Assignment 8

ELP-718 Telecom Software Lab

Manas Ranjan Patro

2018JTM2775

A report presented for the assignment on

Python/Github

Bharti School Of

Telecommunication and Management

IIT Delhi

20 September 2018

Contents

1 Problem Statement 1 **3**

1.1 Problem Statement 3

1.2 Program Structure 6

1.3 Problem Analysis 7

2 Problem Statement 2 **8**

2.1 Problem Statement 8

2.2 Problem analysis 9

3 Appendix **9**

3.1 Appendix-A : code for ps1 9

3.2 Appendix-B : code for ps2 11

3.3 Screenshots 13

List of Figures

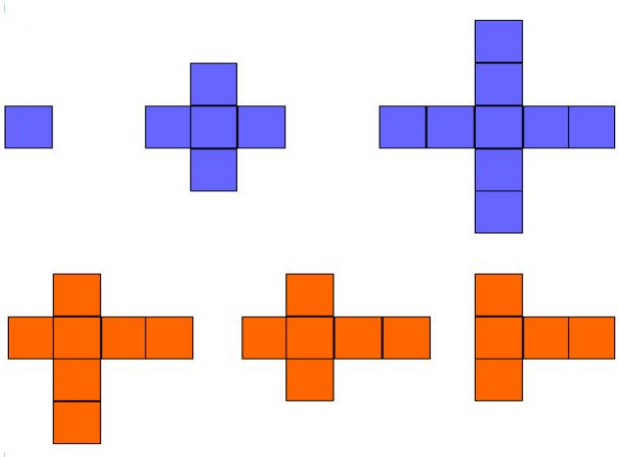
1 *Problem statement 1 output* 14
2 *Problem statement 2 output* 15

1 Problem Statement 1

1.1 Problem Statement

IIT Delhi, has just got the strongest computer. The professors in charge wants to check the computational capacity of the computer. So, they decided to create the problem which is to be given as an assignment to students. Can you help the professor to check the computation capability of the computer?

A valid cross is defined here as the two regions (horizontal and vertical) of equal lengths crossing over each other. These lengths must be odd, and the middle cell of its horizontal region must cross the middle cell of its vertical region.



In the diagram above, the blue crosses are valid and the orange ones are not valid.

Find the two largest valid crosses that can be drawn on smart cells in the grid, and return two integers denoting the dimension of the each of the

two largest valid crosses. In the above diagrams, our largest crosses have dimension of 1, 5 and 9 respectively.

Note: The two crosses cannot overlap, and the dimensions of each of the valid crosses should be maximal.

Input Format The first line contains two space-separated integers, n and m.

Each of the next lines n contains a string of m characters where each character is either S (Smart) or D (Dull). These strings represent the rows of the grid. If the jth character in the ith line is S, then (i,j) is a cell smart. Otherwise it's a dull cell.

Constraints

- n=[2,105]
- n=[2,105]

Output Find two valid crosses that can be drawn on smart cell of the grid, and return the dimension of both the crosses in the reverse sorted order(i.e. First Dimension should be the larger one and other should be smaller one).

Sample Input

5 6

SSSSSS

SDDDSD

SSSSSS

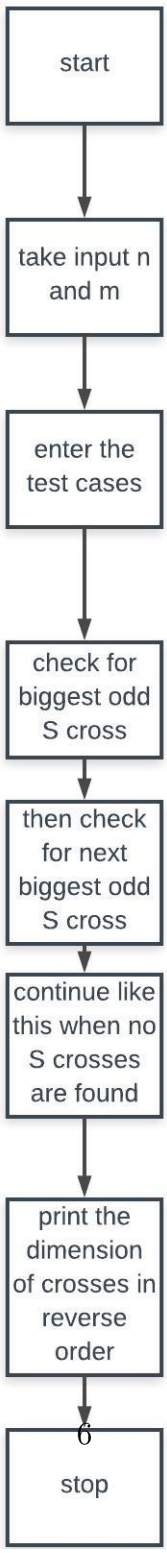
SSDDSD

SSSSSS

Sample Output

5 1

1.2 Program Structure



1.3 Problem Analysis

- First take two inputs n and m. n is for test cases and m is for no of characters in each line.
- Search for biggest valid S cross especially of odd order.
- Continue search for next biggest odd order S cross.
- Continue till no cross can be found.
- Print the dimension in reverse order.

2 Problem Statement 2

2.1 Problem Statement

[**?**] After, getting mix results of valid crosses, professors decided to test the computation abilities on one more problem. This time professors wanted to test the decryption capabilities of the computer.

