PROBLEM-SOLVING

(Solving Various Problems Using Python)

Summer Internship Report Submitted in partial fulfillment

of the requirement for the undergraduate degree of

Bachelor of Technology

In

Computer Science Engineering

By

K Sai Charan

221710313019

https://github.com/SaiCharan29/ProblemSolving-221710313019/tree/master

Under the Guidance of

_____

Assistant Professor

Department Of Computer Science Engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
June 2020

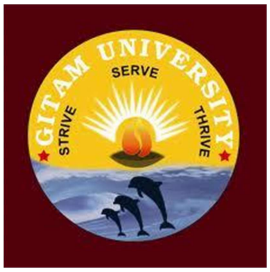
DECLARATION

I submitted this industrial training workshop entitled **"SOLVING VARIOUS PROBLEMS USING PYTHON**" to GITAM (Deemed to be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology**" in "**Computer Science Engineering**". I declare that it was carried out independently by me under the guidance of ___, Asst. Professor, GITAM (Deemed to be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD K Sai Charan

Date: 221710313019



GITAM (Deemed To Be University)
Hyderabad-502329
Telangana, India.

Date:

CERTIFICATE

This is to certify that the Industrial Training Report entitled **"SOLVING VARIOUS PROBLEMS USING PYTHON"** is being submitted by K Sai Charan (221710313019) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science Engineering** at GITAM (Deemed To Be University), Hyderabad during the academic year 2019-20.

It is faithful record work carried out by him at the **Computer Science Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

_____

Assistant Professor
Department of ECE

Dr. S Phani Kumar
Professor and HOD
Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank Dr. **N. Siva Prasad,** Pro Vice-Chancellor, GITAM Hyderabad and **Dr. N. Seetharamaiah,** Principal, GITAM Hyderabad

I would like to thank respected **Dr. S. Phani Kumar,** Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties ____ who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

K Sai Charan
221710313019

Table of Contents

1 Introduction

Problem-solving is the act of defining a problem; determining the cause of the problem; identifying, prioritizing, and selecting the alternatives for a solution; and implementing a solution.

The following eight problems require different approaches in order to solve the problem and get the desired solution.

1. Counting sheep - In this problem, we are going to print the last number that sheep will name before falling asleep.
2. Trouble sort - In this problem, we are going to print “OK” if the algorithm correctly sorts the list, or we are going to print the index of the first sorting error.
3. String rotation - In this problem, we are going to generate a string by performing left and right shift operations and check whether it is the substring of a given string or not.
4. Word break problem - In this problem, we are going to find out whether the input string can be broken down into a sequence of the words from the given list of words.
5. Bike Tour - In this problem, we are going to find the number of peaks Li encountered throughout the tour.
6. TestVita - In this problem, we are going to calculate the minimum number of hours required based on the dependencies.
7. Big City Skyline - In this problem, we are going to calculate the area common to all the buildings.
8. Mural - In this problem, we are going to find out the sum of all the beauty scores of the sections thanh painted.

I solved all the problems using python programming and used jupyter notebook as the interpreter to execute those problems/programs.

2 Problem 1
Counting Sheep

In this problem, we are going to print the last number that sheep will name before falling asleep.

2.1 Problem Statement:

Bleatrix Trotter the sheep has devised a strategy that helps her fall asleep faster. First, she picks a number N. Then she starts naming N, 2 × N, 3 × N, and so on. Whenever she names a number, she thinks about all of the digits in that number. She keeps track of which digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) she has seen at least once so far as part of any number she has named. Once she has seen each of the ten digits at least once, she will fall asleep.

Bleatrix must start with N and must always name (i + 1) × N directly after i × N. For example, suppose that Bleatrix picks N = 1692. She would count as follows:

N = 1692. Now she has seen the digits 1, 2, 6, and 9.
2N = 3384. Now she has seen the digits 1, 2, 3, 4, 6, 8, and 9.
3N = 5076. Now she has seen all ten digits and falls asleep.
What is the last number that sheep will name before falling asleep? If she will count forever, print INSOMNIA instead.

