Assignment-8
ELP- 718 Telecom Software Laboratory

Anshuman Singh
2018JTM2004
2018-19

A report presented for the assignment on
Python and GitHub

Bharti School
of
Telecommunication Technology and Management
IIT DELHI, Delhi
September 27, 2018

Contents

List of Figures

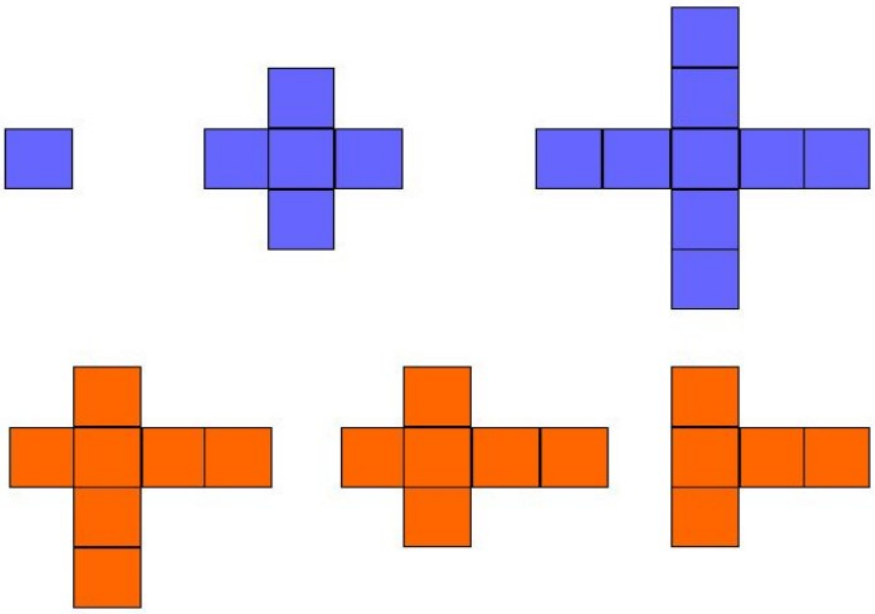
1 Problem Statement-1

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.

Figure 1: Valid and Invalid crosses



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.

1.2 Constraints

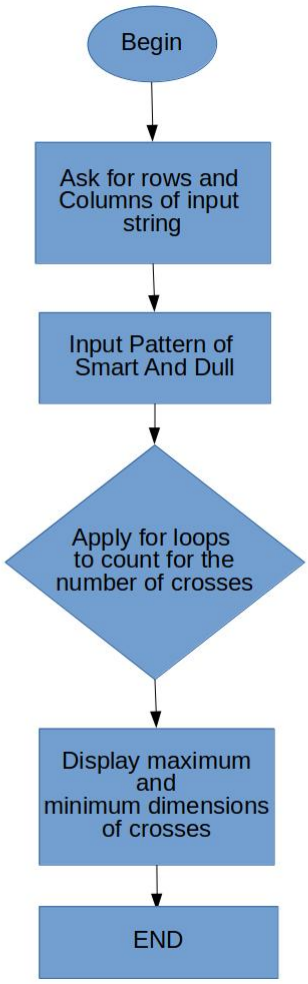
- $2 <= n <= 105$
- $2 <= m <= 105$

1.3 Algorithm and Implementation [1] [3]

- Create a file ps1.py
- First ask for no. of columns and rows in the given input pattern of S and D
- Then read the entered rows and columns in a two dimensional list.
- Using various for loops determine the number of crosses present and their size.
- Print the cross with maximum and minimum size.

1.4 Flow Chart

Figure 2: Flow Chart for Figure 1



1.5 Input and Output Format

- 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.

- Output Format:
  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)

  1. Sample Input 1

     5 6
     SSSSSS
     SDDDSD
     SSSSSS
     SSDDSD
     SSSSSS

  2. Sample Input 2

     6 6
     DSDDSD
     SSSSSS
     DSDDSD
     SSSSSS
     DSDDSD
     DSDDSD

1.6 Screenshots

Figure 3: Terminal Output of Problem 1

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.

2.2 Constraints

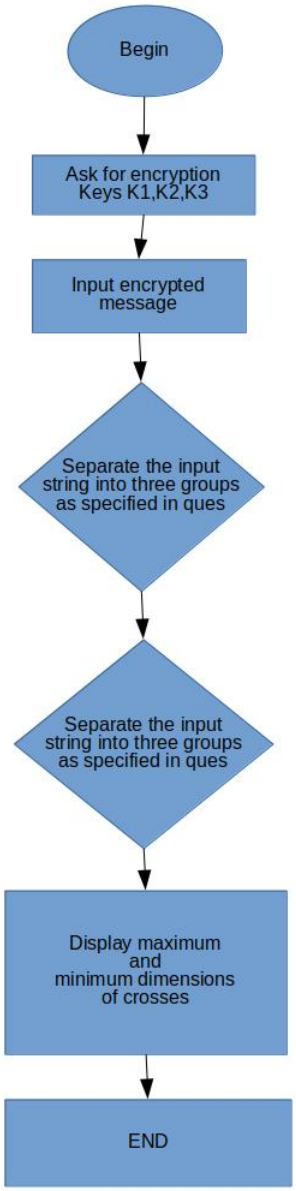
- $1 <= Lengthofthestring <= 150$
- $1 <= ki <= 150 (i = 1, 2, 3)$

2.3 Algorithm and Implementation [1] [2]

- Create a file ps2.py
- Ask for the encryption keys K1, K2, K3.
- Separate the input string into three parts as per the defined grouping.
- Rotate the groups by the specified keys.
- Then print the decrypted message by combining different groups after rotating it corresponding to the original bit position.

