Assignment 0 (Warm-Up, Variant A): Clean the Wumpus Cave

AI-1 Systems Project (Summer Semester 2024)

Jan Frederik Schaefer

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik

Topic:	Agents in AI, search
Due on:	June 29, 2023
Version from:	April 15, 2024
Author:	Jan Frederik Schaefer
Important notes:	To be solved individually
	Earlier deadline for first results (see your solution repository)
	Do not use someone else's solution code (even as inspiration)
	Ask for help if you are stuck (office hours, assignment room, . . .)
	Every assignment has guide with tips – you can find it at [AG]

1   Task summary

A recurring theme in the AI lecture is the Wumpus world. The Wumpus is a mystical creature that lives in a cave that is organized as a grid of squares. We want to clean the Wumpus cave using a vacuum cleaner robot, which we can control with a sequence of instructions. You have to implement two tasks:

1. Check if a sequence of instructions cleans the entire Wumpus cave.
2. Come up with a sequence of instructions yourself – the shorter, the better.

The assignment repository [AR] contains files with problem representations that you have to solve. Your grade will largely be based on those solutions (see Section 6). The assignment repository also contains example solutions that you can use to test your implementation.

Didactic objectives

1. Develop an algorithm to solve a non-trivial problem,
2. implement a small software project from scratch,
3. get hands-on experience with a search problem,
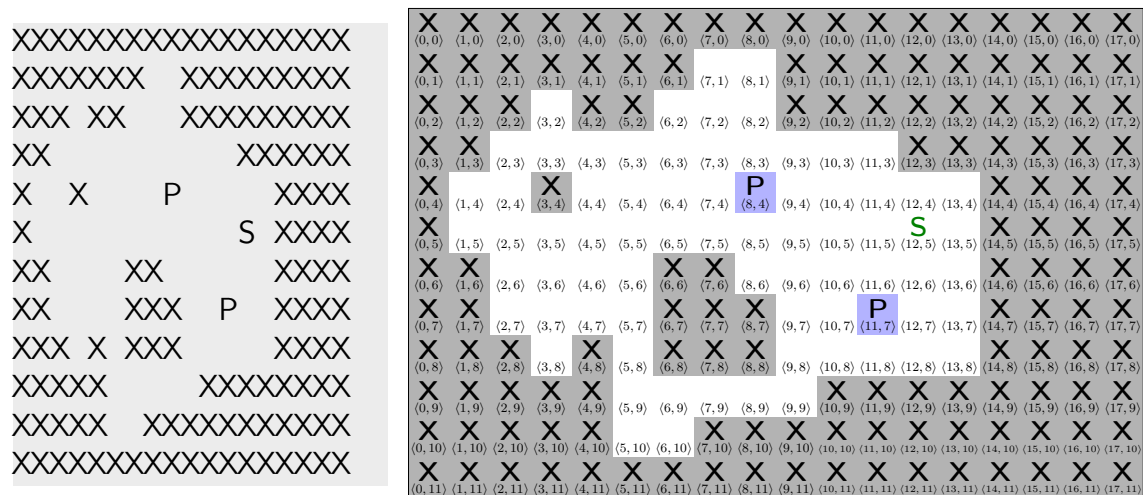4. improve the efficiency of an algorithm,

Figure 1: An example map of the Wumpus cave with the starting position marked at $\langle 12, 5 \rangle$ and two portals marked with Ps. The text representation (left) is used in the problem files.

    5. get to know the AISysProj setup and workflows.

Prerequisites and useful methods

1. The basics of computer science and programming,
2. Search (in a very general sense).

2   Maps

You have a **map** of the Wumpus cave, which consists of 18×12 squares. Figure 1 shows an example map. Every square has coordinates associated with it. As is common in computer science, the y-axis points down and the origin, $\langle 0, 0 \rangle$, is in the top-left square.

    The properties of each square are represented by a single character:

1. Walls are marked with an X.
2. Empty squares are marked with a space.
3. The starting position of the vacuum cleaner (if it is known) is marked with an S.
4. Portals (if they exist) are marked with P.

    The maps are stored in the problem files (see Section 4) using a text representation: each row of the map corresponds to a line in the text representation and each square to a character. Figure 1 shows an example map with both the text representation and a more visual representation.