Input:

The first line of the input gives the number of test cases, T. T test cases follow. Each consists of one line with a single integer N, the number Bleatrix has chosen.

Output:

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the last number that Bleatrix will name before falling asleep, according to the rules described in the statement.

Sample Input:

5
0
1
2
11
1692

Sample Output:

Case #1: INSOMNIA
Case #2: 10
Case #3: 90
Case #4: 110
Case #5: 5076

Concepts Used To Solve:

loops, list comprehension, list remove( ) method, list len( ) method

Loops:

Loops allow us to execute a statement or group of statements multiple times.
Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
   statements

2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
   statements

List Comprehension:

List Comprehensions provide an elegant way to create new lists.
Syntax:
list2 = [expression for variable in list1 if(variable satisfies this condition)]

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items that are of different data types.
Syntax:
list_name = [value 1, value 2,..., value n]

remove( ) method:

remove( ) is an inbuilt function that removes a given object from the list. It does not return any value.

Syntax:
list.remove(obj)

Parameters:

The object to be removed from the list.

Return value:
The method does not return any value but removes the given object from the list.

len( ) method:

len() function is an inbuilt function which returns the length of the list.

Syntax:
len(list_name)

Parameters:

It takes a list as the parameter.

Return value:
It returns an integer which is the length of the string.

2.2 Coding:

```
#CountingSheep
for i in range(int(input())):
    n=int(input())
    if(n!=0):
        num=[i for i in range(10)]
        j=1
        while(len(num)!=0):
            p=j*n
            res=list(str(p))
            for k in res:
                if(int(k) in num):
                    num.remove(int(k))
            j+=1
        print("Case #",end="")
        print(i+1,p,sep=" :")
    else:
        print("Case #",end="")
        print(i+1,"INSOMNIA",sep=" :")
```

Fig 2.2.1

2.3 Output:

```
5
0
Case #1 :INSOMNIA
1
Case #2 :10
2
Case #3 :90
11
Case #4 :110
1692
Case #5 :5076
```

Fig 2.3.1

3 Problem 2
Trouble Sort

In this problem, we are going to print “OK” if the algorithm correctly sorts the list, or we are going to print the index of the first sorting error.

3.1 Problem Statement:

Deep in Code Jam's secret algorithm labs, we devote countless hours to wrestling with one of the most complex problems of our time: efficiently sorting a list of integers into non-decreasing order. We have taken a careful look at the classic bubble sort algorithm, and we are pleased to announce a new variant.

The basic operation of the standard bubble sort algorithm is to examine a pair of adjacent numbers and reverse that pair if the left number is larger than the right number. But our algorithm examines a group of three adjacent numbers, and if the leftmost number is larger than the rightmost number, it reverses that entire group. Because our algorithm is a "triplet bubble sort", we have named it Trouble Sort for short.

For example, for L = 5 6 6 4 3, Trouble Sort would proceed as follows:

First pass:
inspect 5 6 6, do nothing: 5 6 6 4 3
inspect 6 6 4, see that 6 > 4, reverse the triplet: 5 4 6 6 3
inspect 6 6 3, see that 6 > 3, reverse the triplet: 5 4 3 6 6

Second pass:
inspect 5 4 3, see that 5 > 3, reverse the triplet: 3 4 5 6 6
inspect 4 5 6, do nothing: 3 4 5 6 6
inspect 5 6 6, do nothing: 3 4 5 6 6

Then the third pass inspects the three triplets and does nothing, so the algorithm terminates.
We were looking forward to presenting Trouble Sort at the Special Interest Group in Sorting conference in Hawaii, but one of our interns has just pointed out a problem: it is possible that Trouble Sort does not correctly sort the list! Consider the list 8 9 7, for example.
We need your help with some further research. Given a list of N integers, determine whether Trouble Sort will successfully sort the list into non-decreasing order. If it will not, find the index (counting starting from 0) of the first sorting error after the algorithm has finished: that is, the first value that is larger than the value that comes directly after it when the algorithm is done.

