LTCE

Experiment No. 5

Job Sequencing Problem

Aim - To implement Job sequencing problem.

Theory -

Algo -

1) Sort all the jobs in decreasing order of profit.
2) Iterate on jobs in decreasing order of profit for each job do the followings

a) Find a time slot i is greatest. Put the job in this slot & mark this slot filled.

b) If no such exists, then ignore the job.

Example -

Input - Five Jobs with following deadlines & profits.

Job-ID	Deadline	Profit
a	2	100
b	1	19
c	2	27
d	1	25
e	3	15

Output - following is the maximum profit sequence of jobs

c, a, e.

LTCE

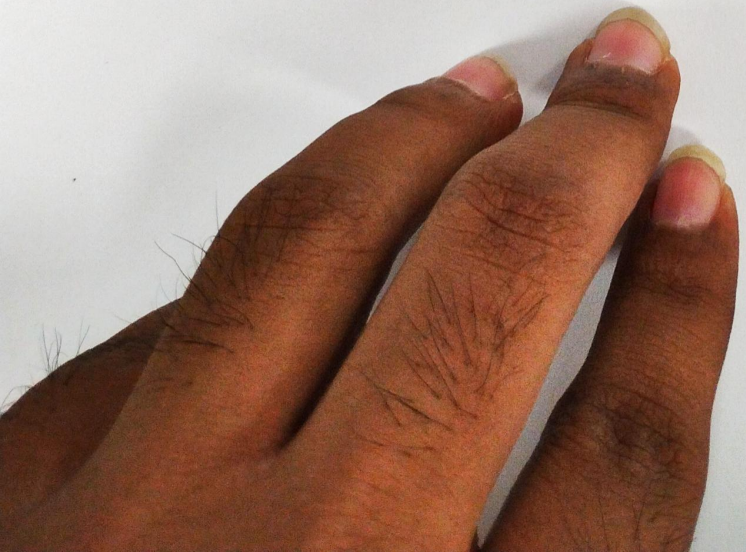
Time complexity -

O(n2) is the time complexity of the program.

Conclusion -

Hence, we have successfully implemented algo + analysed the time complexity.

```
#job seqenching                                script
class Job:
        def __init__(self, taskId, deadline, profit):
                self.taskId = taskId
                self.deadline = deadline
                self.profit = profit
def scheduleJobs(jobs, T):
        profit = 0
        slot = [-1] * T
s       jobs.sort(key=lambda x: x.profit, reverse=True)

        for job in jobs:
                for j in reversed(range(job.deadline)):
                        if j < T and slot[j] == -1:
                                slot[j] = job.taskId
                                profit += job.profit
                                break
        print('The scheduled jobs are', list(filter(lambda x: x != -1, slot)))

        print('The total profit earned is', profit)

if __name__ == '__main__':

        jobs = [
                Job(1, 9, 15), Job(2, 2, 2), Job(3, 5, 18), Job(4, 7, 1), Job(5,
4, 25),
                Job(6, 2, 20), Job(7, 5, 8), Job(8, 7, 10), Job(9, 4, 12),
Job(10, 3, 5)
        ]

        T = 15
        scheduleJobs(jobs, T)
```



Shell

```
The scheduled jobs are [7, 6, 9, 5, 3, 4, 8, 1]
The total profit earned is 109
> 
```

LTCE

Practical No. 6

All pair shortest path : Floyed Warshall Algorithm

Aim - Implementation of floyed Warshall algo

Theory -

Algo -

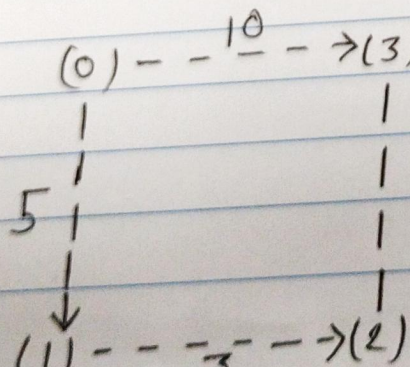
```
// L is the matrix of size n.n representaing original graph
// D is the distance matrix.
  D ← L
  for k ← 1 to n do
    for i ← 1 to n do
      for j ← 1 to n do
      D[i,j]k ← min (D[i,j]k-1, D[i,k]k-1 + D[k,j]k-1)
       end
    end
  end
  return D
```

Example -

Input:

```
graph[][] = {{0, 5, INF, 10},
             {INF, 0, 3, INF},
             {INF, INF, 0, 1},
             {INF, INF, INF, 0}}
```

which represents the following graph



Note that the value of graph[i][j] is 0 if i is equal to j & graph[i][j] is ∞ (infinite) if there is no edge from vertex

LTCE

Output - Shortest distance matrix.