3   Plans and potentially missed squares

You can control the vacuum cleaner by making a **plan**, which is a sequence of instructions. The following instructions are available:

1. **N**: The vacuum cleaner moves one square *north* (up).
2. **E**: The vacuum cleaner moves one square *east* (right).
3. **S**: The vacuum cleaner moves one square *south* (down).
4. **W**: The vacuum cleaner moves one square *west* (left).

We assume that the vacuum cleaner cleans every square it is on. If the instruction would move the vacuum cleaner onto a wall, it will instead remain on its current square. If the vacuum cleaner gets moved onto a portal square, it will immediately get teleported to the other portal square (there should always be exactly 0 or 2 portals). Note that portals do not have to get cleaned by the vacuum cleaner.

A good plan for the vacuum cleaner makes sure that everything gets cleaned. More concretely, we are interested in the **potentially missed squares**, which might still be dirty after executing the cleaning plan. To illustrate this, we will take a look at a few examples.

Example: Simple cave   In this example, we will explore what happens if we execute the plan EENS in the cave shown in Figure 2a. We start in $\langle 2, 2 \rangle$. After the instruction E, we will be in $\langle 3, 2 \rangle$. Now there is a wall to the east, which means that we will stay in $\langle 3, 2 \rangle$ when we try to go east again. Next, we go north ($\langle 3, 1 \rangle$) and then south ($\langle 3, 2 \rangle$ again). Therefore, we will not have cleaned $\langle 1, 1 \rangle$, $\langle 2, 1 \rangle$ and $\langle 1, 2 \rangle$.

Example: Portals   In this example, we will explore what happens if we execute the plan EENSS in the cave shown in Figure 2b. We start at $\langle 1, 2 \rangle$. Going east, we land on the portal at $\langle 2, 2 \rangle$, which immediately teleports us to $\langle 3, 3 \rangle$. Now there is a wall in the east, which means that we will stay at $\langle 3, 3 \rangle$ for the next action. Then we go north ($\langle 3, 2 \rangle$) and south again, which means we will teleport from $\langle 3, 3 \rangle$ to $\langle 2, 2 \rangle$. Going south again, we land on $\langle 2, 3 \rangle$. Therefore, we will clean $\langle 1, 2 \rangle$, $\langle 2, 3 \rangle$ and $\langle 3, 2 \rangle$. We will miss the squares $\langle 1, 3 \rangle$ and $\langle 3, 1 \rangle$ (the portals do not have to be cleaned).

Example: Portals and unknown initial position   In this example, we will explore what happens if we execute the plan NS in the cave shown in Figure 2c. If we start at $\langle 2, 3 \rangle$, we will miss $\langle 3, 3 \rangle$. If we start at $\langle 3, 2 \rangle$, we will also miss $\langle 3, 3 \rangle$. But if we start at $\langle 3, 3 \rangle$, we will miss $\langle 2, 3 \rangle$. Therefore, the potentially missed squares are $\langle 2, 3 \rangle$ and $\langle 3, 3 \rangle$.
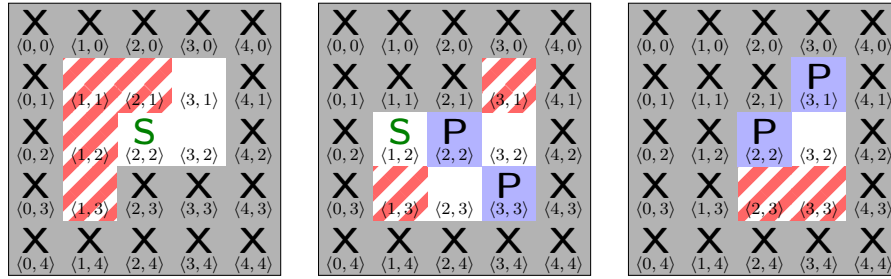
(a) plan EENS   (b) plan EENSS   (c) plan NS

Figure 2: Example caves. The potentially missed squares are marked with diagonal lines (▨).

4 Problem and solution files

The assignment repository [AR] contains many **problem files**. Your implementation is supposed to generate a **solution file** for each problem file. This section describes the format of problem and solution files.

