Assignment-8

ELP - 718 Telecom Software Laboratory

Devendra Khatri
2018JTM2243
2018-2020

A report presented for the assignment on
Python & GitHub

Bharti School Of
Telecommunication Technology and Management
IIT Delhi
India
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.

1.2 specification

- Detect the "S".
- If the number of S is cross equal and 4 direction.
- find the odd number of square.
- Taking max1 and max2 number of squares.

1.3 Assumptions

- Commond line argument input.
- The 2-D matrix formation.

1.4 Program Structure

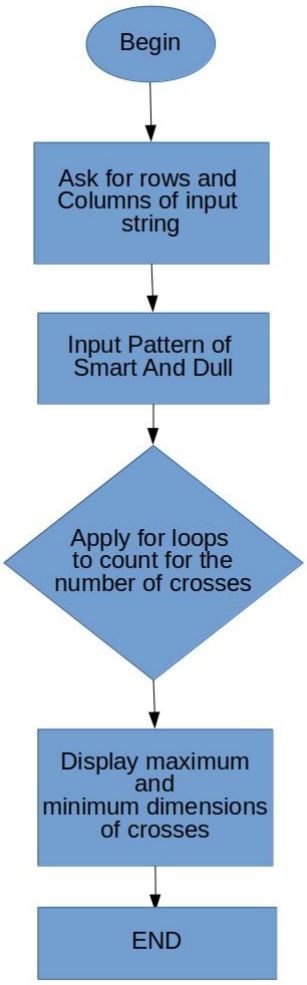


Figure 1: flowchart for ps1

1.5 Algorithm and Implementation

- Store the elements in 2-D list
- Using i,j parameter access each element.

- When a "s" detected we find the near by "S" and cunt increased.
- Select min value from them. and store in new list correspond the each.

1.6 Input and Output format

- **Input Format** 5 6
  SSSSSS
  SDDDSD
  SSSSSS
  SSDDSD
  SSSSSS
- **Output Format**.
  5 1

1.7 Test Cases

1. 6 6
   DSDDSD
   SSSSSS
   DSDDSD
   SSSSSS
   DSDDSD
   DSDDSD

1. 5 9
   SSSSDSDDD
   DDSDDDDDD
   SSSSSDDDD
   DDSDDSDDD
   DSSSDDDDD

1.8 Output

1. 9 1

1.9 Screen-shots

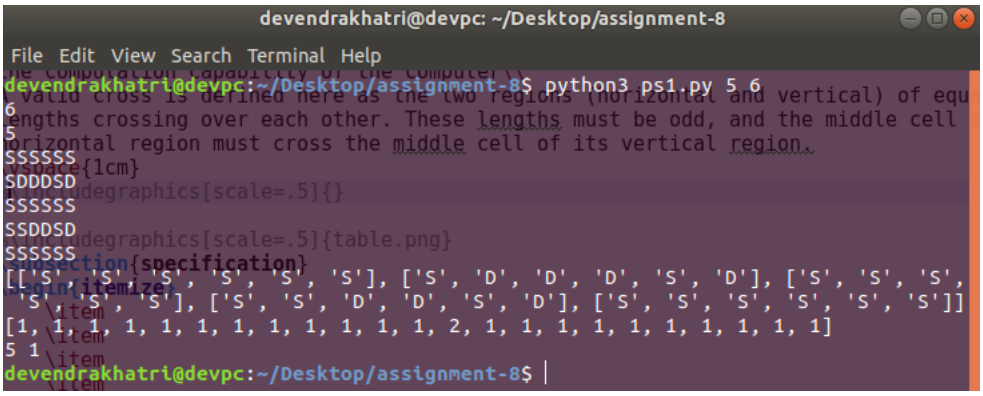


Figure 2: ps1- Decision 1

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.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.

- Enter the the rotation numbers
- Taking the inputs
-

2.2 Assumptions

- Taking the character from each string.
- divide the charater in three groups.
- match with the grouping.
- then rotate and update in new string

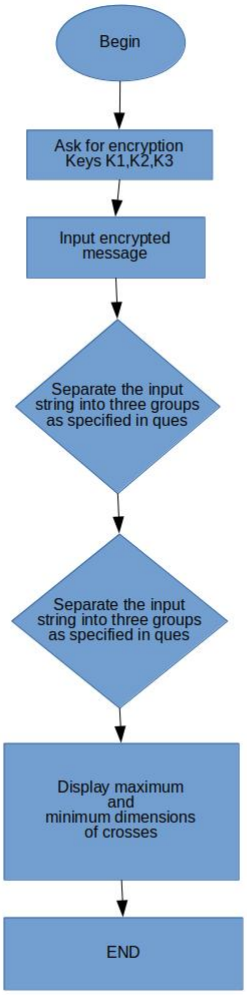
2.3 Program Structure

- Commond line argument input.
- Taking the character from each string.
- match with the grouping.

