Assignment 2

Daniel Fišer (Jan Mrkos)

Department of Computer Science,
Faculty of Electrical Engineering,
mrkosja1@fel.cvut.cz

April 12, 2021

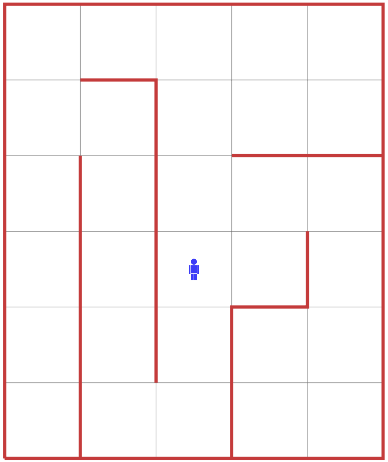
Content:

- Assignment 2 domain
- FF replan solution example
- Requirements
- Grading

Assignment 2 domain

Consider the following task:

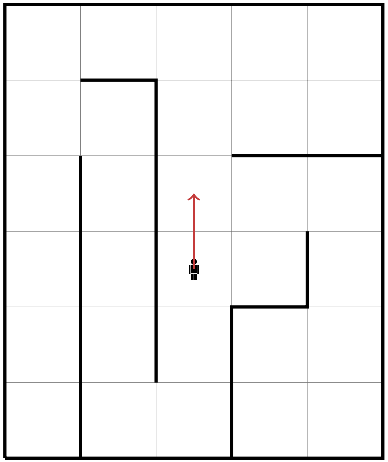
- We have a rectangular maze with walls and an agent.



Assignment 2 domain

Consider the following task:

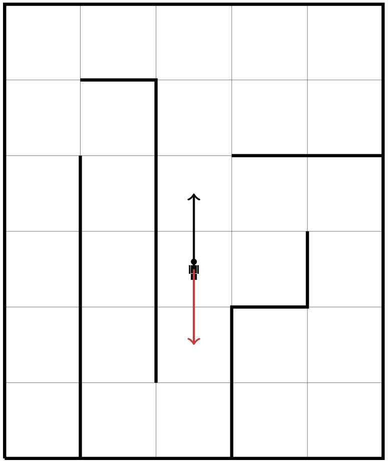
- We have a rectangular maze with walls and an agent.
- The agent can move up,



Assignment 2 domain

Consider the following task:

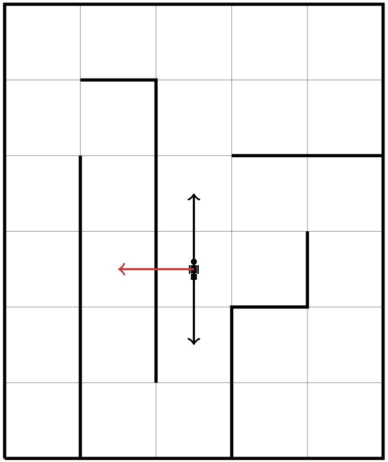
- We have a rectangular maze with walls and an agent.
- The agent can move up, down,



Assignment 2 domain

Consider the following task:

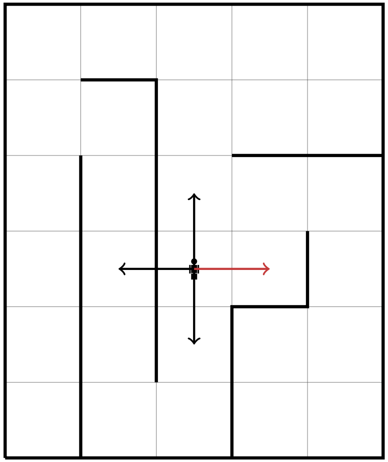
- We have a rectangular maze with walls and an agent.
- The agent can move up, down, left,



Assignment 2 domain

Consider the following task:

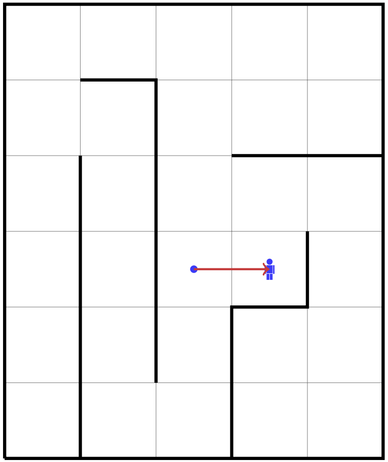
- We have a rectangular maze with walls and an agent.
- The agent can move up, down, left, and right.