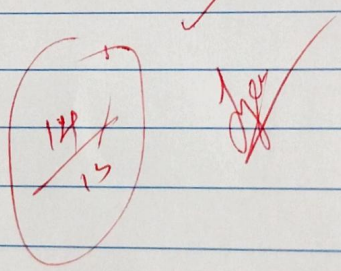
0	5	8	9
INF	0	3	4
INF	INF	0	1
INF	INF	INF	0

Time complexity -

$O(V^3)$ is the time complexity.

Conclusion -

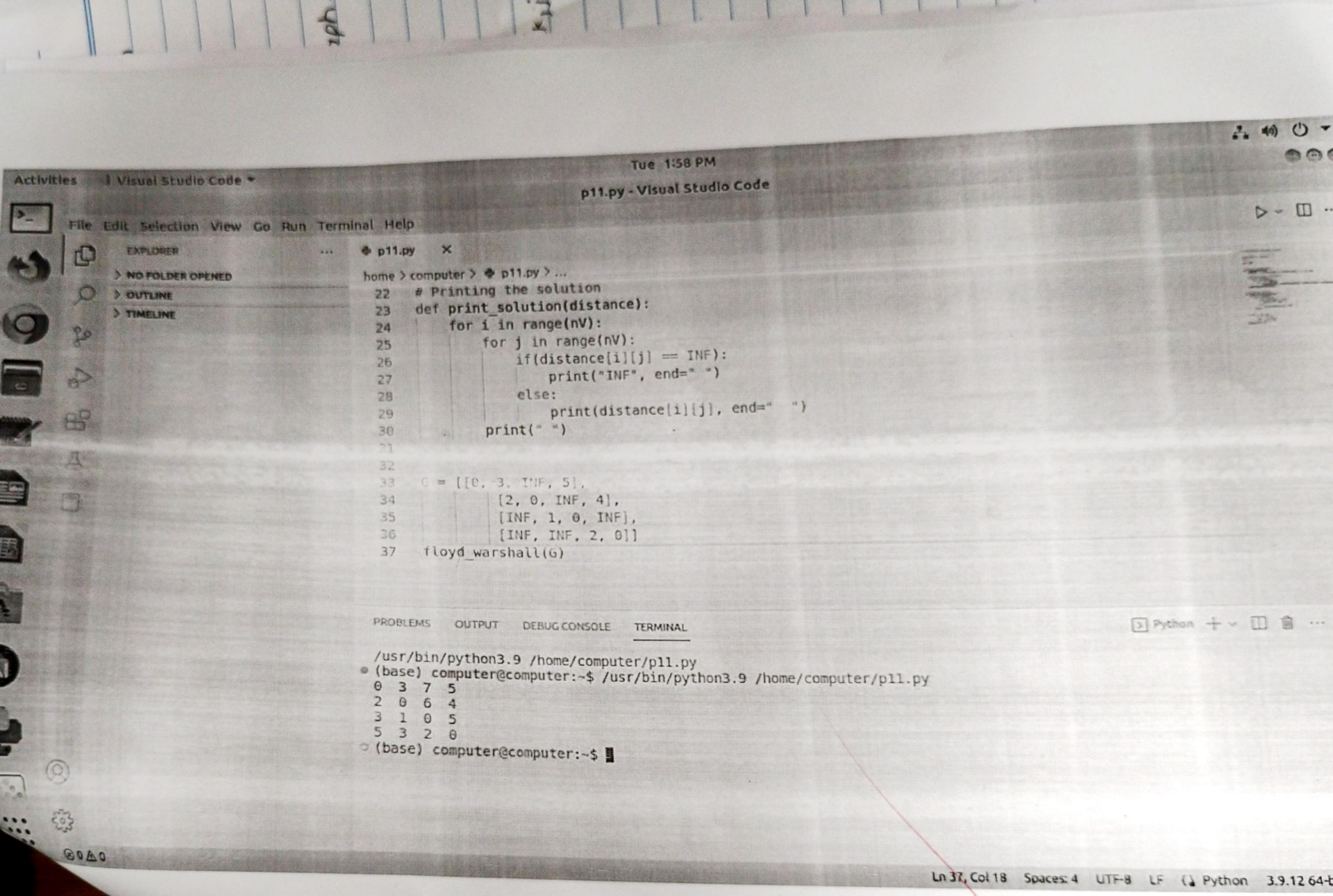
We have successfully implemented the algorithm & analyzed the algo.



