Assignment-8

ELP-780 Software Lab

Sakshi Agrawal
2017EET2291
2017-19

A report presented for the assignment on Python.

Department of Electrical Engineering
IIT DELHI
India

September 27, 2018

Contents

1 Problem Statement-1

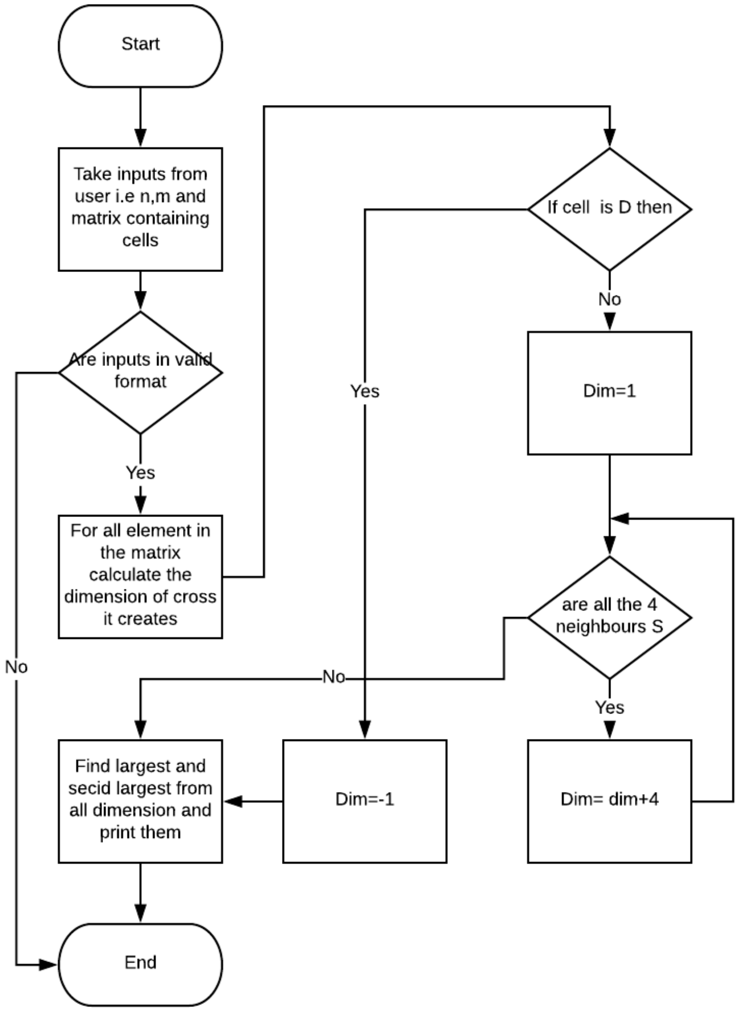
1.1 Problem statement

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

1.2 Assumptions

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

1.3 Program structure


Start
Take inputs from user i.e n,m and matrix containing cells
Are inputs in valid format
Yes
No
For all element in the matrix calculate the dimension of cross it creates
If cell is D then
No
Yes
Dim=1
are all the 4 neighbours S
Yes
No
Find largest and secid largest from all dimension and print them
Dim=-1
Dim= dim+4
End

1.4 Algorithm and Implementation

- Take input from user and check for valid inputs
- For each cell in the input find its dimension
- If the cell is 'D', its dimension is -1
- else if cell is 'S' its dimension is 1
- If all four neighbouring cell are also 'S' then increase dimension by 4
- Check for next 4 pair and continue until you get a 'D' or end of matrix
- From all dimensions find largest and second largest and print them.

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).

1.6 Test cases

- **Sample Input 1**

  ```
  5 6
  SSSSSS
  SDDDSD
  SSSSSS
  SSDDSD
  SSSSSS
  ```