Assignment 2 domain

Consider the following task:

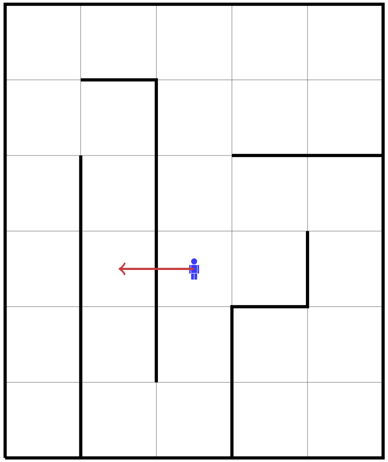
- We have a rectangular maze with walls and an agent.
- The agent can move up, down, left, and right.
- The agent can move if the movement is not blocked by a wall.



Assignment 2 domain

Consider the following task:

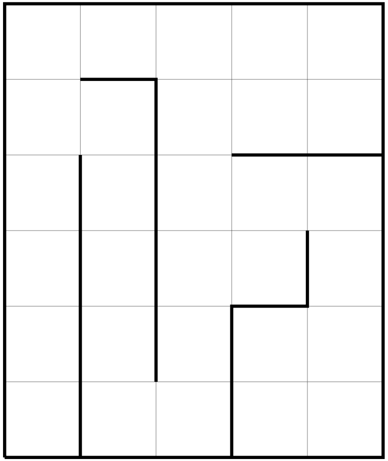
- We have a rectangular maze with walls and an agent.
- The agent can move up, down, left, and right.
- The agent can move if the movement is not blocked by a wall.
- But if the movement is blocked by a wall, the agent stays put.



Assignment 2 domain

Consider the following task:

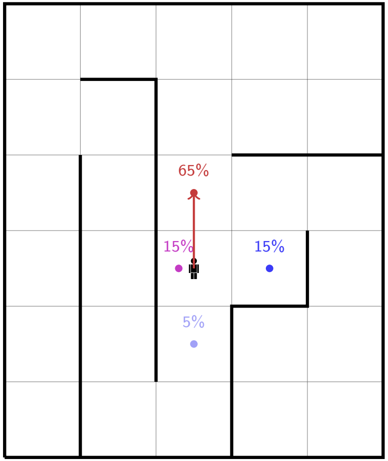
- We have a rectangular maze with walls and an agent.
- The agent can move up, down, left, and right.
- The agent can move if the movement is not blocked by a wall.
- But if the movement is blocked by a wall, the agent stays put.
- There is, however, a catch: Whenever the agent decides to go in one direction, he succeeds with probability of 65%, he will go sideways with probability of 15%, and he will go in the opposite direction with probability of 5%.



Assignment 2 domain

Consider the following task:

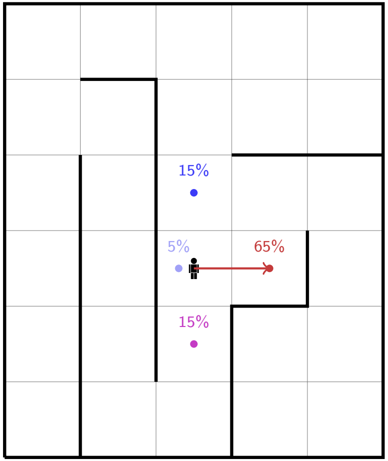
- There is, however, a catch: Whenever the agent decides to go in one direction, he succeeds with probability of 65%, he will go sideways with probability of 15%, and he will go in the opposite direction with probability of 5%.
- So for example, if the agent decides to go up, he ends up here with 65% probability, here with 15% probability, here with 15% probability, and here with 5% probability,



Assignment 2 domain

Consider the following task:

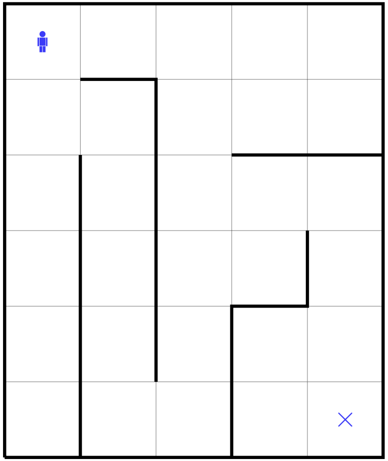
- There is, however, a catch: Whenever the agent decides to go in one direction, he succeeds with probability of 65%, he will go sideways with probability of 15%, and he will go in the opposite direction with probability of 5%.
- And if the agent decides to go right, he ends up here with 65% probability, here with 15% probability, here with 15% probability, and here with 5% probability,