LTCE

floyd

```
# Python3 Program for Floyd Warshall Algorithm
V = 4
INF = 99999

def floydWarshall(graph):

        dist = list(map(lambda i: list(map(lambda j: j, i)), graph))
        for k in range(V):
                for i in range(V):
                        for j in range(V):
                                dist[i][j] = min(dist[i][j],
                                dist[i][k] + dist[k][j]
                                )
        printSolution(dist)

def printSolution(dist):
        print("Following matrix shows the shortest distances\
between every pair of vertices")
        for i in range(V):
                for j in range(V):
                        if(dist[i][j] == INF):
                                print("%7s" % ("INF"), end=" ")
                        else:
                                print("%7d\t" % (dist[i][j]), end=' ')
                        if j == V-1:
                                print()

if __name__ == "__main__":
graph = [[0, 5, INF, 10],
                [INF, 0, 3, INF],
                [INF, INF, 0, 1],
                [INF, INF, INF, 0]
                ]
floydWarshall(graph)
```


Activities
Visual Studio Code
Tue 1:58 PM
p11.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NO FOLDER OPENED
OUTLINE
TIMELINE
p11.py
home > computer > p11.py > ...
22 # Printing the solution
23 def print_solution(distance):
24 for i in range(nV):
25 for j in range(nV):
26 if(distance[i][j] == INF):
27 print("INF", end=" ")
28 else:
29 print(distance[i][j], end=" ")
30 print(" ")
33 G = [[0, 3, INF, 5],
34 [2, 0, INF, 4],
35 [INF, 1, 0, INF],
36 [INF, INF, 2, 0]]
37 floyd_warshall(G)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
/usr/bin/python3.9 /home/computer/p11.py
(base) computer@computer:~$ /usr/bin/python3.9 /home/computer/p11.py
0 3 7 5
2 0 6 4
3 1 0 5
5 3 2 0
(base) computer@computer:~$
Ln 37, Col 18 Spaces: 4 UTF-8 LF Python 3.9.12 64-b

LTCE

Experiment No. 7

Largest Common Subsequence

Aim - To implement longest common subsequences problem

Theory -

Algo -

```
LCS - Length (x, y)
1   m ← length [x]
2   n ← length [y]
3   for i ← 1 to m
4   do c[i, 0) ← 0
5   for j ← 0 to m
6   do c[0, j] ← 0
7   for i ← 0 to m
8   do for j ← 1 to n
9   do if x i = y j
10  then c[i, j] ← c[1-i, j-i] +1
11  b[i, j] ← "↖"
12  else if c[i, -1, j] ≥ c[i, j-1]
13  then c[i, j] ← c[i, -1, j]
14  b[i, j] ← "↑"
15  else c[i, j] ← c[i, j, -1]
16  b[i, j] ← "←"
17  return c & b
```

Example -

Input sequences for LCS "ABCDEFGH" & ~~"ABC"~~ "AEDFHR" is "ADH" of length 3.

Input sequences for LCS "AGGTAB" & "GXTXAYB" is "GTAB" of length 4.

Consider the input strings "AGGTAB" & "GXTXAYB" Last char match the strings, so length of LCS can be written as: L("AGGTAB","GXTXAYB") = 1+L("AGGTA", "GXTXAY")

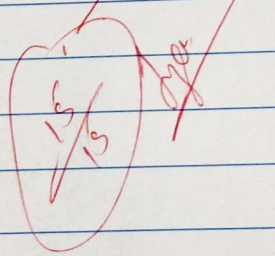
	A	G	G	T	A	B
G	-	-	4	-	-	-
X	-	-	-	-	-	-
T	-	-	-	3	-	-
X	-	-	-	-	-	-
A	-	-	-	-	2	-
Y	-	-	-	-	-	-
B	-	-	-	-	-	1