2.4 Flow Chart

Figure 4: Flow Chart of Problem 2



2.5 Input and Output Format

- **Input Format:** All input strings comprises of only lowercase English alphabets and underscores(_)

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

1. **Sample Input 1**

   2 3 4

   dikhtkor_ey_tec_ocsusrsw_ehas_

2. **Sample Output 1**

   hardwork_is_the_key_to_success

2.6 Screenshots

Figure 5: Terminal Output of Problem 2

3 Appendix

3.1 Code for ps1

```
#!/usr/bin/python3

"""Taking Inputs"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
row, column = list(map(int,input().split()))
print(row,column)
"""Declaring empty lists"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""

given = []
findings=[]
count = 0

"""Choosing one element and then findings number of S at its top,
   bottom, left and right"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""

for i in range(row):
    s = list(input())
    given = given + [s]
print(given)
```

```
for i in range(row):
    for j in range(column):
        if given[i][j]=="S":
                find_size=[]
                l,r,t,b=0,0,0,0
                x,y,w,z=1,1,1,1
                while j+y<column:
                    if given[i][j+y]!='S':
                        break
                    else:
                        r+=1
                        y+=1
                find_size.append(r)
                while j-x>0:
                    if given[i][j-x]!='S':
                        break
                    else:
                        l+=1
                        x+=1
                find_size.append(l)
                while i+w<row:
                    if given[i+w][j]!='S':
                        break
                    else:
                        t+=1
                        w+=1
                find_size.append(t)
                while i-w>0:
                    if given[i-w][j]!='S':
                        break
                    else:
                        b+=1
                        z+=1
                find_size.append(b)

                findings.append(min(find_size))
        count+=1
        print(count)
print(findings)
max_size = (max(findings)*4)+1
print(max_size,"1")
```

3.2 Code for ps2

```
#!/usr/bin/python3
"""Defining rotate function"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
def rotate(lst, x):
    new=lst.copy()
    for i in range(x):
        new.insert(0,new.pop(-1))
    return new

"""Taking Inputs"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
m1,m2,m3 = input().split(" ") #Input m1 m2 and m3
k1 = int(m1)
k2 = int(m2)
k3 = int(m3)
input_string = input("Enter the message to be decrypted : ")

"""Making different groups"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
group1 = "abcdefghi" #Making group1
group2 = "jklmnopqr" #Making group2
group3 = "stuvwxyz_" #Making group3

"""Updating elements from input into different lists."""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
p1,p2,p3=[],[],[]
for i in input_string:
    if i in group1:
        p1.append(i)
    if i in group2:
        p2.append(i)
    if i in group3:
        p3.append(i)
print(p1)
print(p2)
print(p3)
# for i in input_string:
#     if i in group2:
#         p2.append(i)
# for i in input_string:
#     if i in group3:
#         p3.append(i)
```

```python
# print(p1,p2,p3)
"""Rotate groups by specified amounts."""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""

rp1 = rotate(p1,k1) #rotating by specified amounts
rp2 = rotate(p2,k2) #rotating by specified amounts
rp3 = rotate(p3,k3) #rotating by specified amounts

# print(p1,p2,p3)
# print(rp1,rp2,rp3)
decrypted_message="" #Taking empty strings

""""~~~~~~~~~~~~~~~Writing the decrypted_message~~~~~~~~~~~~~~~~~"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
for loop in input_string:
    if (set(loop) & set(group1)):
        for ind,i in enumerate(p1):
            # print(ind,i)
            if i==loop:
                decrypted_message=decrypted_message+rp1[ind]
    if (set(loop) & set(group2)):
        for ind,i in enumerate(p2):
            if i==loop:
                decrypted_message=decrypted_message+rp2[ind]
    if (set(loop) & set(group3)):
        for ind,i in enumerate(p3):
            if i==loop:
                decrypted_message=decrypted_message+rp3[ind]

"""Printing decrypted message"""
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
print(decrypted_message)
"""~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"""
```

References

[1] BItBucket. *Become a git guru.* https://www.atlassian.com/git/tutorials.

[2] Python 3.7.1rc1 documentation. *Python Software Foundation.* https://docs.python.org/3/.

[3] Tutorials Point. *Python Tutorial.* https://www.tutorialspoint.com/python/.