Chair Individual and Technology (iTec)
Stefan Schiffer, Johannes Neuhaus

Knowledge Representation and Autonomous Systems — SoSe 2023

Sheet 1

20.04.2023

due by 09.05.2023 12pm

General Information

- Form teams of 2 to work on the assignments.
- Hand in your solutions via moodle.
- **Don't print your code or create pdfs!** Instead, hand-in the source files.
- You need 50% of all regular points to take the exam.
- You may hand in solutions in teams no larger than 2 members.
- Write your names and matriculation numbers onto your solution sheet.

	Task 1	Task 2	Task 3	total
Points	10	10	5	25
reached				

Task 1 **(10pt.)**

Pacman is hungry and he started to notice that running random errands in all those mazes won't get him far. Implement two new agents which use the following strategies to select the next food to eat.

1. Depth-First Search
2. Breadth-First Search

Always start your search towards north (`Directions.NORTH`) and choose following directions in a clockwise manner. Which strategy works best under which circumstances? Give reasons for your answer using the provided sample mazes.

Task 2 **(10pt.)**

Implement Uniform-Cost Search to calculate the shortest distance from any point to any other point in a given maze, i.e. compute the set of distances $\mathcal{D}$ in a maze $\mathcal{M}$ where

$$\mathcal{D} = \{(p_1, p_2, d) \mid d = distance(path^*_{\mathcal{M}}(p_1, p_2)) \text{ and } p_1, p_2 \in \mathcal{M}\}$$

For this, implement the function `computeDistances` in `core/pacman/distanceCalculation.py`. Your code must return a dictionary `d` where `d[`p_1`,`p_2`]` accesses the distance between p_1 and p_2. Do not use any libraries.

Hint: Use `layout.isWall((x,y))` to check if there is a wall at coordinate x, y. You may also use the `PriorityQueue` implementation from `pacman.core.util`.

Task 3 **(5pt.)**

Submit any agent to the single player competition. Submit your team name to earn these points.