Input:

The first line of the input gives the number of test cases, T. T test cases follow. Each test case consists of two lines: one line with an integer N, the number of values in the list, and then another line with N integers Vi, the list of values.

Output:

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is OK if Trouble Sort correctly sorts the list, or the index (counting starting from 0) of the first sorting error, as described above.

Sample Input:

2
5
5 6 8 4 3
3
8 9 7

Sample Output:

Case #1: OK
Case #2: 1

Concepts Used To Solve:

functions, loops, lists, list slicing, list sort( ) method, map( ) function

Loops:

Loops allow us to execute a statement or group of statements multiple times.
Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
       statements

2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
       statements

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items that are of different data types.

Syntax:

list_name = [value 1, value 2... value n]

sort( ) method:

The sort function can be used to sort a list.

Syntax:

list_name.sort( )

Parameters:

It doesn't require any parameters.

Return value:

The method does not return any value but sorts the list.

map( ) function:

map( ) function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple, etc.)

Syntax:

map(fun, iter)

Parameters:

1. **fun:** It is a function to which map passes each element of a given iterable.
2. **iter:** It is iterable which is to be mapped.

Return value:

Returns a list of the results after applying the given function to each item of a given iterable (list, tuple, etc.)

Functions:

A function is a reusable piece of code that performs a specific task.

Syntax:

```
def functionname( parameters ):
    statements
    return statement
```

List slicing:

To print a specific range of elements from the list, we use Slice operation. Slice operation is performed on Lists with the use of a colon (:).

Syntax:

To print elements from beginning to a range - [: Index]

To print elements from the end - [:-Index]

To print elements from specific Index till the end - [Index:]

To print elements within a range - [Start Index: End Index]

To print the whole List with the use of slicing operation - [:]

To print the whole List in reverse order - [::-1]

3.2 Coding:

```
#Trouble Sort
def findError(li):
    for i in range(1,len(li)-1):
        if(not(li[i]>li[i-1] and li[i]<li[i+1])):
            return i
def issort(li):
    li1=li[:]
    li1.sort()
    if(li1==li):
        return True
    else:
        return False
def TroubleSort(li):
    done=False
    while not done:
        done = True
        for i in range(len(li)-2):
            if li[i]>li[i+2]:
                done=False
                a=li[i+2]
                li[i+2]=li[i]
                li[i]=a
    return li
for i in range(int(input())):
    a=int(input())
    l=list(map(int,input().split()))
    l=TroubleSort(l)
    if(issort(l)):
        print("Case #",end="")
        print(i+1,"OK",sep=" :")
    else:
        print("Case #",end="")
        print(i+1,findError(l),sep=" :")
```

Fig 3.2.1

3.3 Output:

```
2
5
5 6 8 4 3
Case #1 :OK
3
8 9 7
Case #2 :1
```

Fig 3.3.1

4 Problem 3
String Rotation

In this problem, we are going to generate a string by performing left and right shift operations and check whether it is the substring of a given string or not.

4.1 Problem Statement:

Rotate a given String in the specified direction by specified magnitude.
After each rotation makes a note of the first character of the rotated String After all rotations are performed the accumulated first character as noted previously will form another string, say FIRSTCHARSTRING.
Check If FIRSTCHARSTRING is an Anagram of any substring of the original string.
If yes, print "YES" otherwise "NO".

Input:

The first line contains the original string s. The second line contains a single integer q. The ith of the next q lines contains character d[i] denoting direction and integer r[i] denoting the magnitude.
Constraints
1 <= Length of original string <= 30
1<= q <= 10

Output:

YES or NO

Sample Input:

carrace
3
L 2
R 2
L 3

Sample Output:

NO

Concepts Used To Solve:

strings, lists, functions, string slicing, loops, string find( ) method

Strings:

A string is a sequence of characters. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double-quotes.

Syntax:

string_name = “string”

Loops:

Loops allow us to execute a statement or group of statements multiple times.

Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
   statements
2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
   statements

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items that are of different data types.