Assignment 2 domain

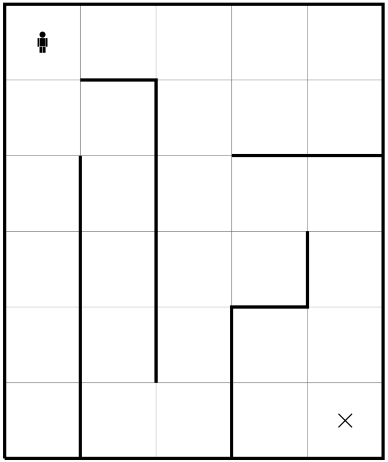
Consider the following task:

- The agent wants to find a way through the maze.
- He starts at the top left corner and wants to reach the bottom right corner.



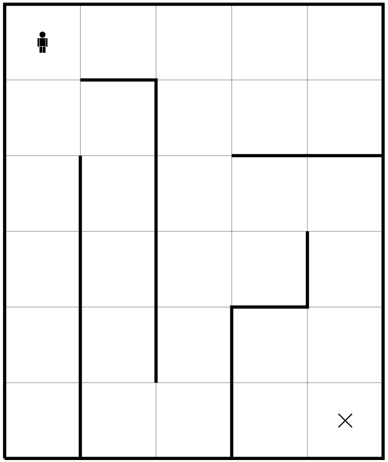
Assignment 2 domain

- Now, stop for a moment and try to figure out how would you solve this problem. How would you navigate the agent through the maze?



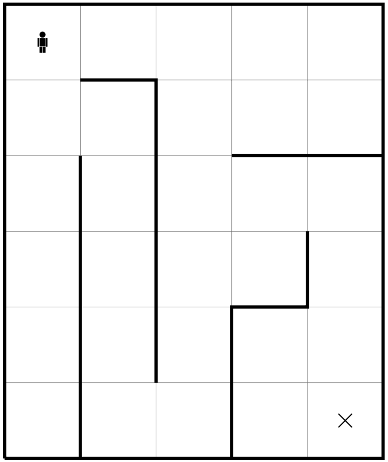
Assignment 2 domain

- Now, stop for a moment and try to figure out how would you solve this problem. How would you navigate the agent through the maze?
- Go back to the lectures on probabilistic planning. Make sure you understand how to model this task as a probabilistic planning problem.



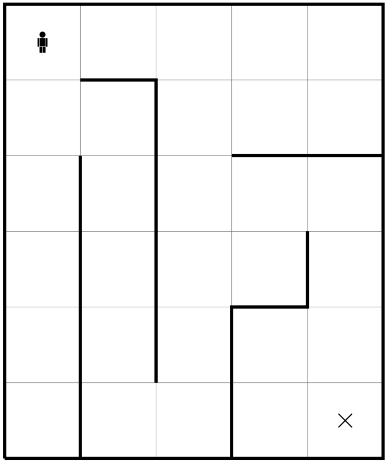
Assignment 2 domain

- Now, stop for a moment and try to figure out how would you solve this problem. How would you navigate the agent through the maze?
- Go back to the lectures on probabilistic planning. Make sure you understand how to model this task as a probabilistic planning problem.
- Each state describes the position of the agent in the maze. And in every state, there are four actions that can be executed, each of which has four outcomes with the prescribed probabilities.



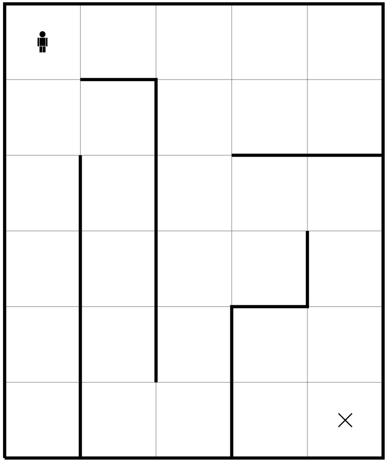
FF-replan example

- You will be allowed to solve the problem however you wish. Here, we will show you one way to solve it, which should also give you a general idea about how FF-Replan works.



