Middle East Technical University   Department of Computer Engineering

CENG 462

Artificial Intelligence

Fall '2022-2023

Homework 1

Due date: 30 October 2022, Sunday, 23:55

1 Objectives

This assignment aims to familiarize you with uninformed searching methods, namely Breadth-First Search (BFS), Depth-first Search (DFS), Uniform-Cost Search (UCS), at the implementation level and enable you to gain hands-on experience on solving problems with these methods.

2 Problem Definition

You are working in a sales company. They build an agent that delivers products to customers. Figure 1 is the actual representation of start, finish, and customer points. The company translates these maps into a grid environment where your agent can move up, down, left, and right(without direct diagonal movement). In the grid environment illustrated in Figure 2, the S is the start point, the F is the finish point, and the C points are customers. An agent must deliver at least some packages to be considered successful. The agent needs to finish its task as quickly as possible, so it needs to deliver the minimum required packages.

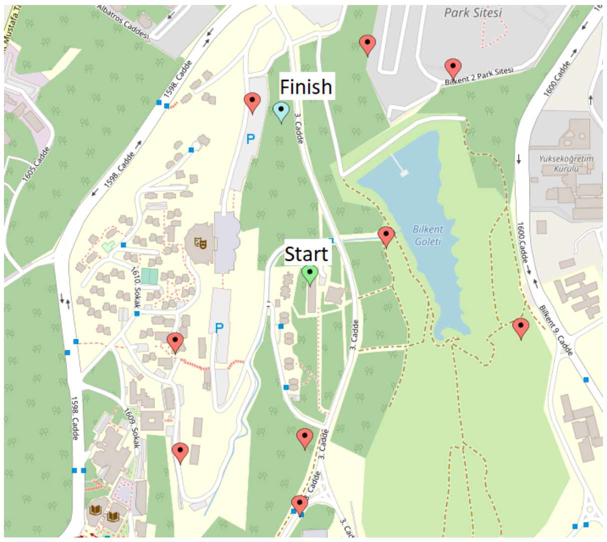


Figure 1: The actual representation of points.

In this assignment, you are expected to implement BFS,DFS and UCS methods in order to find optimal paths on given search problems.

```
.  .  .  .  .  .  C  .
.  C  .  C  .  .  .  .
.  .  .  .  .  .  C  .
.  .  F  .  .  .  .  .
.  .  .  C  .  .  .  .
.  C  .  .  .  .  C  .
.  .  .  C  .  .  .  .
.  C  .  .  S  .  .  .
```

Figure 2: Grid environment.

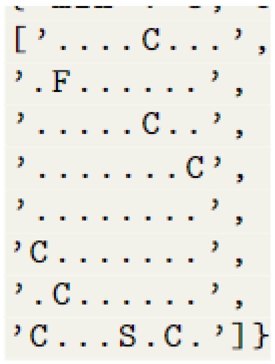
- S start cell.
- F Finish cell.
- . empty cell.
- C cell with customer

The grid environment has some features:

- The grid environment size can be changed, such as 8x8, 12x12, etc.
- The left top point is (0,0).
- S, F, and C points can be anywhere.
- The cost between cells is calculated directly from the grid environment with Manhattan distance.For example,In 2, the S coordinate is (7,4) and the F coordinate is (3,2). The agent need to 6 moves to reach F from S.The cost of between S and F is 6.

Before implementing the mentioned methods, you should represent the grid environment as a graph. In DFS and BFS, you don't need the cost between cells, but in UCS, you need to consider the cost.As mentioned above the cost is known directly from grid environment.

3 Grid environment to Graph



```
['....C...',
'.F......',
'.....C..',
'.......C',
'........',
'C.......',
'.C......',
'C...S.C.']}
```

Figure 3: Grid environment.

Let say we have In figure 3 grid environment. In DFS and BFS we need some regulations.
Firstly the left top point is (0,0).
Let give names:
(0,4) C is X1
(2,5)C is X2
(3,7)C is X3
(5,0)C is X4
(6,1)C is X5
(7,0)C is X6
(7,6)C is X7
Then the connection
S $\rightarrow X1$
$S \rightarrow X2$
$S \rightarrow X3$
$S \rightarrow X4$
$S \rightarrow X5$
$S \rightarrow X6$
$S \rightarrow X7$
$S \rightarrow F$
$X1 \rightarrow X2$
$X1 \rightarrow X3$
$\ldots$
$X1 \rightarrow F$
$X2 \rightarrow X1$
$\ldots$

Take this as a reference when you do DFS and BFS.
NOTE: In DFS and BFS, we cannot guarantee optimality. So when you reach the condition terminate and return.