Syntax:

list_name = [value 1, value 2,..., value n]

String slicing:

Slicing is about obtaining a sub-string from the given string by slicing it respectively from start to end.

Syntax:

string[start:end:step]

Functions:

A function is a reusable piece of code that performs a specific task.

Syntax:

```
def functionname( parameters ):
    statements
    return statement
```

find( ) function:

The find() method returns the lowest index of the substring if it is found in a given string. If it is not found then it returns -1.

Syntax:

```
str.find(sub,start,end)
```

Parameters:

1. **sub:** It's the substring which needs to be searched in the given string.
2. **start:** Starting position where sub needs to be checked within the string.
3. **end:** Ending position where suffix is needed to be checked within the string.

Return value:

returns the lowest index of the substring if it is found in a given string. If it's not found then it returns -1.

4.2 Coding:

```
s=input()
a=int(input())
sttt=s
stri=""
def lr(s):
    st=s[1:len(s)]+s[0]
    return st
def rr(s):
    st=s[len(s)-1]+s[0:len(s)-1]
    return st
for qwerty in range(a):
    a=list(input().split())
    for i in range(int(a[1])):
        if(a[0]=='L'):
            s=lr(s)
        elif(a[0]=='R'):
            s=rr(s)
    stri+=s[0]
if(s.find(stri)!=-1):
    print("YES")
else:
    print("NO")
```

Fig 4.2.1

4.3 Output:

```
carrace
3
L 2
R 2
L 3
NO
```

Fig 4.3.1

5 Problem 4
Word break problem

In this problem, we are going to find out whether the input string can be broken down into a sequence of the words from the given list of words.

5.1 Problem Statement:

Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words. See the following examples for more details.
Consider the following dictionary { i, like, sam, sung, samsung, mobile, ice, cream, icecream, man, go, mango}

Input: ilike
Output: Yes
Explanation:
The string can be segmented as "i like".

Input: ilikesamsung
Output: Yes
Explanation:
The string can be segmented as "i like samsung" or "i like sam sung".

Input:

The first line is integer T denoting the number of test cases. 1<=T<=100.
Every test case has 3 lines.
The first line is the N number of words in dictionary.1<=N<=12.
Second-line contains N strings denoting the words of the dictionary.
The length of each word is less than 15.
The third line contains a string S of length less than 1000.

Output:

Print 1 is possible to break words, else print 0.

Concepts Used To Solve:

Lists, loops, strings, list reverse( ) method, string replace( ) method

Loops:

Loops allow us to execute a statement or group of statements multiple times.
Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
       statements

2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
       statements

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items that are of different data types.

Syntax:
list_name = [value 1, value 2,..., value n]

replace( ) method:

replace( ) is an inbuilt function that returns a copy of the string where all occurrences of a substring are replaced with another substring.

Syntax:
str.replace(old, new, count)

Parameters:

1. **old –** old substring you want to replace.
2. **new –** new substring which would replace the old substring.
3. **count –** the number of times you want to replace the old substring with the new substring.

Return value:
It returns a copy of the string where all occurrences of a substring is replaced with another substring.

Strings:

A string is a sequence of characters. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double-quotes.

Syntax:
string_name = “string”

reverse( ) method:

reverse() is an inbuilt method that reverses objects of the list in place.

Syntax:
list_name.reverse()

Parameters:

It doesn't require any parameters.

Return value:
The reverse() method does not return any value, but reverses the list.

Sample Input:

2
12
i like sam sung samsung mobile ice cream icecream man go mango
ilike
12
i like sam sung samsung mobile ice cream icecream man go mango
idontlike

Sample Output:

1 0

5.2 Coding:

```
for i in range(int(input())):
    m=int(input())
    l=list(input().split())
    l=sorted(l,key=len)
    l.reverse()
    s=input()
    for i in l:
        if(i in s):
            s=s.replace(i,'')
    if(len(s)==0):
        print(1)
    else:
        print(0)
```

Fig 5.2.1

5.3 Output:

```
2
12
i like sam sung samsung mobile ice cream icecream man go mango
ilike
1
12
i like sam sung samsung mobile ice cream icecream man go mango
idontlike
0
```

Fig 5.3.1

6 Problem 5
Bike Tour

In this problem, we are going to find the number of peaks Li encountered throughout the tour.

