Middle East Technical University | Department of Computer Engineering

CENG 462

Artificial Intelligence

Fall '2021-2022

Homework 1

Due date: 13 November 2021, Saturday, 23:55

1 Objectives

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

2 Problem Definition

The problems such as calculating the shortest path on a given map for a pair of start and destination locations, path-planning for robot guidance in an environment, extracting the router sequence for a network package to be delivered from a source node to a destination node on a network require a search operation to be performed. Finding optimal paths or actions are the ultimate goals in the fields that involve these kinds of problems. Previously mentioned methods provide solutions to these problems by having different assumptions. For instance, BSF, Depth-first Search (DFS), DLS, IDDFS do not take the cost information into account during a search operation, whereas Uniform-Cost search directly performs and directs its calculations by depending on cumulative costs. Both sets of methods have their own application areas. For instance, in a computer network, if it is asked to find the route that contains the fewest routers (minimum number of hops) between a source and destination nodes, the methods in the first group can be considered, whereas, when the path which incurs the least amount of communication cost is needed, the Uniform-Cost search method is the inevitable choice.

In this assignment, you are expected to implement previously mentioned methods in order to find optimal paths or routes on given search problems. These include two types: finding the shortest path on a map for a pair of source and destination locations and extracting the best/fastest communication route between two nodes. In Figure 1 a graphical representation of the map of some states in the USA is depicted, where the values on the edges are fictional distance values (in kilometers) between the states. Similarly, Figure 2 illustrates a sample network topology consisting of different types of network components (nodes), and the communication costs (packet transferring time) between nodes are given as edge labels.

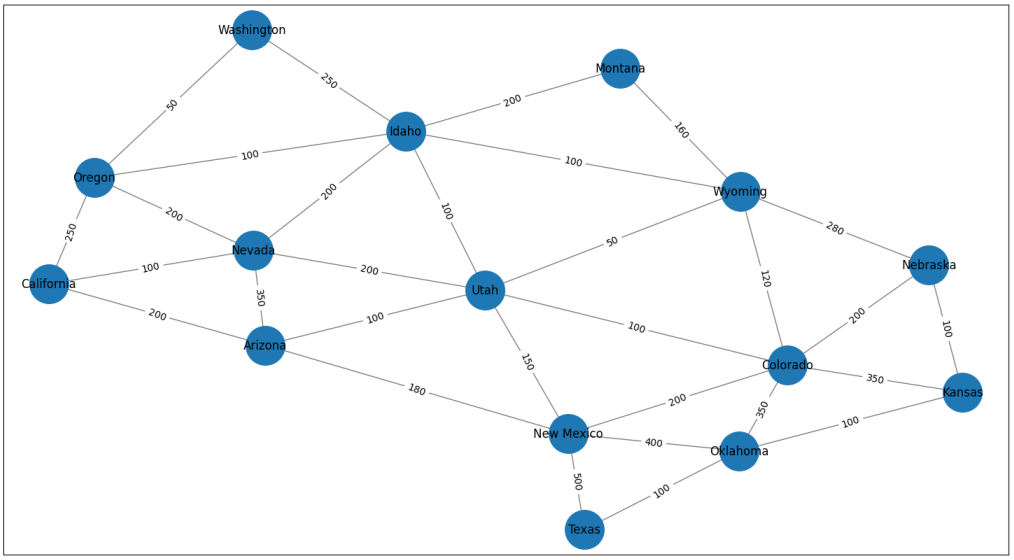

Figure 1: Some states in the USA with fictional distance values.