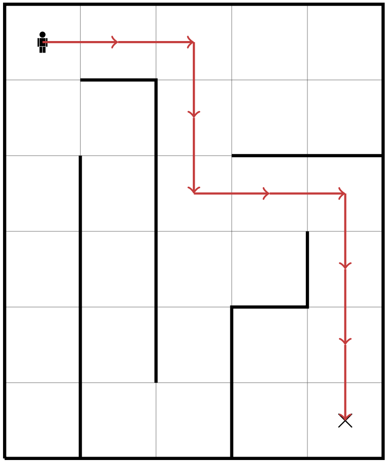
FF-replan example

- The idea is the following:
  1. We deteminize the planning problem. In this case, we will simply assume that actions will not move the agent sideways or backwards, i.e., the action up will always move the agent up, down will always move the agent down, and so on. (Note that this corresponds to the "Most-likely-outcome determinization".)
  2. We plan the path (e.g., using dijkstra algorithm).
  3. We execute the plan step by step (now with the prescribed probabilities of outcomes) and:
     a. If the result of the action agrees with the plan, then we continue with the next action in the plan.
     b. If the result differs (i.e., the action moves the agent sideways or backwards), we go to step 1 and replan from the current position.



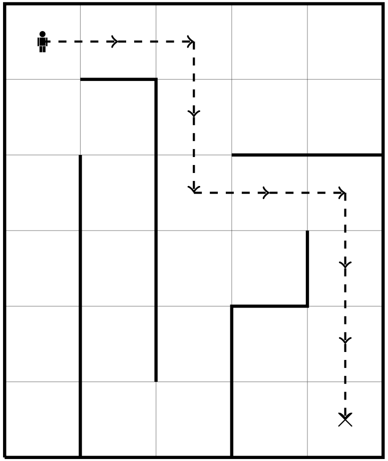
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)



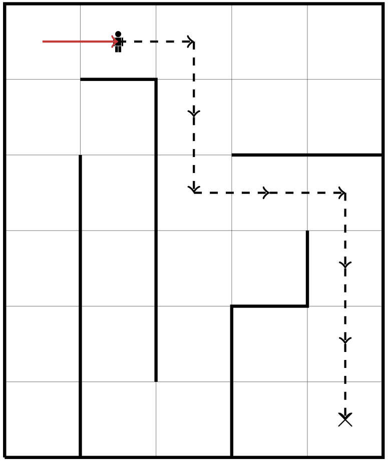
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)
Now, we execute the plan.



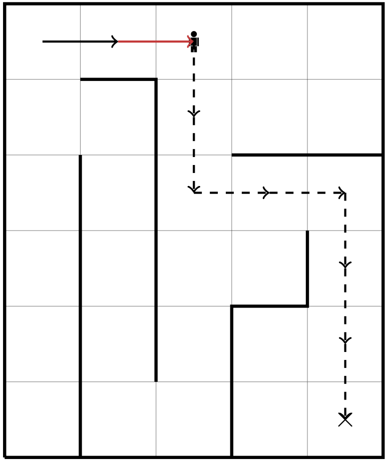
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)
Now, we execute the plan.
The first action right moves the agent right, so we continue.



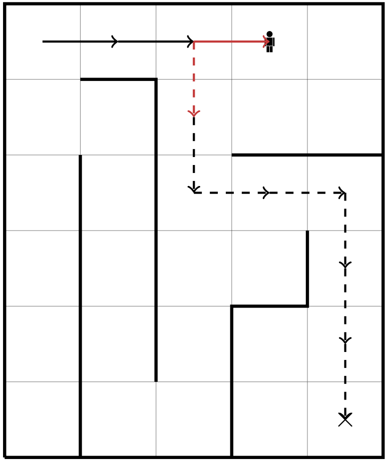
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)
Now, we execute the plan.
The first action right moves the agent right, so we continue.
The second action right also moves the agent right, so we continue.



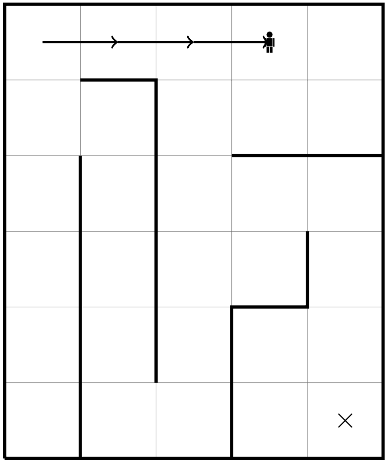
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)
Now, we execute the plan.
The first action right moves the agent right, so we continue.
The second action right also moves the agent right, so we continue.
But, the third action down moves the agent right (this was the step sideways with 15% probability).