2.4 Algorithm and Implementation

- Created ps2.py to write a code for decryption algorithm
- key and encrypted string is taken from user.
- Decryption algorithm is applied as given in the problem statement.
- input Encrypted string along with a key gives Original Decrypted string

2.5 flowchart



2.6 Input Format and output format

- **Input Format**
  Input is key and Encrypted string.
- **Output Format**
  Output is Decrypted String

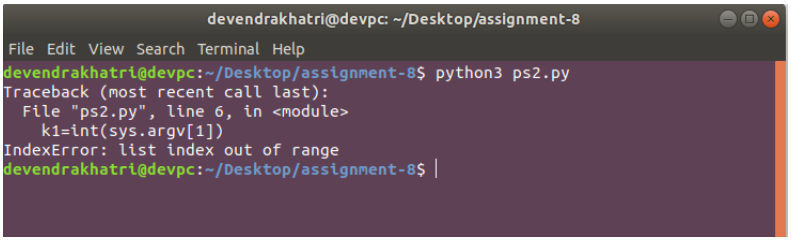


Figure 3: ps2

2.7 Test Cases

1. Input 1 1 1
   bktcluajs

2. Output
   ajsbktclu

2.8 Screen-shots

3 Appendix

3.1 Appendix-A: code for ps1

```
###### this is the first .py file ###########

####### write your code here ##########
import sys                          #include system module
n=int(sys.argv[1])                  #For commond line argument
m=int(sys.argv[2])
x=0
y=0
c=0
z=0
j=0
print(m)                            #check the inputs come
print(n)
lists=[]
list2=[]

if m>106 or m<3 :                   #Input condition
    print("invalid m value")
    exit()


if n>106 or n<3  :
    print("Invalid n value")
    exit()
s=0
a=[]
for i in range(n):
    s= list(input())
    a = a+ [s]


for i in range(n):
    for j in range(m):  #Checking the "S" avalibility and go in four
        direction
        if a[i][j]=="S":
            x=0
            y=0
```

```
                z=0
                c=0
                while i-x >= 0 :
                    if a[i-x][j]!="S":
                        break
                    x=x+1
                while i+y<n :
                    if a[i+y][j]!="S":
                        break
                    y=y+1
                while j-z>=0 :
                    if a[i][j-z]!="S":
                        break
                    z=z+1
                while j+c<m :
                    if a[i][j+c]!="S":
                        break
                    c=c+1
                list1=[x,y,z,c]
                e = min(list1)
                list2.append(e)
print(list2)
u=max(list2) #taking max value
del list2[list2.index(u)] #remove max value
v=max(list2)
 #now get 2nd max value

u=(u-1)*4+1
v=(v-1)*4+1
print(u,v)
```

3.2 Appendix-B:code for ps2

```
###### this is the second .py file ############

####### write your code here ##########
import sys

k1=int(sys.argv[1])
k2=int(sys.argv[2]) #commond line input
k3=int(sys.argv[3])
x=len(sys.argv)
input_string=0
list1
if k1>151 or k1<2 or k2>151 or k2<2 or k3>151 or k3<2: # Boundary
    condition
     print("Invalid input")
     exit()
input_string=input()
for i in input_string:  #Make a list of each caharacter
     list1.append(i)

print(list1)
```

References

[1] Python 3.7 documentation. https://docs.python.org/3/.

[2] Python3 tutorial. https://www.tutorialspoint.com/python3/index.htm.

[3] Run python in linux(ubuntu), Oct 2015. https://www.youtube.com/watch?v=3xp-ixFbDuE.

[4] Paul Barry. *Head First Python*. https://doc.lagout.org/programmation/python/Head%20First%20Python%2C%20First%20Edition%20%282010%29.pdf.

[5] Bitbucket. *Become a git guru*. https://www.atlassian.com/git/tutorials.

[6] Microwave Sam. How to Get Started with Github-Beginer. https://www.youtube.com/watch?v=73I5dRucCds&spfreload=5.