4 Specifications

- You are going to implement BFS, DFS, and UCS in Python 3.
- Each test case is represented with a file and this file will be fed to your implementation along with one of the methods.
- To unify all methods in a single place, you are expected to write the following function:

```python
def UnInformedSearch(method_name, problem_file_name):
```

  whose parameter specifications are as follows:

  - `method_name`: specifies which method to be run for the given problem. It can be assigned to one of the values from `["BFS", "DFS", "UCS"]`.
  - `problem_file_name`: specifies the path of the problem file to be solved.

  During the evaluation process, this function will be called and returned information will be inspected. The returned information is dependent on the method fed and as follows:

  - `method_name=="BFS"`: `UnInformedSearch` returns the solution sequence (as a list) when condition is satisfy, the list of delivered customer coordinates. If no solution exists then it returns a single `None`.
  - `method_name=="DFS"`: `UnInformedSearch` returns the solution sequence (as a list) when condition is satisfy, the list of delivered customer coordinates. If no solution exists then it returns a single `None`.
  - `method_name=="UCS"`: `UnInformedSearch` returns the optimal solution sequence (as a list), the list of delivered customer coordinates. If no solution exists then it returns a single `None`.

- Problems are described in ".txt" files and have the following structure:

```
{'min': 2,'env':
['....C...',
'.F......',
'.....C..',
'.......C',
'........',
'C.......',
'.C......',
'C...S.C.']}
```

- No `import` statement is allowed and all methods should be implemented in a single solution file. You are expected to implement all the necessary operations by yourself without utilizing any external library.
- Commenting is crucial for understanding your implementation and decisions made during the process. Your implementation can be manually inspected at random or when it does not satisfy the expected outputs.

5 Sample I/O

sampleproblem1.txt:

```
{'min': 2,'env':
['....C...',
'.F......',
'.....C..',
'.......C',
'........',
'C.......',
'.C......',
'C...S.C.']}
```

Expected outputs of the UnInformedSearch function with different methods on applied on sampleproblem1.txt:

```
>>> import hw1solutioneXXXXXXX as hw1
>>> print(hw1.UnInformedSearch("BFS", "sampleproblem1.txt"))
#return the path as list
>>> print(hw1.UnInformedSearch("DFS", "sampleproblem1.txt"))
#return the path as list
>>> print(hw1.UnInformedSearch("UCS", "sampleproblem1.txt"))
#return the path as list
```

6 Regulations

1. **Programming Language:** You must code your program in Python 3. Since there are multiple versions and each new version adds a new feature to the language, in order to concur on a specific version, please make sure that your implementation runs on İnek machines.

2. **Implementation:** You have to code your program by only using the functions in the standard module of python. Namely, you **cannot** import any module in your program.

3. **Late Submission:** No late submission is allowed. Since we have a strict policy on submissions of homework in order to be able to attend the final exam, please pay close attention to the deadlines.

4. **Cheating: We have zero-tolerance policy for cheating**. People involved in cheating (any kind of code sharing and codes taken from the internet included) will be punished according to the university regulations.

5. **Discussion:** You must follow ODTUClass for discussions and possible updates on a daily basis. If you think that your question concerns everyone, please ask them on ODTUClass.

6. **Evaluation:** Your program will be evaluated automatically using the "black-box" technique so make sure to obey the specifications. A reasonable timeout will be applied according to the complexity of test cases. This is not about the code efficiency, its only purpose is avoiding infinite loops due to an erroneous code. In the case your implementation does not conform to expected outputs for the solutions, it will be inspected manually. So having comments that explain the implementation in an elaborate manner may become crucial for grading in this case.

7 Submission

Submission will be done via ODTUClass system. You should upload a **single** python file named in the format **hw1_e<your-student-id>.py** (i.e. hw1_e1234567.py).

8 References

- For the `TREE-SEARCH` algorithm, you can refer to the lecture's textbook.
- Announcements Page
- Discussions Page