Encryption of a message requires three keys, k1, k2, and k3. The 26 letters of English and underscore are divided in three groups, **[a-i]** form one group, **[j-r]** a second group, and everything else **([s-z] and underscore)** the third group. Within each group the letters are rotated left by ki positions in the message. Each group is rotated independently of the other two. Decrypting the message means doing a right rotation by ki positions within each group.

Input Format

All input strings comprises of only lowercase English alphabets and underscores.

Constraints

- Length of the string=[1,150]
- ki=[1,150] (i=1,2,3)

Output For each encrypted message, the output is a single line containing the decrypted string.

Sample Input 1

1 1 1

bktcluajs

Sample Output

ajsbktclu

2.2 Problem analysis

- take 3 inputs K1,K2,K3.
- take the encrypted string.
- Divide the alphabets and the under score according to the given group.
- Carry out the operations according to the values of K1,K2,K3.
- Print the decrypted message.

3 Appendix

3.1 Appendix-A : code for ps1

```
/*
# @FILE ps1.py
# @author Manas Ranjan Patro
# @date 27 September 2018

n, m = map(int, raw_input().split())
A = [raw_input() for x in range(n)]
```

```
M = []

def update(x, y):
    up = x; down = x; left = y; right = y;
    pos = set([(x, y)])
    while up >= 0 and down < n and left >=0 and right < m:
        costs = [A[elem[0]][elem[1]] == 'D' for elem in [(up, y)
    , (down, y), (x, left), (x, right)]]
        if sum(costs) > 0: break;
        for elem in [(up, y), (down, y), (x, left), (x, right)]:
            pos.add(elem)
        up-=1
        left-=1
        down+=1
        right+=1
    return pos

def switch(x):
    return {
        0: (0, 0),
        1: (1, 0),
    }.get(x, (x-4, 1))

for ii in range(n):
    for jj in range(m):
        if A[ii][jj] == 'S':
            M.append(update(ii, jj))

# for ii in M:
#     print ii
```

```
37
38 res = [(max([len(x) for x in M]), 0)]
39 for ii in range(len(M) - 1):
40     for jj in range(ii+1, len(M)):
41         len_mii = len(M[ii])
42         len_mjj = len(M[jj])
43
44         if len(M[ii].intersection(M[jj])) > 0:
45             continue
46
47         nres = [len_mii, len_mjj]
48         nres.sort(reverse=True)
49         res.append(tuple(nres))
50
51 res.sort(reverse=True)
52 # print res
53
54 if len(res) == 0: print 0, 0
55 else: print res[0][0], res[0][1]
```

3.2 Appendix-B : code for ps2

```
1 /*
2 # @FILE ps2.py
3 # @author Manas Ranjan Patro
4 # @date 27 September 2018
5 group = ['abcdefhi', 'jklmnopqr', 'stuvwxyz_']
6 K = map(int, raw_input().split())
7 A = raw_input()
8
9 data = ['', '', '']
```

```
ids = [[], [], []]
for ii in range(len(A)):
    for gid in range(3):
        if A[ii] in group[gid]:
            data[gid]+=A[ii]
            ids[gid].append(ii)
            break

def rotate(gid):
    k = K[gid] % len(ids[gid])
    data[gid] = data[gid][-k:] + data[gid][:-k]

res = ['x' for x in range(len(A))]
for gid in range(3):
    rotate(gid)
    for ii in range(len(ids[gid])):
        res[ids[gid][ii]] = data[gid][ii]

print ''.join(res)
```

