py file has the following functions for you to implement (you could also have helper functions as you see fit):
1. Successor function – receives a state of the game and returns a list of tuples (each tuple is a pair of action and following state achieved from that action)
2. Goal_test function – receives a state of the game and returns a Boolean if the state has achieved the goal requirement.
3. H function – the heuristic function receives a node which holds the state as a property; you must develop a heuristic function that will be able to estimate effectively the cost to the goal. Please note that the quality and speed of the heuristic function will affect the performance of the pathfinding algorithms.

DO NOT CHANGE THE FUNCTIONS NAMES OR INPUT!

Input:

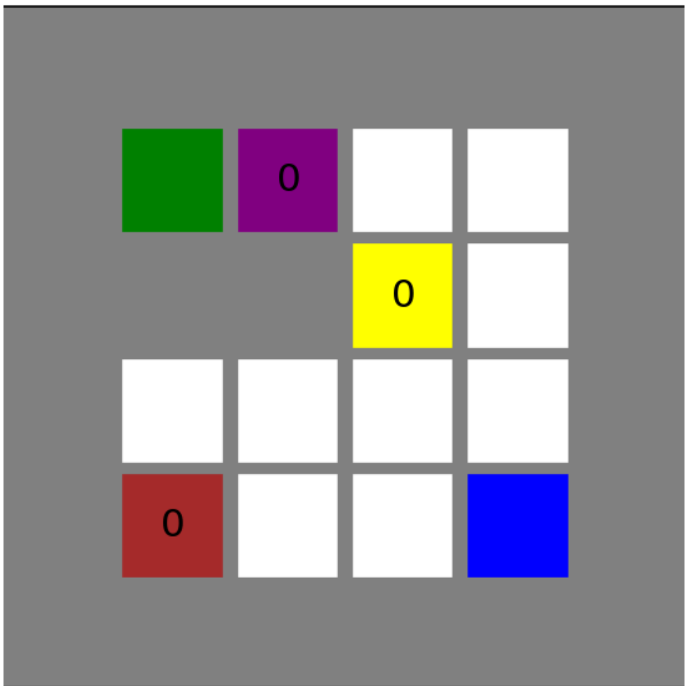
In the ex1_check.py there will be the solve_problems function that will receive the problem matrix and which algorithm to use (either "gbfs" or "astar").
The problem matrix will be NxM tuple of tuples that each value will be an int that can be the following values:
BLANK = 0
WALL = 99
FLOOR = 98
AGENT = 1
GOAL = 2
AGENT_ON_GOAL = 3
LOCKED_DOORS = 40 ... 49
PRESSED_PLATES = 30 ... 39
PRESSURE_PLATES = 20 ... 29
KEY_BLOCKS = 10 ... 19

Example input:



```
problem1 = (
    (99,99,99,99,99,99),
    (99,2,40,98,98,99),
    (99,99,99,10,98,99),
    (99,98,98,98,98,99),
    (99,20,98,98,1,99),
    (99,99,99,99,99,99),
)
```

This input represents the following game:



Don’t base the x and y axis on the image, base it on the input!

Action representation:

The agent has 4 possible actions:

- Moving right – represented by “R” string
- Moving left - represented by “L” string
- Moving up - represented by “U” string
- Moving down - represented by “D” string

Every action makes the agent move towards the desired direction if its applicable (either a floor tile or keyblock or an unlock door or the goal), otherwise it’s an invalid move and therefore won’t be checked.

Notes:

- It is possible to create a dead-end state from multiple different scenarios. You must check for this and make sure to avoid it or set them in the correct priority order.
- Every locked door will have a type (e.g. 47 is of type 7).
- Once all the pressure plates of one type are set/pressed they open all of the locked doors of their type (all 27s must become 37s to open all of the 47s).
- Only key blocks of the same type can set/press pressure plates of the same type (17 can only press 27 to make 37). If there is only one 17 block but two 27 plates the 47 locked doors cannot be opened.
- In the successor function and the __init__ you can represent the state however you want for your needs, but note that it must be hashable, so lists or dictionaries or even numpy arrays (but there are ways to transfer to tuples or to make them hashable).
- Please be consistent with your representations (actions and states, e.g. capitalized letters, underscores, etc.). The testing is automatic and even if the code is compiled correctly you will lose points for inconsistencies.
- Please make sure that your code works on both Linux and Windows, and please ask permission in the assignment forum to use any non-built-in libraries/packages (any library/package that you have to perform pip install to use it).

Grading:

The assignment will be checked automatically, and it is important that the correct names of the files, classes and functions are kept.
The grading program will consist of 20 different "PressurePlate" problems of increasing difficulty.
Your code will need to finish each problem linearly solving GBFS and A*.
If your code gets stuck on an earlier problem the later problems won't be solved and therefore will receive a 0.
You will not receive full marks for getting a non-optimal solution.
You will need to finish all 20 problems, solved both in GBFS and A*, in under 5 minutes to receive a full grade.
Each successful optimal solution for each of the problems will grant 2.5 points (GBFS and A* separately).
You can assume that maximal matrix size is 20x20 and that at most there will be 10 key blocks.

Self-check: The attached file ex1_check.py has two example problems. It will not check the correctness of code or is enough for sufficient testing. It is encouraged to create more example problems to check the robustness of your code.

Attached files:

The files include:

1. Ex1.py – The only file to be submitted and the only one relevant for grading.
2. Ex1_check.py – The file to use for self-checking, any changes here should only be to adding more problems for you to check (the submission won't check your added problems).

3. Utils.py and Search.py – auxiliary files used to have the different algorithms and classes for the assignment.

Submission:

- Deadline – 8.5.25
- Solo submissions.
- Please put your ID in the variable id in ex1.py and along with submitting ex1.py please submit a details.txt that includes your name (in English only) and your ID (One line after the other).
- Any questions please use the Lamda forum for the assignment.
- The submission will use the submit system. Upload only the ex1.py and details.txt. Do not upload any other files or in any other format (like zip or rar).