Koç University
COMP341
Introduction to Artificial Intelligence
Assignment 1

Instructor: Barış Akgün
Due Date: November 09 2023, 23:59
Submission Through: Blackboard
Make sure you read and understand every part of this document

This programming assignment will test your knowledge and your implementation abilities of what you have learned in the uninformed and informed search parts of the class. You are asked to complete a coding part and answer a few questions about how it runs. The coding part of the homework will follow the Berkeley CS188 Spring 2023 Pacman Project 1: Search at `https://inst.eecs.berkeley.edu/~cs188/sp23/projects/proj1/`. The questions for the report part are given in this document.

This homework must be completed individually. Discussion about algorithms, algorithm properties, code structure, and Python is allowed but group work is not. Coming up with the same approach and talking about the ways of implementation leads to very similar code which is treated as plagiarism! Furthermore, do not discuss the answers directly as it will lead to similar sentences which is treated as plagiarism. Any academic dishonesty, will not be tolerated. **By submitting your assignment, you agree to abide by the Koç University codes of conduct.**

You may find yourself having trouble implementing the coding part. In this case, we are going to let you use someone else's code to answer the given questions, as long as you credit the person or the website you take the code from. If you chose this option, we are only going to grade your report.

Important: Make sure you go over all the submission instructions at the end of this document before submission. Once you upload your submission, download it to make sure it is not corrupted and it has your latest report/code. You are only going to be graded by your blackboard submission.

Grading

You are given two options about submitting your homework: (1) both code and report, and (2) only report. The second option is given to you in case you are not able to implement the programming part. These options will be graded differently:

- **Code and Report:** The code part and the report will have 2:1 weight ratio in your submission (programming 2/3, report 1/3).
- **Only Report:** The report part will be treated as 2/3 of the total grade. You **must** credit the code you used and **must not** submit a code part.

The solution code for the homeworks can be found online. We are going to compare your submission with these sources. We are also going to compare your code to previous submissions of Koç students. If your code's similarity level is above a certain threshold, your code will be scrutinized. If there is strong suspicion of plagiarism, we will take action based on university policies.

Warnings:

- **Credit the code you used if you are submitting only a report**. You may not get any points, if you fail to do so. You can give a link to the code repository you used or write the name of your friend if you used their code.
- **Do not submit code with a report only submission**. If you include code with a report-only submission and if that code is flagged during plagiarism checks, you may not get any points.

- **You can use someone else's code to answer report questions, if you are not able to solve the corresponding programming questions**. Sometimes students are not able to solve some of the programming questions. You can use someone elses code to answer the correspondin report questions, as long as you credit the owner (see the first bullet point!). However, we cannot mix and match grading (i.e. 2/3 for the report questions that you used somebody else's code for). It is up to you to decide whether you want to be considered for report-only submission or not.

Part 1: Programming

You are going to do the 8 programming questions about search given in the website. You are only required to change *search.py* and *searchAgents.py*. If you have any issues with other parts of the code let your instructor or TA know ASAP, even if you manage to solve your problem. Use the data structures in *util.py* for the autograder to work properly. If the you think you have the right answer but the autograder is not giving you any points, try to run it on individual questions, which we have covered during the lecture time.

Hints

- We understand that having too many files to go through might be troublesome, however keep in mind that **you do not need to go over all the files**. Read the Project 1:Search documentation in the Berkeley Website and the comment sections of the relevant files/functions that you are going to work on. If you start worrying about implementation details of Pacman, you will get lost and lose track of how to handle the assignment requirements.
- **Always** read the comment sections of the given code before starting your implementation. These comments are very useful and they will save you precious time later on.

Part 2: Report

This part includes answering the following questions based on your programs' output on the given pacman tests. Look specfically at *bigMaze* and the *openMaze* for stark comparisons.

You are expected to answer the questions concisely. Five sentences is more than enough for most of them. **Limit yourself to 200 words.** It is okay if you over-generalize, as long as your direction is clear and correct. Note that some answers are already in the provided link to the Berkeley site.
Create a PDF file named *report.pdf* containing your answers for submission. **Write your name and your number on the report as well!**

Written Q1:
What are some differences between DFS and BFS in terms of path cost and number of expanded nodes? When and why would you prefer BFS over DFS? When and why would you prefer DFS over BFS?

Written Q2:
What are some differences between UCS and A* in terms of path cost and number of expanded nodes? When and why would you use UCS over A*? When and why would you prefer A* over UCS?

Written Q3:
Comment on your choice of state in the finding all the corners problem. Why does it allow you to solve the problem?

Written Q4:
Comment on your choice of heuristic in the finding all the corners problem.. Why did you settle on that heuristic? Why is it admissible and consistent?

Written Q5:
Comment on your choice of heuristic in the eating all the dots problem. Why did you settle on that heuristic? Why is it admissible and consistent?

Written Q6:
What are some practical differences between a consistent and an inadmissible heuristic, in terms of path cost and number of expanded nodes? When and why would you prefer an inadmissible heuristic over a consistent one? When and why would you prefer a consistent heuristic over an inadmissible one?

Submission

You are going to submit a compressed archive through the blackboard site. The file should extract to a folder which only contains *report.pdf*, *search.py* and *searchAgents.py*. Other files will be deleted and/or overwritten. Do not submit any code if you only want us to only grade your report.

Submission Instructions

- You are going to submit a **single** compressed archive through the blackboard site. The file can have *zip*, *rar*, *tar*, *tar.gz* or *7z* format. If you submit files individually they may get lost.
- You are fine as long as the compressed archive has the required files within 4 folder levels.
- Code that does not run (e.g. due to syntax errors), that does not terminate (e.g. due to infinite loops) or that blows up memory will not get any points.
- **Important:** Download your submission to make sure it is not corrupted and it has your latest report/code. You are only going to be graded by your blackboard submission.

Best of luck and happy coding!