3.3 Screenshots

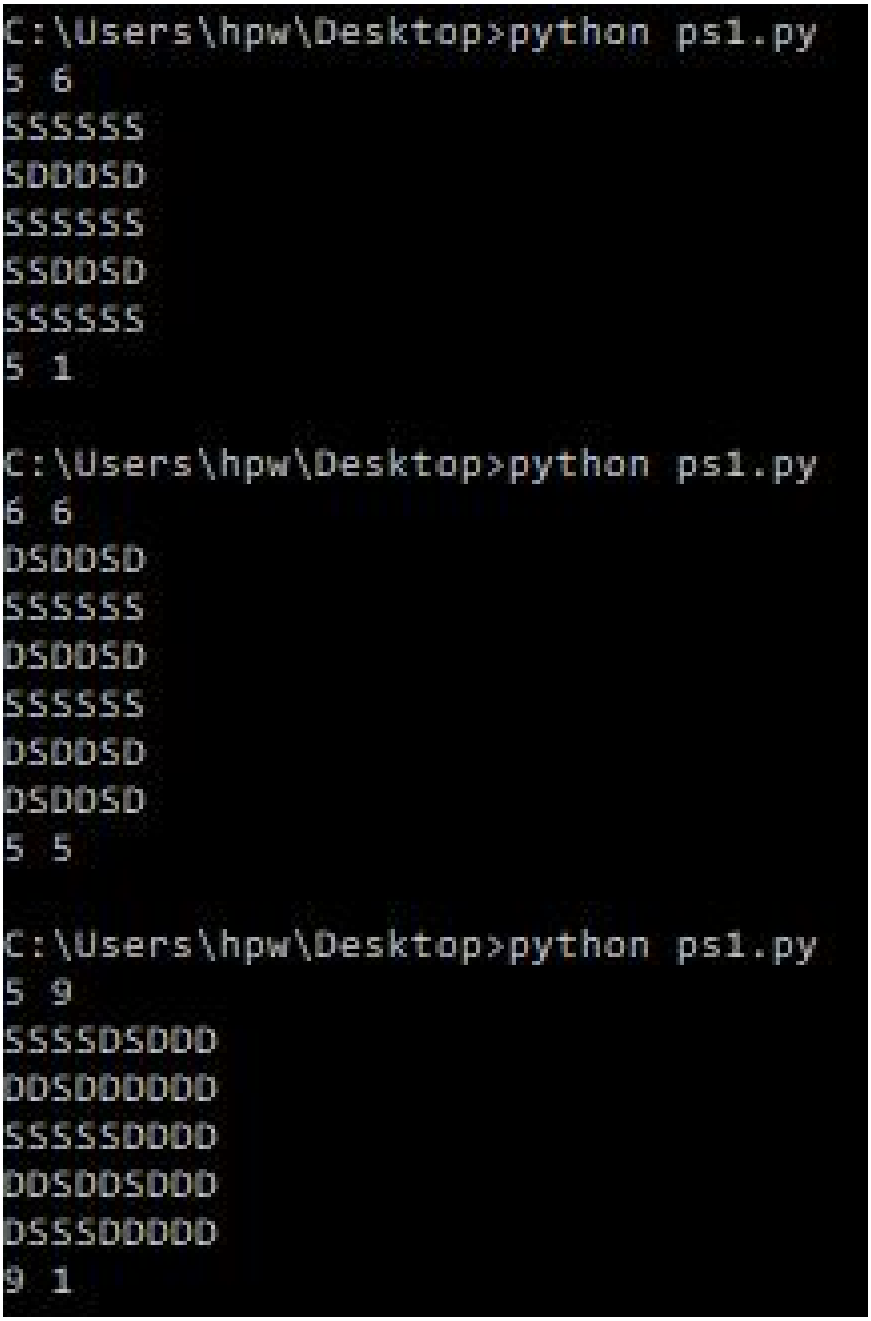


Figure 1: *Problem statement 1 output*

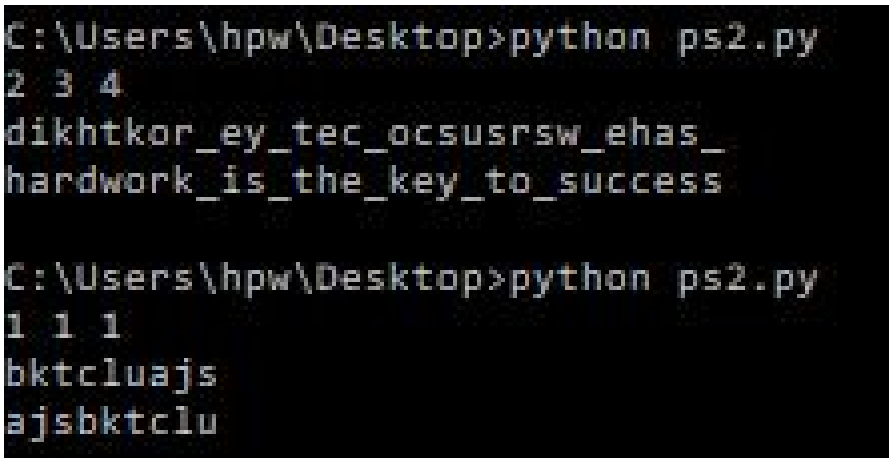


Figure 2: *Problem statement 2 output*