4.1 Checking plans

The easier problem files require you to check a cleaning plan. They begin with the line CHECK PLAN, followed by a plan as described in Section 3, followed by the text representation of a map as described in Section 2.

If there are no potentially missed squares, the solution file should contain the text GOOD PLAN. Otherwise, the solution file should contain the text BAD PLAN, followed by a list of the potentially missed squares (the order does not matter). For example, if the squares $\langle 2, 3 \rangle$ and $\langle 1, 5 \rangle$ are potentially missed, the solution file should be

```
BAD PLAN
2, 3
1, 5
```

4.2 Finding plans

The more difficult problem files require you to find a cleaning plan. They begin with the line FIND PLAN, followed by the text representation of a map. The solution file should then

contain the plan as described in Section 3.

If the format is not clear, you can take a look at the assignment repository [AR], which contains example problems and solutions.

> **Important:** The number of points for plan finding problems depends on the plan lengths (see Section 6 for details).

5 What to submit

Your solution should be pushed to your git repository for this assignment. For this warm-up assignment, we have an early deadline, which should be stated in the README of your repository. At this deadline, the repository should contain:

1. all the code you have so far,
2. solutions to the problem_a_*.txt files.

Otherwise, we might assume that you are not actually interested in the project and give your spot to someone else.

Your grade will be based on your final submission (deadline: June 29, 2023). Concretely, your repository should contain:

1. all your code for solving this assignment,
2. a README.md file explaining
    i. dependencies how to run your code (programming language, version, external libraries and how to get them),
    ii. the repository structure,
    iii. how to run your code to solve other problems,
    iv. anything else we should know,
3. a solution summary (see [SoS] for more details – it should describe the main ideas, not document the code),
4. solution files (as described in Section 4) for the problem files. The solution file for problem_X_YZ.txt should be called solution_X_YZ.txt.

6 Points

The total number of points for this assignment is 100.

Up to 80 points are awarded for the solutions to the problem files. Figure 3 shows how many points can be achieved for each part. For the FIND PLAN problems, the number of

points depends on the total plan length T of your solutions, i.e. T is the sum of the lengths of the plans you found for that part.

Note that partial points (for solving only part of a problem range correctly) are only awarded in exceptional cases.

The remaining 20 points are awarded for the submission quality. The points are primarily awarded for the solution summary (see [SoS]), but it also includes the README (instructions on how we can run your code) and the overall organization of your repository (can we find the files? are they in the correct format? etc.) Note that we do not grade the code quality itself.

If the grading scheme doesn't seem to work well, we might adjust it later on (likely in your favor).

Important: You get points for *correct* solutions. You generally do not e.g. get partial points for code that "looks roughly correct but produces wrong results". The assignment repository contains a script that you can use to check your solutions for the example problems.

References

[AG]   *Guide for "Assignment 0 (Warm-Up, Variant A): Clean the Wumpus Cave".* URL: https://kwarc.info/teaching/AISysProj/SS24/assignment-1.0.A-guide.pdf.

[AR]   *Repository for Assignment 0 (Warm-Up, Variant A): Clean the Wumpus Cave.* URL: https://gitlab.rrze.fau.de/wrv/AISysProj/ss24/a1.0.a-clean-wumpus-cave/assignment.

[SoS]   *Solution Summary.* URL: https://gitlab.rrze.fau.de/wrv/AISysProj/admin/general/-/blob/main/solution-summary.md.

Problems	Mode	Challenges	points	
problem_a_*.txt	check	—	10	
problem_b_*.txt	check	portals	10	
problem_c_*.txt	check	unknown start position portals	10	
problem_d_*.txt	find	—	15	if $T \leq 10000$
			10	if $T \leq 25000$
			5	if $T \leq 40000$
			0	if $T > 40000$
problem_e_*.txt	find	portals	15	if $T \leq 15000$
			10	if $T \leq 25000$
			5	if $T \leq 40000$
			0	if $T > 40000$
problem_f_*.txt	find	unknown start position portals	20	if $T \leq 10000$
			15	if $T \leq 15000$
			10	if $T \leq 25000$
			5	if $T \leq 40000$
			0	if $T > 40000$

Figure 3: Points per part.