

LAB SESSION 07:

Date of the Session: __/__/__

Time of the Session: __ to __

Pre-Lab:

- 1) A student named Satish is eagerly waiting to attend tomorrow's class. As he searched the concepts of tomorrow's lecture in the course handout and started solving the problems, in the middle he was stroked in the concept of strings because he was poor in this concept so help him so solve the problem, given two strings str1 and str2 and below operations that can performed on str1. Find minimum number of edits (operations) required to convert 'str1' into 'str2'.

1. Insert
2. Remove
3. Replace

Input

str1 = "cat", str2 = "cut"

Output

1

We can convert str1 into str2 by replacing 'a' with 'u'.

Input

str1 = "sunday", str2 = "saturday"

Output

3

```
public int getMinConversions(string str1, string str2)
```

```
{
    int dp[][] = new int [str1.length()+1]
                      [str2.length()+1]
```

```
    for (int i=0; i<=str1.length(); i++) {
```

```
        for (int j=0; j<=str2.length(); j++) {
```

```
            if (i==0)
```

```
                dp[i][j] = j;
```

```
            else if (j==0)
```

```
                dp[i][j] = i;
```

```
            else if (str1.charAt(i-1) == str2.charAt(j-1))
```

```
                dp[i][j] = dp[i-1][j-1];
```

else

{

$dp[i][j] = 1 + \text{Math.min}(dp[i-1][j-1], dp[i-1][j])$

}

}

return $dp[\text{str1.length}()][\text{str2.length}()];$

}

- 2) Given N numbers n_1, n_2, \dots, n_N and Q queries q_1, q_2, \dots, q_Q . Your task is to print Q ($Q < j < \text{numbers}$) f_1, f_2, \dots, f_Q , corresponding to query q_j $\max(n_1=f_j, n_2, \dots, n_Q)$ using dynamic programming.

Input

5 3

5 4 8 6 9

2 3 5

Output

5 8 9

```
def next_greater(next, a, n)
```

```
    s = []
```

```
    s.append(0)
```

```
    for i in range(1, n):
```

```
        while (len(s) != 0):
```

```
            cur = s[-1]
```

```
            if (a[cur] < a[i]):
```

```
                next[cur] = i;
```

```
            s.pop();
```

```
        else
```

```
            break;
```

```
    s.append(i);
```

```
    while (len(s) != 0):
```

```
        cur = s[-1]
```

```
        next[cur] = -1;
```

```
        s.pop();
```

```
def answer = query(a, next, index):
```

```
    position = next[index];
```

```
    if (position == -1)
```

```
        return -1;
```

else

return a[position];

next.greater(next, a, n)

print (ans_query (a, next, n, 3), end = " ")

print (ans_query (a, next, n, 6), end = " ")

print (ans_query (a, next, n, 7), end = " ")

O/p

589

In-Lab:

- 1) Bhanu is a student at KL University who likes playing with strings, after reading a question from their lab workbook for the ADA Course she found what is meant by a subsequence.
 (A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements)
 So, she created 2 strings out of which one she considered as a master string and the other one as a slave string. She challenged her friend Teju to find out whether the slave string is a subsequence of the master string or not, As Teju is undergoing her CRT classes she decided to code the logic for this question. Help her in building the logic and write a code using Dynamic programming concept.

```
def les_algo (s1, s2, m, n):
```

```
    L = (0 for x in longest) for x in range(n+1)
```

```
    for i in range(n+1)
```

```
        for j in range(m+1)
```

```
            if i == 0 or j == 0
```

```
                L[i][j] = 0
```

```
            elif s1[i-1] == s2[j-1]:
```

```
                L[i][j] = L[i-1][j-1] + 1
```

```
            else
```

```
                L[i][j] = max(L[i-1][j], L[i][j-1])
```

```
    index = L[m][n]
```

```
    i = m
```

```
    j = n
```

```
    while i > 0 and j > 0:
```

```
        if s1[i-1] == s2[j-1]:
```

```
            les_algo(index-1) = s1[i-1]
```

```
            i -= 1
```

```
            j -= 1
```

```
    index = 1
```

else

return a[position];

next.greater(next, a, n)

print (ans_query(a, next, n, 3), end = " ")

print (ans_query(a, next, n, 6), end = " ")

print (ans_query(a, next, n, 7), end = " ")

O/p

589

In-Lab:

- 1) Bhanu is a student at KL University who likes playing with strings, after reading a question from their lab workbook for the ADA Course she found what is meant by a subsequence.