6.1 Problem Statement:

Li has planned a bike tour through the mountains of Switzerland. His tour consists of N checkpoints, numbered from 1 to N in the order he will visit them. The ith checkpoint has a height of H_i.

A checkpoint is a *peak* if:

- It is not the 1st checkpoint or the N-th checkpoint, and
- The height of the checkpoint is *strictly greater than* the checkpoint immediately before it and the checkpoint immediately after it.

Please help Li find out the number of peaks.

Input:

The first line of the input gives the number of test cases, T. T test cases follow. Each test case begins with a line containing the integer N. The second line contains N integers. The ith integer is H_i.

Output:

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the number of peaks in Li's bike tour.

Sample Input:

```
4
3
10 20 14
4
7 7 7 7
5
10 90 20 90 10
3
10 3 10
```

Sample Output:

```
Case #1: 1
Case #2: 0
Case #3: 2
Case #4: 0
```

Concepts Used To Solve:

loops, lists, map( ) function

Loops:

Loops allows us to execute a statement or group of statements multiple times.

Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
   statements
2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
   statements

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items that are of different data types.

Syntax:
list_name = [value 1, value 2,..., value n]

map( ) function:

map( ) function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

Syntax:
map(fun, iter)

Parameters:

1. **fun:** It is a function to which map passes each element of a given iterable.
2. **iter:** It is an iterable which is to be mapped.

Return value:
Returns a list of the results after applying the given function to each item of a given iterable (list, tuple, etc.)

6.2 Coding:

```
#BikeTour
for j in range(int(input())):
    c=0
    n=int(input())
    l=list(map(int,input().split()))
    for i in range(1,len(l)-1):
        if(l[i+1]<l[i] and l[i-1]<l[i]):
            c+=1
    print("Case #",end="")
    print(j+1,c,sep=" :")
```

Fig 6.2.1

6.3 Output:

```
4
3
10 20 14
Case #1 :1
4
7 7 7 7
Case #2 :0
5
10 90 20 90 10
Case #3 :2
3
10 3 10
Case #4 :0
```

Fig 6.3.1

7 Problem 6
TestVita

In this problem, we are going to calculate the minimum number of hours required based on the dependencies.

7.1 Problem Statement:

TCS is working on a new project called "TestVita". There are N modules in the project. Each module (i) has completion time denoted in the number of hours (Hi) and may depend on other modules. If Module x depends on Module y then one needs to complete y before x. As a Project manager, you are asked to deliver the project as early as possible. Provide an estimation of the amount of time required to complete the project.

Input:

First-line contains T, number of test cases.

For each test case:
1. First-line contains N, the number of modules.
2. Next N lines, each contains:
(i) Module ID
(Hi) Number of hours it takes to complete the module
(D) Set of module ids that depend on integers delimited by space.

Output:

Output the minimum number of hours required to deliver the project.

Sample Input:

1
5
1 5
2 6 1
3 3 2
4 2 3
5 1 3

Sample Output:

16

Concepts Used To Solve:

loops, lists, list append( ) method

Loops:

Loops allows us to execute a statement or group of statements multiple times.

Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
   statements
2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
   statements

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items which are of different data types.

Syntax:

list_name = [value 1, value 2,..., value n]

append( ) method:

Adds its argument as a single element to the end of a list.

Syntax:

list_name.append(element to be added)

Parameters:

Element to be added.

Return value:

The append() method does not return any value but appends the element to the list.