  Sample Output 1

  ```
  5 1
  ```

- **Sample Input 2**

  ```
  6 6
  DSDDSD
  SSSSSS
  ```

DSDDSD

SSSSSS

DSDDSD

DSDDSD

Sample Output 2

5 5

- **Sample Input 3**

  5 9

  SSSSDSDDD

  DDSDDDDDD

  SSSSSDDDD

  DDSDDSDDD

  DSSSDDDDD

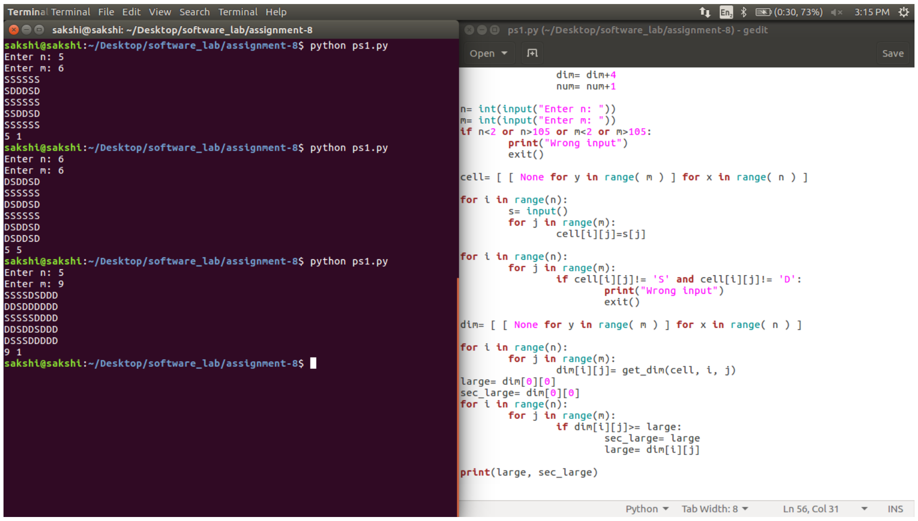
  Sample Output 3

  9 1

1.7 Difficulties/Issues faced

- In getting non overlapping crosses.

1.8 Screenshots



Terminal Terminal File Edit View Search Terminal Help
sakshi@sakshi: ~/Desktop/software_lab/assignment-8
(0:30, 73%) 3:15 PM
ps1.py (~/Desktop/software_lab/assignment-8) - gedit
Open
Save
Python
Tab Width: 8
Ln 56, Col 31
INS

```
sakshi@sakshi:~/Desktop/software_lab/assignment-8$ python ps1.py
Enter n: 5
Enter m: 6
SSSSSS
SDDDSD
SSSSSS
SSDDSD
SSSSSS
5 1
sakshi@sakshi:~/Desktop/software_lab/assignment-8$ python ps1.py
Enter n: 6
Enter m: 6
DSDDSD
SSSSSS
DSDDSD
SSSSSS
DSDDSD
DSDDSD
5 5
sakshi@sakshi:~/Desktop/software_lab/assignment-8$ python ps1.py
Enter n: 5
Enter m: 9
SSSSDSDDD
DDSDDDDDD
SSSSSDDDD
DDSDDSDDD
DSSSDDDDD
9 1
sakshi@sakshi:~/Desktop/software_lab/assignment-8$
```

```python
                dim= dim+4
                num= num+1

n= int(input("Enter n: "))
m= int(input("Enter m: "))
if n<2 or n>105 or m<2 or m>105:
        print("Wrong input")
        exit()

cell= [ [ None for y in range( m ) ] for x in range( n ) ]

for i in range(n):
        s= input()
        for j in range(m):
                cell[i][j]=s[j]

for i in range(n):
        for j in range(m):
                if cell[i][j]!= 'S' and cell[i][j]!= 'D':
                        print("Wrong input")
                        exit()

dim= [ [ None for y in range( m ) ] for x in range( n ) ]

for i in range(n):
        for j in range(m):
                dim[i][j]= get_dim(cell, i, j)
large= dim[0][0]
sec_large= dim[0][0]
for i in range(n):
        for j in range(m):
                if dim[i][j]>= large:
                        sec_large= large
                        large= dim[i][j]

print(large, sec_large)
```