(A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements)

So, she created 2 strings out of which one she considered as a master string and the other one as a slave string. She challenged her friend Teju to find out whether the slave string is a subsequence of the master string or not. As Teju is undergoing her CRT classes she decided to code the logic for this question. Help her in building the logic and write a code using Dynamic programming concept.

```
def lcs-algo (s1, s2, m, n):
```

```
    L = (0 for x in longest) for x in range(n+1)
```

```
    for i in range(n+1)
```

```
        for j in range(m+1)
```

```
            if i == 0 or j == 0
```

```
                L[i][j] = 0
```

```
            elif s1[i-1] == s2[j-1]:
```

```
                L[i][j] = L[i-1][j-1] + 1
```

```
            else
```

```
                L[i][j] = max(L[i-1][j], L[i][j-1])
```

```
    index = L[m][n]
```

```
    i = m
```

```
    j = n
```

```
    while i > 0 & j > 0:
```

```
        if s1[i-1] == s2[j-1]:
```

```
            lcs-also L[index-1] = s1[i-1]
```

```
            i -= 1
```

```
            j -= 1
```

```
    index -= 1
```

- 2) Geeta is working in a company, and she has n different projects to work on, where every project is scheduled to be done from $\text{startTime}[i]$ to $\text{endTime}[i]$, obtaining a profit of $\text{profit}[i]$. You are given the startTime , endTime and profit arrays, return the maximum profit you can take such that there are no two projects that she is working on in that given subset with overlapping time range. If she chooses a project that ends at time a then she will be able to start another project that starts at time b .

Input

$\text{startTime} = [1, 2, 3, 4, 6]$, $\text{endTime} = [3, 5, 10, 6, 9]$, $\text{profit} = [20, 20, 100, 70, 60]$

Output

150

Explanation: The subset chosen is the first, fourth and fifth project.
Profit obtained $150 = 20 + 70 + 60$.

```
import java.util.*;
```

```
public class solution {
```

```
    public static int jobScheduling (int L,
                                     starttime, int[] end)
                                     int[] profit)
```

```
    {
```

```
        int n = starttime.length;
```

```
        Map<Integer, List<int>> map =
```

```
            new HashMap<>();
```

```
        int maxEndTime = 0;
```

```
        for (int i = 0; i < n; i++)
```

```
        {
```

```
            if (!map.containsKey(endTime[i]))
```

```
            {
```

```
                map.put(i, new ArrayList<>());
```

```
            }
```

```
            dp[0] = 0;
```

```
            dp[1] = 0;
```



```

for (int i = 2; i < max & Time; i++)
{
    dp[i] = dp[i-1];
}

```

1/p

starttime: [1, 2, 3, 4, 6]

end Time = [3, 5, 10, 6, 9]

profit = [20, 10, 70, 10, 60]

o/p

150

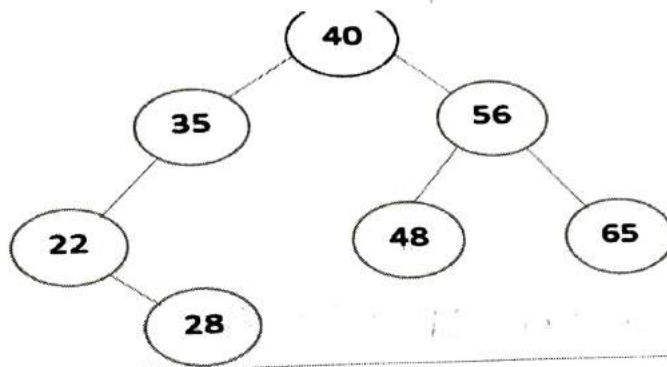
3) Teacher: good morning students!!!
 Students: good morning mam

Teacher: Today topic is binary tree. Does anyone know what is binary tree?