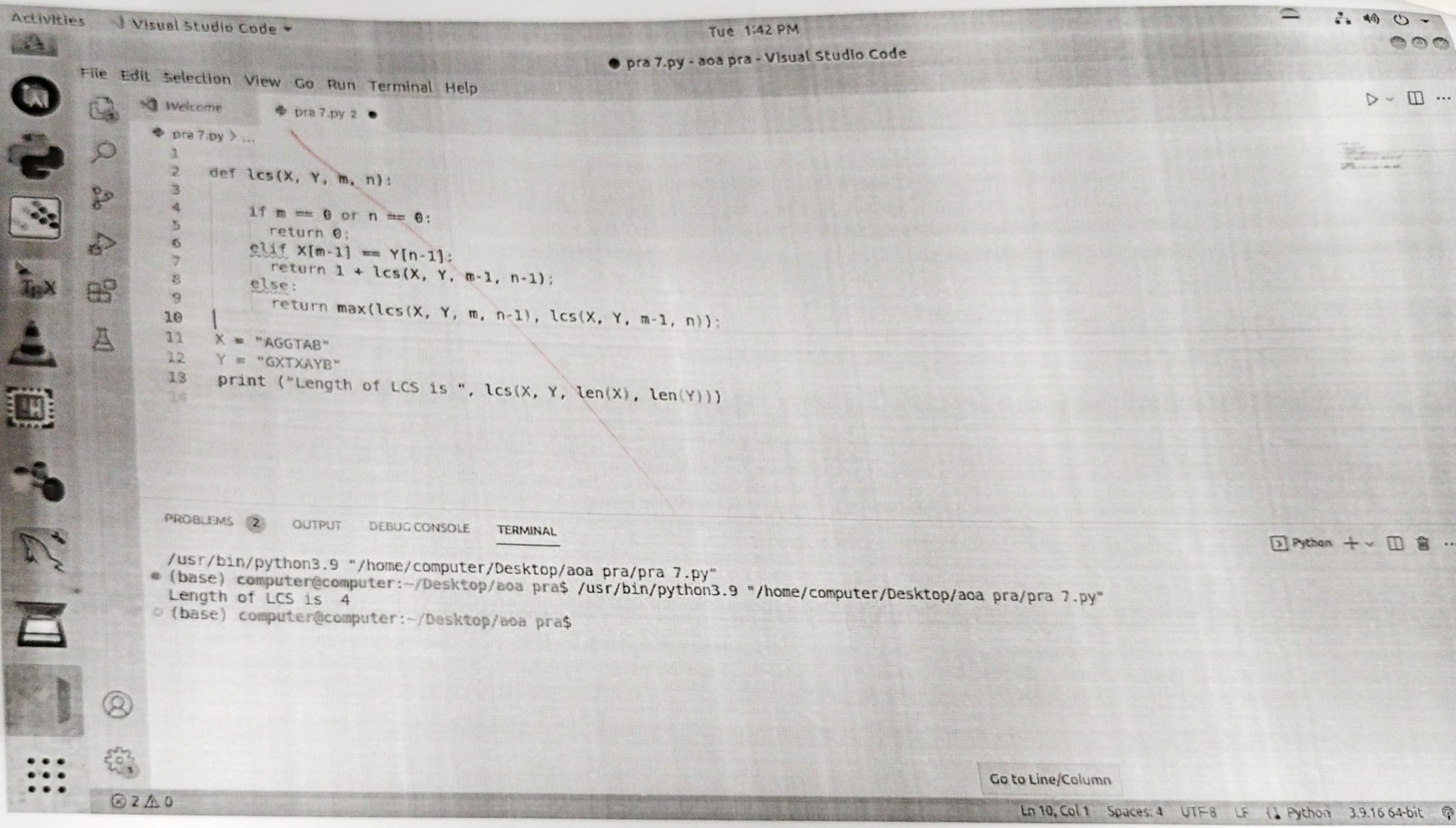
Time complexity -

$O(2^{\wedge\theta})$ is time complexity.

Conclusion -

We have successfully implemented algo & analysed the complexity.




Activities
Visual Studio Code
Tue 1:42 PM
pra 7.py - aoa pra - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Welcome
pra 7.py 2
pra 7.py > ...
def lcs(X, Y, m, n):
if m == 0 or n == 0:
return 0;
elif X[m-1] == Y[n-1]:
return 1 + lcs(X, Y, m-1, n-1);
else:
return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
X = "AGGTAB"
Y = "GXTXAYB"
print ("Length of LCS is ", lcs(X, Y, len(X), len(Y)))
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
Python
/usr/bin/python3.9 "/home/computer/Desktop/aoa pra/pra 7.py"
(base) computer@computer:~/Desktop/aoa pra$ /usr/bin/python3.9 "/home/computer/Desktop/aoa pra/pra 7.py"
Length of LCS is 4
(base) computer@computer:~/Desktop/aoa pra$
Go to Line/Column
Ln 10, Col 1
Spaces: 4
UTF-8
LF
Python
3.9.16 64-bit