Assignment - 8

ELP - 780 Software Lab

Prateek Arora
2017EET2841
2017-19

Python and Github

Computer technology
IIT Delhi
India

Sept 27, 2018

Contents

1 Problem Statement-1

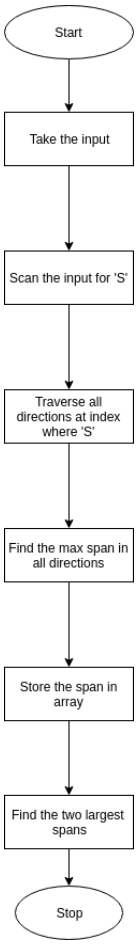
1.1 Problem Statement

- 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

- $2 \leq \mathrm{n} \leq 105$
- $2 \leq \mathrm{m} \leq 105$

1.3 Program Structure



1.4 Algorithm and Implementation

1. Take the input
2. Scan the input for 'S'.
3. Traverse all directions at index where 'S'.
4. Find the max span in all directions.
5. Store the span in array
6. Find the two largest spans

1.5 Input and Output format

Input Format

- "row" "col"
- "string matrix"

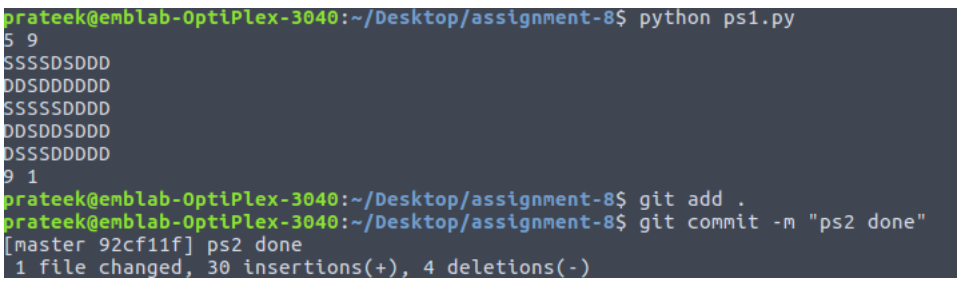
Output Format

- "max1" "max2"

1.6 Difficulties/Issues Faced

- Finding nonoverlapping crosses.

1.7 Screenshots



```
prateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$ python ps1.py
5 9
SSSSDSDDD
DDSDDDDDD
SSSSSDDDD
DDSDDSDDD
DSSSDDDDD
9 1
prateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$ git add .
prateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$ git commit -m "ps2 done"
[master 92cf11f] ps2 done
 1 file changed, 30 insertions(+), 4 deletions(-)
```

2 Problem Statement-2

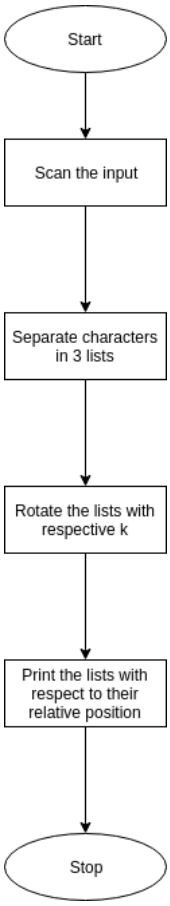
2.1 Problem Statement

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
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

- $1 \leq$ Length of the string ≤ 150
- $1 \leq$ ki ≤ 150 (i=1,2,3)

2.3 Program Structure



2.4 Algorithm and Implementation

1. Scan the input.
2. Separate characters in 3 lists.
3. Rotate the lists with respective k
4. Print the lists with respect to their relative position.

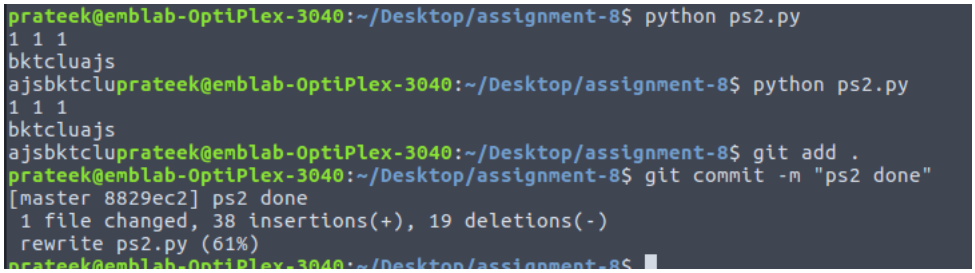
2.5 Input and Output format

Input
k1 k2 k3
"encrypted string"
Output
"decrypted string"

2.6 Difficulties/Issues Faced

- Rotating the list.[1]

2.7 Screenshots



```
prateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$ python ps2.py
1 1 1
bktcluajs
ajsbktcluprateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$ python ps2.py
1 1 1
bktcluajs
ajsbktcluprateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$ git add .
prateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$ git commit -m "ps2 done"
[master 8829ec2] ps2 done
 1 file changed, 38 insertions(+), 19 deletions(-)
 rewrite ps2.py (61%)
prateek@emblab-OptiPlex-3040:~/Desktop/assignment-8$
```

3 Appendix

3.1 Appendix A: Code for ps1

```
# scan row and col
row, col = input().split()
# scan the input
s = []
for i in range(int(row)):
    str = input()
    s.append(str)
# var to store various counts
count = 0
num = 1
flag = 0

maxlist = []
# scan the string and check for valid plus
for i in range(int(row)):
    for j in range(int(col)):
        flag = 0
        if s[i][j] == 'S':
            flag = 1
            num = 1
            count = 0
            while i - num >= 0 and i + num < int(row) and j - num >= 0 and j + n
                count += 1
                num += 1
        if flag == 1:
            maxlist.append(count*4 + 1)
# max1 and max2 for 2 largest
max1 = 0
max2 = 0
# find two largest

if (len(maxlist) == 0):
    print("0, 0")
else:
    for i in range(len(maxlist)):
        if maxlist[i] >= max1:
            max2 = max1
            max1 = maxlist[i]
    print (max1, max2)
```

3.2 Appendix B: Code for ps2

```python

# for all 3 rotations
k1, k2, k3 = input().split()
str = input()
# separating the characters
first = []
second = []
third = []
for i in str:
    if ord(i) >= ord('a') and ord(i) <= ord('i'):
        first.append(i)
    elif ord(i) >= ord('j') and ord(i) <= ord('r'):
        second.append(i)
    else:
        third.append(i)

#Roatating the lists

first_r = (first[-int(k1):] + first[:-int(k1)])
second_r = (second[-int(k2):] + second[:-int(k2)])
third_r = (third[-int(k3):] + third[:-int(k3)])

ctr1 = 0
ctr2 = 0
ctr3 = 0

#printing the result

for i in str:
    if ord(i) >= ord('a') and ord(i) <= ord('i'):
        print(first_r[ctr1],end = '')
        ctr1 += 1
    elif ord(i) >= ord('j') and ord(i) <= ord('r'):
        print(second_r[ctr2],end = '')
        ctr2 += 1
    else:
        print(third_r[ctr3],end = '')
        ctr3 += 1
```

References

[1] https://stackoverflow.com/questions/9457832/python-list-rotation