Students: no mam

Teacher: In computer science, a binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child. Satish you completed this topic yesterday can you explain this topic?

Satish: Sure, mam but I got doubt this concept, do anyone help me in solving this the question was Given a binary tree, find whether if a given Binary Tree is Balanced?



```
if __name__ == '__main__':
```

```
    root.left = Node(35)
```

```
    root.right = Node(56)
```

```
    root.left.left = Node(22)
```

```
    root.left.leftright = Node(28)
```

```
    root.right.left = Node(6)
```

```
if isHeightBalanced(root) [1]:
```

```
    print("Binary tree is balanced")
```

```
else
```

```
    print("Binary tree is not balanced")
```

Class Node:

```
def __init__(self, key, left=None, right=None):
```

```
    self.key = key
```

```
    self.left = left
```

```
    self.right = right
```

```
def isHeightBalanced(root, isBalanced=True):
```

```
    if root is None or not isBalanced:
```

```
        return 0, isBalanced
```

```
    left_height, is_balanced = isHeightBalanced
```

```
        (root.left, isBalanced)
```

```
    right_height, isBalanced = isHeightBalanced(root.right
```

```
        is_balanced)
```

```
    if abs(left_height - right_height) > 1:
```

```
        isBalanced = False
```

```
    return max(left_height, right_height) + 1, isBalanced
```

Post-Lab:
 1) SURI studies at KL University and a person who is interested in Dynamic Programming, she created a question for you to solve. she decided to give a question related to palindrome, you need to use dynamic programming to solve this problem brute force is not allowed since she hates waiting too long to find the answer. The question follows like this find the longest palindromic subsequence.

(Unlike substrings, subsequences are not required to occupy consecutive positions within the original string.)

Input

ABBDACACB

Output

The length of the longest palindromic subsequence is 5

The longest palindromic subsequence is BCACB

```
def max(x, y):
```

```
    if (x > y)
```

```
        return x
```

```
    return y
```

```
def lps(seq, i, j):
```

```
    if (i == j):
```

```
        return 1
```

```
    if (seq[i] == seq[j] and i+1 == j):
```

```
        return 2
```

```
    if (seq[i] == seq[j]):
```

```
        return lps(seq, i+1, j-1) + 2
```

```
    return max(lps(seq, i, j-1),
```

```
               lps(seq, i+1, j))
```

```
if __name__ == '__main__':
```

```
    seq = "ABBD CACB"
```

```
    n = len(seq)
```

```
    print("the length of LPS is" lps (seq, 0, n-1))
```

Output

The length of the LPS is 5


```
for i in range(1, face+1):
```

```
    ans += NoofWay(face, throw-1, sum-i);
```

```
dp[throws][sum] = ans;
```

```
return ans;
```

```
if __name__ == '__main__':
```

```
    faces = 2; throws = 2; sum = 3;
```

```
    for i in range(55)
```

```
        for j in range(55)
```

```
            dp[i][j] = -1
```

```
    print(NoofWay(faces, throws, sum));
```

- 2) Bhanu and Teju are playing dice game where there are N dice with M faces and the dice are numbered from 1 to M. A person wins the game if the sum of the faces of dice adds up to a value X. you are playing on Bhanu's team, and It is Teju's turn now. You are supposed to find number of ways your opponent can win the game where N, M and X are provided as input. Use Dynamic programming to solve the problem. Using DP (Dynamic programming) to find the number of ways to get sum X.

Input

M = 2

N = 2

X = 3

Output

2

```
import numpy as np
mod = 1000000007
```

```
dp = np.zeros(55, 55);
```

```
def NoofWays(face, throws, sum):
```

```
    if (sum == 0 and throws == 0):
```

```
        return 1
```

```
    if (sum < 0 or throws == 0):
```

```
        return 0;
```

```
    if (dp[throws][sum] != -1):
```

```
        return dp[throws][sum];
```

```
ans = 0)
```

(For Evaluator's use only)

Comment of the Evaluator (if Any)Evaluator's Observation

Marks Secured: out of

Full Name of the Evaluator:

P. Anurag

(R)

Signature of the Evaluator Date of Evaluation