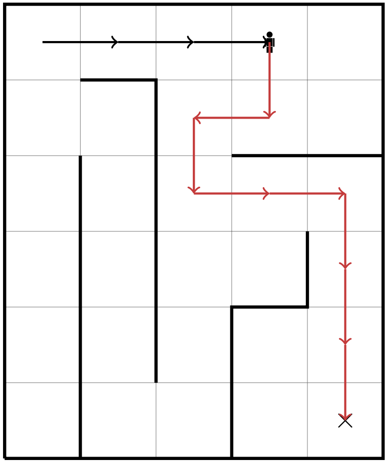
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)
Now, we execute the plan.
The first action right moves the agent right, so we continue.
The second action right also moves the agent right, so we continue.
But, the third action down moves the agent right (this was the step sideways with 15% probability).
So, we discard the current plan.



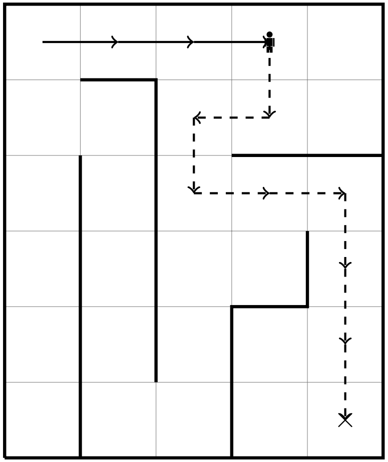
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)
Now, we execute the plan.
The first action right moves the agent right, so we continue.
The second action right also moves the agent right, so we continue.
But, the third action down moves the agent right (this was the step sideways with 15% probability).
So, we discard the current plan.
And replan, again.



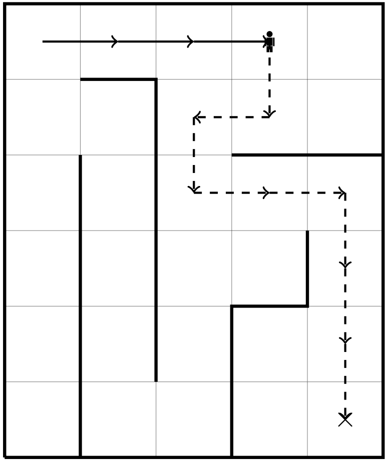
FF-replan example

So, for our example task, we start by finding a plan using dijkstra on the determinized problem.
We get: (right, right, down, down, righ, right, down, down, down)
Now, we execute the plan.
The first action right moves the agent right, so we continue.
The second action right also moves the agent right, so we continue.
But, the third action down moves the agent right (this was the step sideways with 15% probability).
So, we discard the current plan.
And replan, again.
And we execute the plan (down, left, down, right, right, down, down, down) step by step.



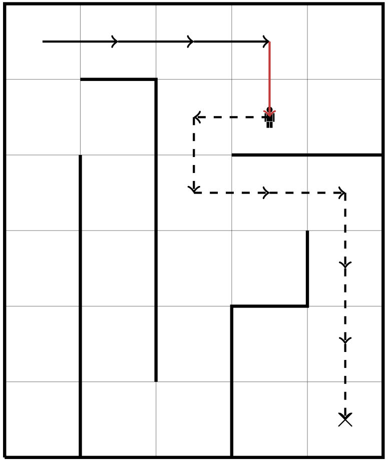
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.



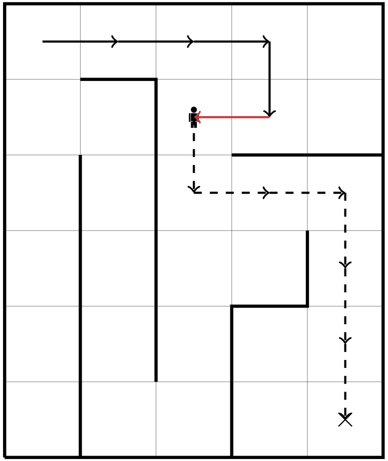
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.