2 Problem Statement-2

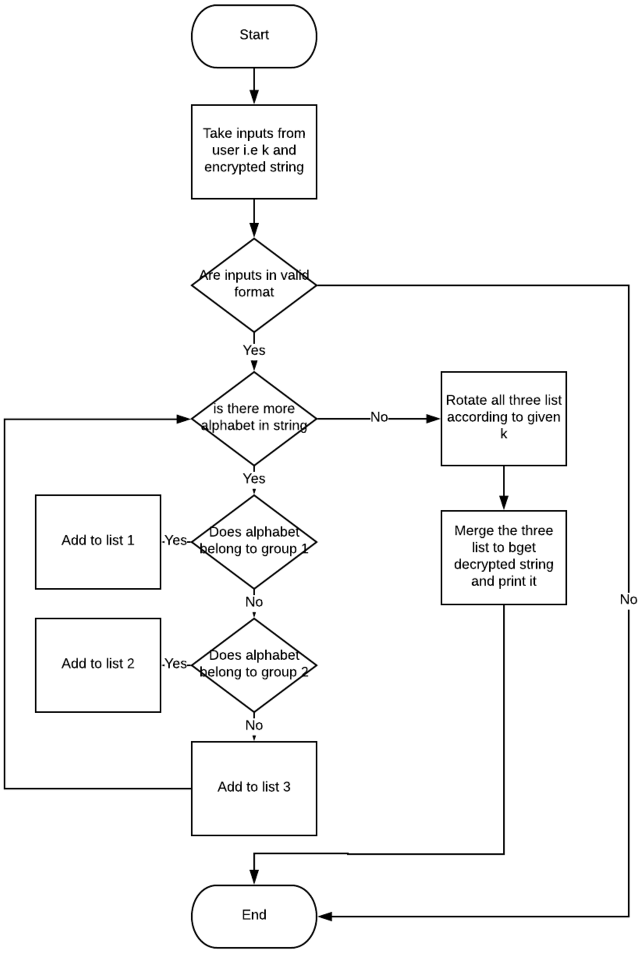
2.1 Problem statement

- Given an Encrypted data we have to decrypt it.
- 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 Assumptions

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

2.3 Program structure


Start
Take inputs from user i.e k and encrypted string
Are inputs in valid format
Yes
No
is there more alphabet in string
No
Rotate all three list according to given k
Yes
Add to list 1
Yes
Does alphabet belong to group 1
Merge the three list to bget decrypted string and print it
No
No
Add to list 2
Yes
Does alphabet belong to group 2
No
Add to list 3
End

2.4 Algorithm and Implementation

1. Take input from users and check for proper format
2. Break the entire string into three lists, each list containing elements of particular group.
3. Rotate all three list independently according to the ki
4. Merge the three list in the same order in which they were created.
5. Display the decrypted message.

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.

2.6 Test cases

- **Sample Input 1:**

  2 3 4

  dikhtkor_ey_tec_ocsusrsw_ehas_

  Sample Output 1 :

  hardwork_is_the_key_to_success

- **Sample Input 2:**

  1 1 1 bktcluajs

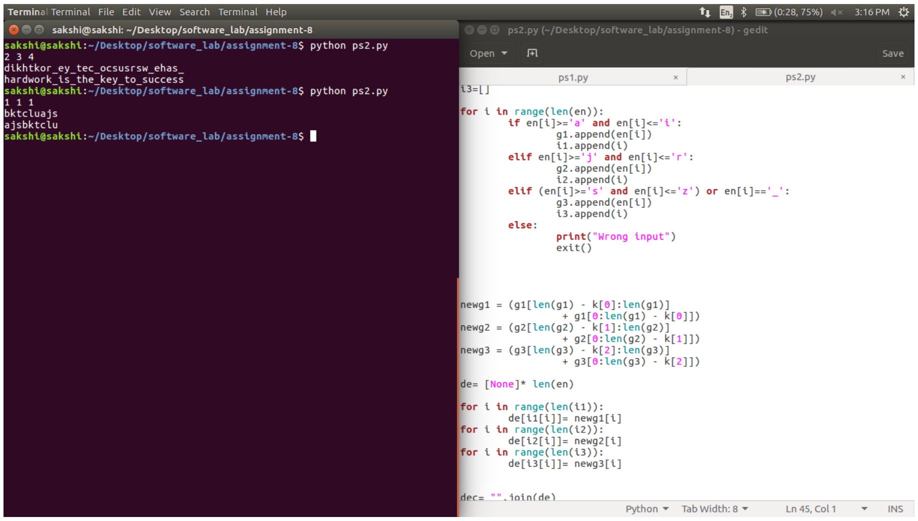
  Sample Output 2:

  ajsbktclu

2.7 Difficulties/Issues faced

- In case insensitive search.