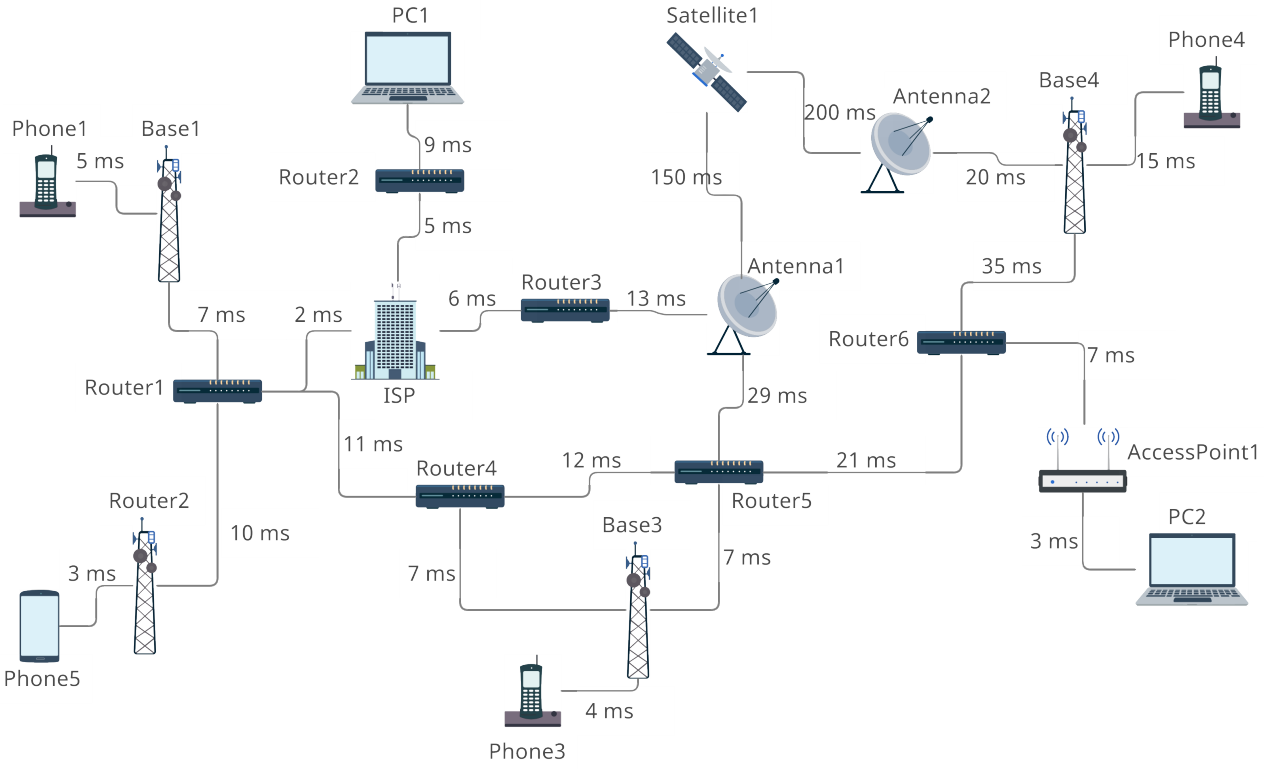

Figure 2: A sample computer network with different types of nodes.

3 Specifications

- You are going to implement BFS, DLS, IDDFS, and UCS in Python 3.

- Each problem 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:

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

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", "DLS", "IDDFS", "UCS"]`.
  - `problem_file_name`: specifies the path of the problem file to be solved.
  - `maximum_depth_limit`: specifies the maximum depth limit for DLS and IDDFS methods. This parameter takes its value from `[1, 2, 3, ... 100]`.

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 optimal solution sequence (as a list), the list of processed nodes/states/locations during search and the depth value of the found target node. If no solution exists then it returns a single `None`.
  - `method_name=="DLS"`: `UnInformedSearch` returns the optimal solution sequence (as a list), the list of processed nodes/states/locations during search and the depth value of the found target node. If no solution is found then it returns a single `None`.
  - `method_name=="IDDFS"`: `UnInformedSearch` returns the optimal solution sequence (as a list), the list of processed nodes/states/locations during search and the depth value of the found target node. If no solution is found then it returns a single `None`.
  - `method_name=="UCS"`: `UnInformedSearch` returns the optimal solution sequence, the list of processed nodes/states/locations during search, the depth value of the found target node and the optimal solution cost. If no solution is found then it returns a single `None`.

NOTE: The order of nodes/states/locations in the processed nodes/states/locations list is not important and the depth value for the starting node is 0.

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

```
<start node/location/state>
<target node/location/state>
node1 node2 cost1
node1 node3 cost2
node2 node4 cost4
.
.
.
```

The first line specifies the node/location/state where the algorithms start searching. Any node can be the source node. Similarly, the second line specifies the target node and any node is a candidate for the target node. The rest of the lines describes the problem graph (undirected graph, as Figures 1 and 2). Each edge on the graph is represented with a line in the file. Nodes of an edge are specified sequentially along with the cost value for the edge.

Node names could be any string sequence (of alphanumeric characters) that does not contain the space character (`" "`) (e.g. metu, node1, ankara, router100, etc.). The cost values are of type integer (e.g. 15, 3, 125, 1000, etc.).

- 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.

NOTE: In this assignment, during search tree construction you are expected not to eliminate multiple occurrences of a node (in other words, you are supposed to implement all methods with the `TREE-SEARCH` algorithm), which causes the DFS method to get stuck in an infinite search loop, this is the reason why DFS is not among the methods to be implemented.