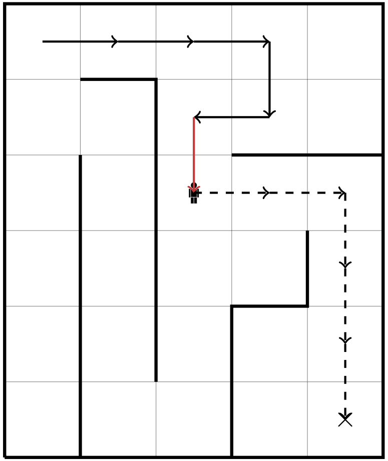
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.



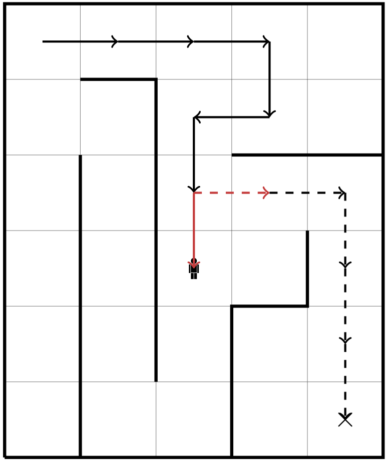
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.



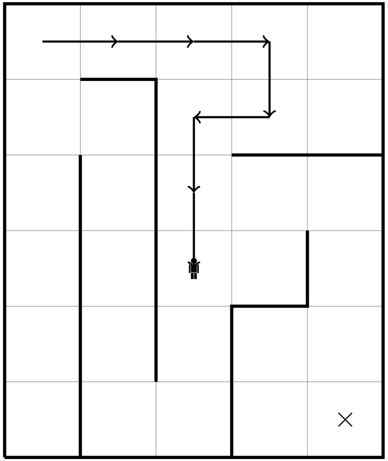
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.
But the next action right, moves the agent down.



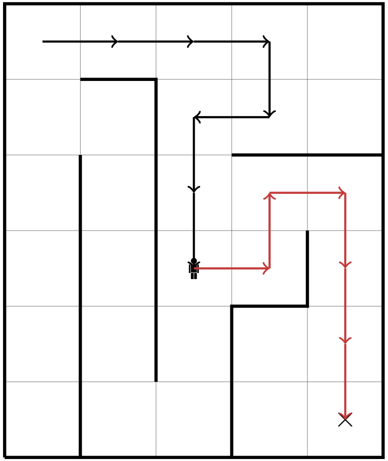
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.
But the next action right, moves the agent down.
So, we, again, discard the plan.



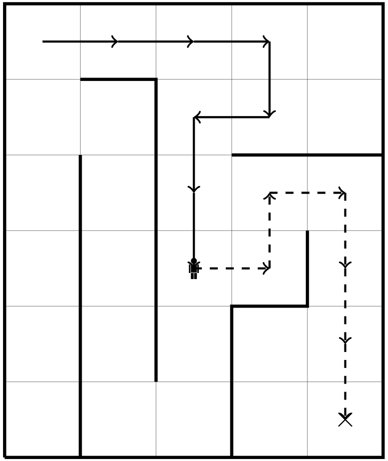
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.
But the next action right, moves the agent down.
So, we, again, discard the plan.
And replan from the current position.



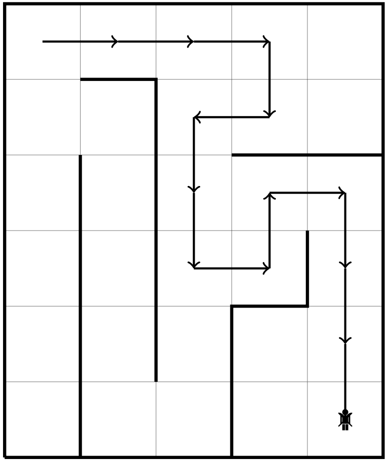
FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let’s say that the next three actions move in the correct direction.
But the next action right, moves the agent down.
So, we, again, discard the plan.
And replan from the current position.
And execute the plan (right, up, right, down, down, down).



FF-replan example

Plan: (down, left, down, right, right, down, down, down)
Not let's say that the next three actions move in the correct direction.
But the next action right, moves the agent down.
So, we, again, discard the plan.
And replan from the current position.
And execute the plan (right, up, right, down, down, down).
Now, let's say that all actions has the desired outcome. So, we reached the goal and we can terminate.