2.8 Screenshots



Terminal Terminal File Edit View Search Terminal Help
sakshi@sakshi: ~/Desktop/software_lab/assignment-8
(0:28, 75%) 3:16 PM
ps2.py (~/Desktop/software_lab/assignment-8) - gedit
Open
Save
ps1.py
ps2.py
Python
Tab Width: 8
Ln 45, Col 1
INS

```
sakshi@sakshi:~/Desktop/software_lab/assignment-8$ python ps2.py
2 3 4
dikhtkor_ey_tec_ocsusrsw_ehas_
hardwork_is_the_key_to_success
sakshi@sakshi:~/Desktop/software_lab/assignment-8$ python ps2.py
1 1 1
bktcluajs
ajsbktclu
sakshi@sakshi:~/Desktop/software_lab/assignment-8$ 
```

```
i3=[]

for i in range(len(en)):
        if en[i]>='a' and en[i]<='i':
                g1.append(en[i])
                i1.append(i)
        elif en[i]>='j' and en[i]<='r':
                g2.append(en[i])
                i2.append(i)
        elif (en[i]>='s' and en[i]<='z') or en[i]=='_':
                g3.append(en[i])
                i3.append(i)
        else:
                print("Wrong input")
                exit()



newg1 = (g1[len(g1) - k[0]:len(g1)]
                 + g1[0:len(g1) - k[0]])
newg2 = (g2[len(g2) - k[1]:len(g2)]
                 + g2[0:len(g2) - k[1]])
newg3 = (g3[len(g3) - k[2]:len(g3)]
                 + g3[0:len(g3) - k[2]])

de= [None]* len(en)

for i in range(len(i1)):
        de[i1[i]]= newg1[i]
for i in range(len(i2)):
        de[i2[i]]= newg2[i]
for i in range(len(i3)):
        de[i3[i]]= newg3[i]


dec= "".join(de)
```

3 Appendix

3.1 Appendix-A Code for ps1

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

####### write your code here ##########



import numpy as np

def get_dim(cell, i, j):
  x= [-1, 0, 1, 0]
  y= [0, -1, 0, 1]
  dim=-1
  n, m = np.array(cell).shape
  if cell[i][j]== 'D':
    return dim
  else:
    dim=1
  num=1
  while True:
    for search in range(4):
      newi= i+num*x[search]
      newj= j+num*y[search]
      if newi<0 or newi >=n or newj<0 or newj >=m or cell[newi][newj]== 'D':
        return dim
    dim= dim+4
    num= num+1

n= int(input("Enter n: "))
m= int(input("Enter m: "))
if n<2 or n>105 or m<2 or m>105:
  print("Wrong input")
  exit()

cell= [ [ None for y in range( m ) ] for x in range( n ) ]

for i in range(n):
  s= input()
  for j in range(m):
    cell[i][j]=s[j]

for i in range(n):
  for j in range(m):
    if cell[i][j]!= 'S' and cell[i][j]!= 'D':
      print("Wrong input")
      exit()

dim= [ [ None for y in range( m ) ] for x in range( n ) ]

for i in range(n):
  for j in range(m):
    dim[i][j]= get_dim(cell, i, j)
large= dim[0][0]
```

```
sec_large= dim[0][0]
for i in range(n):
  for j in range(m):
    if dim[i][j]>= large:
      sec_large= large
      large= dim[i][j]

print(large, sec_large)
```

3.2 Appendix-B Code for ps2

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

####### write your code here ##########


import numpy as np

k= (input().split(" "))

if len(k)!=3:
  print("Wrong number of arguments")
  exit()
for i in range(3):
  k[i]= int(k[i])
  if k[i]<1 or k[i]>150:
    print("Wrong input")
    exit()

en= input()
if len(en) <1 or len(en)>150:
  andprint("Wrong input")
  exit()

g1=[]
g2=[]
g3=[]
i1=[]
i2=[]
i3=[]

for i in range(len(en)):
  if en[i]>='a' and en[i]<='i':
    g1.append(en[i])
    i1.append(i)
  elif en[i]>='j' and en[i]<='r':
    g2.append(en[i])
    i2.append(i)
  elif (en[i]>='s' and en[i]<='z') or en[i]=='_':
    g3.append(en[i])
    i3.append(i)
  else:
    print("Wrong input")
    exit()



```

```python

newg1 = (g1[len(g1) - k[0]:len(g1)]
                  + g1[0:len(g1) - k[0]])
newg2 = (g2[len(g2) - k[1]:len(g2)]
                  + g2[0:len(g2) - k[1]])
newg3 = (g3[len(g3) - k[2]:len(g3)]
                  + g3[0:len(g3) - k[2]])

de= [None]* len(en)

for i in range(len(i1)):
  de[i1[i]]= newg1[i]
for i in range(len(i2)):
  de[i2[i]]= newg2[i]
for i in range(len(i3)):
  de[i3[i]]= newg3[i]


dec= "".join(de)
print(dec)
```

References

[1] 'Python 3.7.1rc1 documentation'. Available: https://docs.python.org/3/.

[2] 'Git tutorial'. Available: https://www.atlassian.com/git/tutorials.

[3] 'Python List/Array Methods'. Available: https://www.w3schools.com/python/python_ref_list.asp.

[4] 'Python Dictionaries'. Available: https://www.w3schools.com/python/python_dictionaries.asp.

[5] 'Python Strings'. Available: https://www.w3schools.com/python/python_strings.asp.