4 Sample I/O

sampleproblem1.txt:

```
izmir
ankara
ankara afyon 300
ankara konya 5
afyon eskisehir 150
eskisehir usak 45
ankara eskisehir 150
manisa usak 250
izmir manisa 125
manisa afyon 150
```

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

```
>>> import hw1solutioneXXXXXXX as hw1
>>> print(hw1.UnInformedSearch("BFS", "sampleproblem1.txt", None))
(['izmir', 'manisa', 'afyon', 'ankara'], ['izmir', 'manisa', 'usak', 'izmir', '
    afyon', 'eskisehir', 'manisa', 'manisa', 'ankara'], 3)
>>> print(hw1.UnInformedSearch("DLS", "sampleproblem1.txt", 2))
None
>>> print(hw1.UnInformedSearch("DLS", "sampleproblem1.txt", 3))
(['izmir', 'manisa', 'afyon', 'ankara'], ['izmir', 'manisa', 'afyon', 'manisa',
    'eskisehir', 'ankara'], 3)
>>> print(hw1.UnInformedSearch("IDDFS", "sampleproblem1.txt", 3))
(['izmir', 'manisa', 'afyon', 'ankara'], ['izmir', 'manisa', 'afyon', 'manisa',
    'eskisehir', 'ankara'], 3)
>>> print(hw1.UnInformedSearch("IDDFS", "sampleproblem1.txt", 20))
(['izmir', 'manisa', 'afyon', 'ankara'], ['izmir', 'manisa', 'afyon', 'manisa',
    'eskisehir', 'ankara'], 3)
>>> print(hw1.UnInformedSearch("UCS", "sampleproblem1.txt", None))
(['izmir', 'manisa', 'usak', 'eskisehir', 'ankara'], ['izmir', 'manisa', 'izmir
    ', 'afyon', 'usak', 'manisa', 'eskisehir', 'eskisehir', 'manisa', 'usak', '
    usak', 'izmir', 'eskisehir', 'eskisehir', 'afyon', 'izmir', 'usak', 'usak',
    'ankara'], 4, 570)
```

sampleproblem2.txt:

```
phone2
phone3
phone1 station1 30
station1 router2 4
station1 router1 2
station1 router1 6
station1 router3 3
station1 phone2 20
router2 router1 15
router2 router1 18
router1 station2 25
router1 router3 20
router3 station2 10
station3 station2 5
station2 phone3 4
station3 station2 5
```

Expected outputs of UnInformedSearch function with different methods applied on sampleproblem2.txt:

```
>>> import hw1solutioneXXXXXXX as hw1
>>> print(hw1.UnInformedSearch("BFS", "sampleproblem2.txt", None))
(['phone2', 'station1', 'router1', 'station2', 'phone3'], ['phone2', 'station1
   ', 'phone1', 'router2', 'router1', 'router3', 'phone2', 'station1', '
   station1', 'router1', 'station1', 'router2', 'station2', 'router3', '
   station1', 'router1', 'station2', 'station1', 'phone1', 'router2', 'router1
   ', 'router3', 'phone2', 'phone1', 'router2', 'router1', 'router3', 'phone2',
    'station1', 'router2', 'station2', 'router3', 'phone1', 'router2', 'router1
   ', 'router3', 'phone2', 'station1', 'router1', 'router1', 'router3', '
   station3', 'phone3'], 4)
>>> print(hw1.UnInformedSearch("DLS", "sampleproblem2.txt", 2))
None
>>> print(hw1.UnInformedSearch("DLS", "sampleproblem2.txt", 3))
None
>>> print(hw1.UnInformedSearch("DLS", "sampleproblem2.txt", 4))
(['phone2', 'station1', 'router3', 'station2', 'phone3'], ['phone2', 'station1
   ', 'phone2', 'station1', 'phone2', 'router3', 'router1', 'router2', 'phone1
   ', 'router3', 'station2', 'phone3'], 4)
>>> print(hw1.UnInformedSearch("IDDFS", "sampleproblem2.txt", 3))
None
>>> print(hw1.UnInformedSearch("IDDFS", "sampleproblem2.txt", 20))
(['phone2', 'station1', 'router3', 'station2', 'phone3'], ['phone2', 'station1
   ', 'phone2', 'station1', 'phone2', 'router3', 'router1', 'router2', 'phone1
   ', 'router3', 'station2', 'phone3'], 4)
>>> print(hw1.UnInformedSearch("UCS", "sampleproblem2.txt", None))
(['phone2', 'station1', 'router3', 'station2', 'phone3'], ['phone2', 'station1
   ', 'router3', 'router2', 'router1', 'station1', 'station1', 'router3', '
   router2', 'router3', 'router2', 'station1', 'router1', 'station1', 'station2
   ', 'router1', 'station1', 'station1', 'router3', 'router3', 'station1', '
   router2', 'router2', 'phone3'], 4, 37)
```

5 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.

6 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).

7 References

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