Assignment 2 description - data

The description of your assignment:

1. We will provide a dataset consisting of 25 mazes of various sizes in ascii format (on the course website).

The format of the data files is the following:

- Each file starts with two numbers specifying the number of rows and columns (in this order).
- Then the description of the maze follows in ascii: '#' corresponds to a wall, 'S' marks the starting position of the agent, 'E' marks the end position, and the space character corresponds to a free space.
- The goal is to navigate the agent from 'S' to 'E' with up/down/left/right probabilistic actions as described before.
- An example of a data file describing a 7x7 maze is depicted on the right.
- All mazes are solvable (and there are no dead-ends).

```
7 7
#S#####
#     #
### # #
# #   #
#   # #
#   # #
#####E#
```

Assignment 2 description

The description of your assignment:

1. We will provide a dataset consisting of 25 mazes of various sizes in ascii format (on the course website).
2. You have to come up with a solution to this problem and implement it in a programming language of your preference. (You can choose to solve it by both online planning as we described here, or by looking for a policy using value-iteration, or you can use any other reasonable approach.)

Assignment 2 description

The description of your assignment:

1. We will provide a dataset consisting of 25 mazes of various sizes in ascii format (on the course website).
2. You have to come up with a solution to this problem and implement it in a programming language of your preference.
3. You have to write a short (try to keep it around 3 pages) report where you have to:
    a. Describe the solution you have chosen and discuss possible shortcomings of your proposed solution.
    b. Experimentally evaluate your implementation on the given dataset.

Assignment 2 description

The description of your assignment:

1. We will provide a dataset consisting of 25 mazes of various sizes in ascii format (on the course website).
2. You have to come up with a solution to this problem and implement it in a programming language of your preference.
3. You have to write a short (try to keep it around 3 pages) report where you have to:
    a. Describe the solution you have chosen and discuss possible shortcomings of your proposed solution.
    b. Experimentally evaluate your implementation on the given dataset.
4. You have to upload your code and your report in pdf to the upload system in one zip file before the deadline (which you can find on the course website).

Assignment 2 description

The description of your assignment:

1. We will provide a dataset consisting of 25 mazes of various sizes in ascii format (on the course website).
2. You have to come up with a solution to this problem and implement it in a programming language of your preference.
3. You have to write a short (try to keep it around 3 pages) report where you have to:
   a. Describe the solution you have chosen and discuss possible shortcomings of your proposed solution.
   b. Experimentally evaluate your implementation on the given dataset.
4. You have to upload your code and your report in pdf to the upload system in one zip file before the deadline (which you can find on the course website).
5. After the deadline, your tutor will look at your solutions, read the reports and mark them with up to 20 points. Bear in mind, that the implementation part is only a smaller part of the grade, so focus on writing the report clearly and concisely. The tutor will provide a feedback on your submissions and discuss it with you (if you disagree with the mark or want more detailed feedback).

Assignment 2 grading

You will be scored on the quality of your report and the methods you have implemented. In total, you can gain **maximum of 20 points**. Each technique you implement can get you following points:

1. FF-replan [max 5 points]
2. VI and derivatives [max 10 points, max 2 points for other additional VI version]
3. MCTS [max 15 points]

You can also use any other method of your choice, ask the tutor for valuation of a technique not on this list.

Regarding the scoring of the report itself, different aspects of it will have following weights:

1. Understandability, grammer, consistency of notation [0.3]
2. Method implementation and description [0.3]
3. Experimental evaluation of methods [0.3]
4. Overall quality [0.1]

Assignment 2 grading

However, your report MUST include both method description AND evaluations section. If either of these sections is missing, you will not get more than 10 points.
For example:

- good report using FF-replan only will get you at most 5 points.
- if you implement MCTS and VI (15+10) and write a well-written report (0.3) with with good description of your methods (0.3) and a poor evaluation section (0.1), you will get $max(20, (.3 + .3 + .1 + .0) * 25) = 15$ points.

Writing tips

- Think about the experiments, consider what is the best way to evaluate and compare your algorithms. We are in the area of probabilistic planning, so you need to evaluate every maze multiple times with different seeds for your pseudo-random generator.
- You hould probably report aggregate running times and number of steps your agent needed to reach the goal. Depending on your choices, should you report some other values?
- Averages seem like a safe choice, but should you also report maximums, minimums, medians, or something else?
- You will run your methods on multiple mazes. Think about how to present the comparison in a concise way that conveys the message of your main results (please don't display huge tables with all the values). Include plots where relevant.
- Try to justify your choices.
- We recommend using latex (overleaf online, WSL with VS code on windows).
- Use spellcheck.