7.2 Coding:

```
#TestVita
for qwerty in range(int(input())):
    m=[]
    nt=[]
    dep=[]
    n=int(input())
    for i in range(n):
        l=input().split()
        m.append(l[0])
        nt.append(l[1])
        dep.append(l[2])
    for i in range(n):
        for j in range(n):
            if(i!=j):
                if(int(dep[i])==int(dep[j])):
                    if(int(nt[i])>int(nt[j])):
                        nt[j]=0
                    else:
                        nt[i]=0
    s=0
    for i in range(n):
        s+=int(nt[i])
    print(s)
```

Fig 7.2.1

7.3 Output:

```
1
5
1 5 0
2 6 1
3 3 2
4 2 3
5 1 3
16
```

Fig 7.3.1

8 Problem 7
Big City Skyline

In this problem, we are going to calculate the largest area which is common to all the buildings.

8.1 Problem Statement:

You've just moved to the Big City so you could start work at the newest Google office, Google BC, and the one thing that interests you most is the beautiful skyline. You're sitting on a hillside ten miles away, looking at the big rectangular buildings that make up the city, and you think back to your childhood.

When you were a child, you used to make toy cities out of rectangular blocks. Each building could be made from multiple blocks, and each block could be part of multiple buildings. Looking at the skyline, you wonder: what is the biggest possible block that could be part of the Big City's skyline?

Write a program that takes the description of the skyline as an input, and gives the area of the maximum area rectangle as output. This program should take less than 4 minutes to run on a 2GHz computer with 512MB of RAM, even for the biggest input size we specify.

Input:

The city is made out of rectangular buildings, all next to each other. The input will consist of an integer N, the number of buildings, followed by a series of N number pairs (wi, hi), indicating the width and height of each building in order from left to right. The buildings' heights will be less than 100,000,000, and their widths will be less than 1000. Easy: $0 < N < 1,000$ Hard: $0 < N < 10,000,000$

Output:

A single number: the area of the largest possible block. (Here's the beautiful skyline of the Big City.)

Sample Input:

5
5 7 1 20 3 5 6 3 2 10

Sample Output:

51

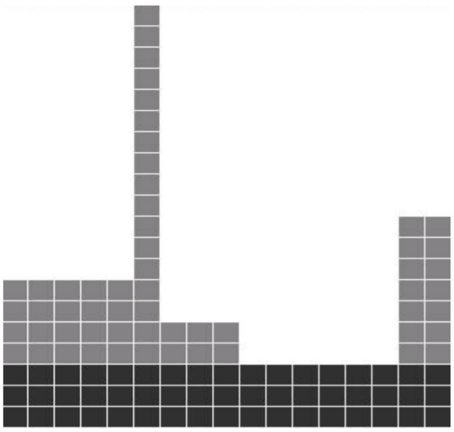


Fig 8.1.1

The biggest block is shaded black in the figure above.

Concepts Used To Solve:

lists, loops, map( ) function

Loops:

Loops allow us to execute a statement or group of statements multiple times.

Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
   statements
2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
   statements

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items that are of different data types.

Syntax:
list_name = [value 1, value 2,..., value n]

map( ) function:

map( ) function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple, etc.)

Syntax:

map(fun, iter)

Parameters:

1. **fun:** It is a function to which map passes each element of a given iterable.
2. **iter:** It is an iterable which is to be mapped.

Return value:

Returns a list of the results after applying the given function to each item of a given iterable (list, tuple, etc.)

8.2 Coding:

```
#Big City Skyline
a=int(input())
li=list(map(int,input().split()))
s=0
m=li[1]
for i in range(len(li)):
    if(i%2==0):
        s+=li[i]
    else:
        if(m>li[i]):
            m=li[i]
print(s*m)
```

Fig 8.2.1

8.3 Output:

```
5
5 7 1 20 3 5 6 3 2 10
51
```

Fig 8.3.1

9 Problem 8:
Mural

In this problem, we are going to find out the sum of all the beauty scores of the sections thanh painted.

9.1 Problem Statement:

Thanh wants to paint a wonderful mural on a wall that is N sections long. Each section of the wall has a beauty score, which indicates how beautiful it will look if it is painted. Unfortunately, the wall is starting to crumble due to a recent flood, so he will need to work fast!

At the beginning of each day, Thanh will paint one of the sections of the wall. On the first day, he is free to paint any section he likes. On each subsequent day, he must paint a new section that is next to a section he has already painted since he does not want to split up the mural.

