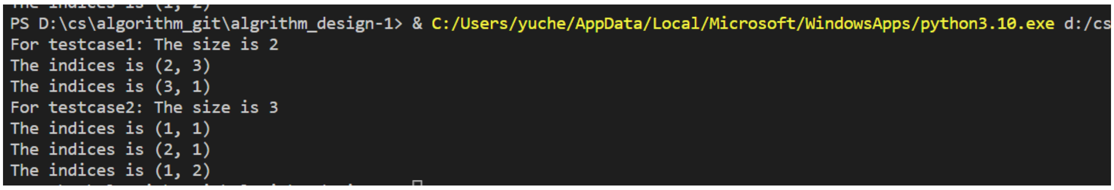
Report

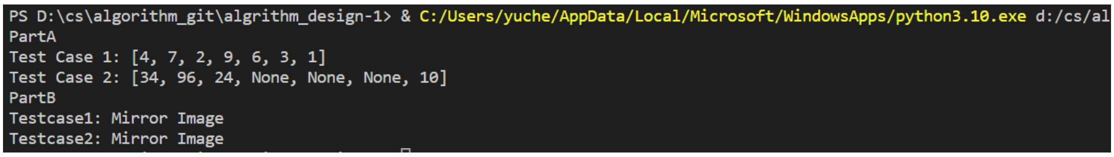
4591774
Yuchen Zhang

Problem 1:



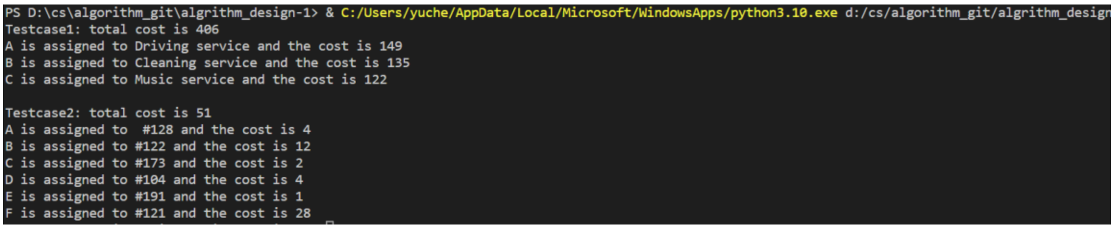
```
The indices is (1, 2)
PS D:\cs\algorithm_git\algrithm_design-1> & C:/Users/yuche/AppData/Local/Microsoft/WindowsApps/python3.10.exe d:/cs
For testcase1: The size is 2
The indices is (2, 3)
The indices is (3, 1)
For testcase2: The size is 3
The indices is (1, 1)
The indices is (2, 1)
The indices is (1, 2)
```

Problem2:



```
PS D:\cs\algorithm_git\algrithm_design-1> & C:/Users/yuche/AppData/Local/Microsoft/WindowsApps/python3.10.exe d:/cs/al
PartA
Test Case 1: [4, 7, 2, 9, 6, 3, 1]
Test Case 2: [34, 96, 24, None, None, None, 10]
PartB
Testcase1: Mirror Image
Testcase2: Mirror Image
```

Problem3:



```
PS D:\cs\algorithm_git\algrithm_design-1> & C:/Users/yuche/AppData/Local/Microsoft/WindowsApps/python3.10.exe d:/cs/algorithm_git/algrithm_design
Testcase1: total cost is 406
A is assigned to Driving service and the cost is 149
B is assigned to Cleaning service and the cost is 135
C is assigned to Music service and the cost is 122

Testcase2: total cost is 51
A is assigned to  #128 and the cost is 4
B is assigned to #122 and the cost is 12
C is assigned to #173 and the cost is 2
D is assigned to #104 and the cost is 4
E is assigned to #191 and the cost is 1
F is assigned to #121 and the cost is 28
```

The time complexity is O(n^3)
The first step and the second step scan and update the elements in the matrix. Since there are a total of m * n(It can also be approximated as the maximum value of n among m and n, so m * n become n^2) elements in the matrix, the time complexity of both steps is O(n^2)
The third step is to cover all zero elements with a straight line, so this step needs to visit all zero elements, where the number of zero elements is up to n^2.
The fourth step requires scanning and updating the elements of the matrix by up to n^2.
Thus the third step, and the fourth step are both O(n^2).
But the third and fourth steps need to be iterated until the minimum number of straight lines is equal to n when the iteration is stopped. At each iteration, the minimum number of straight lines increases by at least 1, so the maximum number of iterations is n.
Therefore, the third step, the fourth step has the highest time complexity of O(n^3).
The fifth step yields the final assignment scheme with a time complexity of O(n).

Thus the time complexity of the Hungarian algorithm is O(n^3).

# The resource is from https://python.plainenglish.io/hungarian-algorithm-introduction-python-implementation-93e7c0890e15, https://brilliant.org/wiki/hungarian-matching/#citation-3 and https://blog.csdn.net/u014754127/article/details/78086014