

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	2
1.1	Problem Statement	2
1.2	Assumptions	2
1.3	Program Structure	3
1.4	Algorithm and Implementation	3
1.5	Input and Output format	4
1.6	Difficulties/Issues Faced	4
1.7	Screenshots	4
2	Problem Statement-2	4
2.1	Problem Statement	4
2.2	Assumptions	4
2.3	Program Structure	5
2.4	Algorithm and Implementation	5
2.5	Input and Output format	5
2.6	Difficulties/Issues Faced	6
2.7	Screenshots	6
3	Appendix	7
3.1	Appendix A: Code for ps1	7
3.2	Appendix B: Code for ps2	7

1 Problem Statement-1

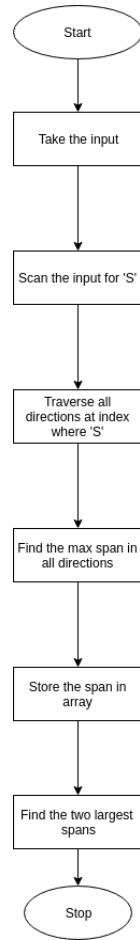
1.1 Problem Statement

- 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

- $2 \leq n \leq 105$
- $2 \leq m \leq 105$

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
DDSDSDDDD
DDSDSDDDD
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, k_1 , k_2 , and k_3 . The 26 letters of English and underscore are divided in three groups

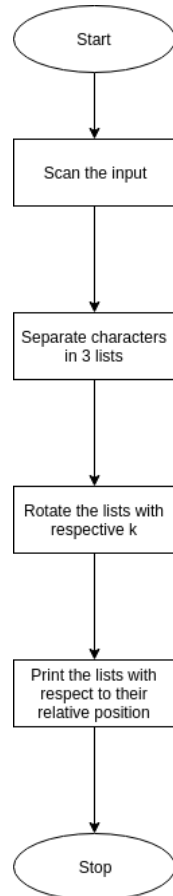
Within each group the letters are rotated left by k_i positions in the message.

Each group is rotated independently of the other two. Decrypting the message means doing a right rotation by k_i positions within each group.

2.2 Assumptions

- $1 \leq \text{Length of the string} \leq 150$
- $1 \leq k_i \leq 150$ ($i=1,2,3$)

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"

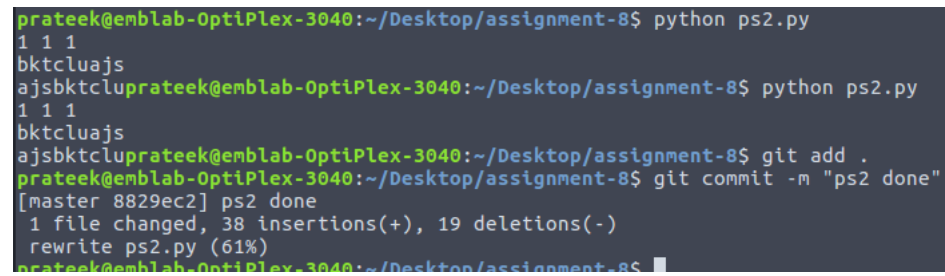
Output

"decrypted string"

2.6 Difficulties/Issues Faced

- Rotating the list.[1]

2.7 Screenshots

A terminal window screenshot showing a series of commands and their outputs. The user is in a directory ~/Desktop/assignment-8. They run 'python ps2.py', which outputs '1 1 1' and 'bktcluajs'. They then run 'python ps2.py' again, which outputs the same. Next, they run 'git add .' to stage the changes. Finally, they run 'git commit -m "ps2 done"', which shows the commit hash [master 8829ec2], the commit message 'ps2 done', and statistics: '1 file changed, 38 insertions(+), 19 deletions(-)', and 'rewrite ps2.py (61%)'.

```
prateek@emlab-OptiPlex-3040:~/Desktop/assignment-8$ python ps2.py
1 1 1
bktcluajs
ajsbktclu
prateek@emlab-OptiPlex-3040:~/Desktop/assignment-8$ python ps2.py
1 1 1
bktcluajs
ajsbktclu
prateek@emlab-OptiPlex-3040:~/Desktop/assignment-8$ git add .
prateek@emlab-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@emlab-OptiPlex-3040:~/Desktop/assignment-8$
```

3 Appendix

3.1 Appendix A: Code for ps1

```
1  # scan row and col
2  row, col = input().split()
3  # scan the input
4  s = []
5  for i in range(int(row)):
6      str = input()
7      s.append(str)
8  # var to store various counts
9  count = 0
10 num = 1
11 flag = 0
12
13 maxlist = []
14 # scan the string and check for valid plus
15 for i in range(int(row)):
16     for j in range(int(col)):
17         flag = 0
18         if s[i][j] == 'S':
19             flag = 1
20             num = 1
21             count = 0
22             while i - num >= 0 and i + num < int(row) and j - num >= 0 and j + num < int(col):
23                 count += 1
24                 num += 1
25             if flag == 1:
26                 maxlist.append(count*4 + 1)
27 # max1 and max2 for 2 largest
28 max1 = 0
29 max2 = 0
30 # find two largest
31
32 if (len(maxlist) == 0):
33     print("0,0")
34 else:
35     for i in range(len(maxlist)):
36         if maxlist[i] >= max1:
37             max2 = max1
38             max1 = maxlist[i]
39     print(max1, max2)
```

3.2 Appendix B: Code for ps2


```

1
2 # for all 3 rotations
3 k1, k2, k3 = input().split()
4 str = input()
5 # separating the characters
6 first = []
7 second = []
8 third = []
9 for i in str:
10     if ord(i) >= ord('a') and ord(i) <= ord('i'):
11         first.append(i)
12     elif ord(i) >= ord('j') and ord(i) <= ord('r'):
13         second.append(i)
14     else:
15         third.append(i)
16
17 #Roatating the lists
18
19 first_r = (first[-int(k1):] + first[: -int(k1)])
20 second_r = (second[-int(k2):] + second[: -int(k2)])
21 third_r = (third[-int(k3):] + third[: -int(k3)])
22
23 ctr1 = 0
24 ctr2 = 0
25 ctr3 = 0
26
27 #printing the result
28
29 for i in str:
30     if ord(i) >= ord('a') and ord(i) <= ord('i'):
31         print(first_r[ctr1],end = '')
32         ctr1 += 1
33     elif ord(i) >= ord('j') and ord(i) <= ord('r'):
34         print(second_r[ctr2],end = '')
35         ctr2 += 1
36     else:
37         print(third_r[ctr3],end = '')
38         ctr3 += 1

```

References

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