At the end of each day, one section of the wall will be destroyed. It is always a section of wall that is adjacent to only one other section and is unpainted (Thanh is using waterproof paint, so painted sections can't be destroyed).

The total beauty of Thanh's mural will be equal to the sum of the beauty scores of the sections he has painted. Thanh would like to guarantee that, no matter how the wall is destroyed, he can still achieve a total beauty of at least B. What's the maximum value of B for which he can make this guarantee?

Input:

The first line of the input gives the number of test cases, T. T test cases follow. Each test case starts with a line containing an integer N. Then, another line follows containing a string of N digits from 0 to 9. The i-th digit represents the beauty score of the i-th section of the wall.

Output:

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the maximum beauty score that Thanh can guarantee that he can achieve, as described above.

Concepts Used To Solve:

lists, loops, list sum( ) method, list slicing

Loops:

Loops allow us to execute a statement or group of statements multiple times.

Python provides us two types of loops. They are

1. **While loop:**
   While loop executes a set of statements as long as a condition is true.
   Syntax:
   while expression:
       statements
2. **For loop:**
   For loop is used when a block of code needs to be repeated a fixed number of times.
   Syntax:
   for variable in sequence:
       statements

Lists:

The list is a most versatile data type which can be written as a list of comma-separated values between square brackets. The list can have items that are of different data types.

Syntax:
list_name = [value 1, value 2,..., value n]

sum( ) function:

sum( ) is an inbuilt function which sums up the numbers in the list.

Syntax:
sum(list_name)

Parameters:

It accepts the list as a parameter.

Return value:
Returns the sum of all the numbers in the list.

Sample Input:

4
4
1332
4
9583
3
616
10
1029384756

Sample Output:

Case #1: 6
Case #2: 14
Case #3: 7
Case #4: 31

9.2 Coding:

```
for qwerty in range(int(input())):
    n=int(input())
    if(n%2==0):
        m=n//2
    else:
        m=n//2+1
    li=int(input())
    li=list(str(li))
    li1=[]
    for i in range(len(li)):
        li[i]=int(li[i])
    ma=li[0]
    i=0
    while(i<n-m):
        s=sum(li[i:i+m])
        if(ma<s):
            ma=s
        i+=1
    print("Case #",end="")
    print(qwerty+1,ma,sep=" :")
```

Fig 9.2.1

9.3 Output:

```
4
4
1332
Case #1 :6
4
9583
Case #2 :14
3
616
Case #3 :7
10
1029384756
Case #4 :31
```

Fig 9.3.1

10 Software Requirements

10.1 Hardware Requirements:

This project can be executed in any system or android phone without prior to any platform. We can use any online compiler and interpreter.

10.2 Software Requirements:

There are two ways to execute this project

1) Online compilers
2) Software applications for execution i.e.., IDE's (DEV C++, ANACONDA…..)

Online Compilers require only an internet connection. We have many free compilers with which we can code.

IDEs need to be installed based on the user's system specification. These help us to completely execute the project. These softwares are based on the platforms. Many of them are free of cost.

Based on availability we can use anyone.

Some of the best Online Compilers are:
OnlineGDB, Tutorials point, Programiz, Code chef and many more.

Some of the best Softwares for execution(IDE's) are:

For Python:
Jupyter Notebook, Python IDLE, Atom, etc.

11 References

1. Counting Sheep – https://codingcompetitions.withgoogle.com/codejam
2. Trouble Sort – https://codingcompetitions.withgoogle.com/codejam
3. String Rotation – https://www.tcscodevita.com/
4. Word Break Problem – https://www.geeksforgeeks.org/
5. Bike Tour – https://codingcompetitions.withgoogle.com/kickstart
6. Test Vita – https://www.tcscodevita.com/
7. Big City Skyline – https://static.googleusercontent.com/media/services.google.com/en//blog_resources/Google_CodeJam_Practice.pdf
8. Mural – https://codingcompetitions.withgoogle.com/kickstart