05 - Dynamic Programming

Ex. No. : 5.1 **Date: 10.09.24**

Register No.: 230701385 **Name: S VISHWAK**

AIM:

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer ‘n’ and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3.Write any efficient algorithm to find the possible ways.

Example 1:

Input: 6

Output:6

Explanation: There are 6 ways to 6 represent number with 1 and 3

1+1+1+1+1+1

3+3

1+1+1+3

1+1+3+1

1+3+1+1

3+1+1+1

Input Format

First Line contains the number n

Output Format

Print: The number of possible ways ‘n’ can be represented using 1 and 3

Sample Input

6

Sample Output

6

ALGORITHM:

Step 1: Start

Step 2: Declare n and read input value

Step 3: Create an array dp of size n + 1, initialize dp[0] to 1, and set all other elements to 0

Step 4: Iterate from 1 to n, updating dp[i] by adding dp[i - 1]. If i is greater than or equal to 3, also add dp[i - 3] to dp[i]

Step 5: Print the value dp[n]

Step 6: End

PROGRAM:

```
#include<stdio.h>

int main() {

    int n;

    scanf("%d", &n);

    long dp[n+1];

    dp[0] = 1;

    for (int i = 1; i <= n; i++) {

        dp[i] = 0;

    }

    for (int i = 1; i <= n; i++) {
```

```
        dp[i] += dp[i - 1];
        if (i >= 3) {
            dp[i] += dp[i - 3];
        }
    }
    printf("%ld\n", dp[n]);
    return 0;
}
```

OUTPUT:

	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	243828195967216 29	243828195967216 29	✔

RESULT :

Hence the above program has been executed successfully..

Ex. No. : 5.2 **Date: 10.09.24**

Register No.: 230701385 **Name: S VISHWAK**

AIM:

Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

Example:
Input
3
1 2 4
2 3 4
8 7 1
Output:
19

Input Format
First Line contains the integer n
The next n lines contain the n*n chessboard values

Output Format
Print Maximum monetary value of the path

ALGORITHM:

Step 1: Start

Step 2: Declare n, read input value, and create a 2D array board of size n x n to store its values

Step 3: Initialize dp[0][0] with board[0][0], populate the first row and column of dp by accumulating values from board

Step 4: Iterate through the remaining cells of the dp array, updating each cell with the maximum path sum from either the top or left cell, and board[i][j]

Step 5: Print the value dp[n-1][n-1]

PROGRAM:

```
#include<stdio.h>
int max(int a,int b) {
    return(a>b) ? a:b;
}
int maxMonetaryPath(int n,int board[n][n]){
    int dp[n][n];
    dp[0][0]=board[0][0];
    for(int j=1;j<n;j++){
        dp[0][j]=dp[0][j-1]+board[0][j];
    }
    for (int i=1;i<n;i++) {
        dp[i][0]=dp[i-1][0]+board[i][0];
    }
    for (int i=1;i<n;i++) {
        for (int j=1;j<n;j++) {
            dp[i][j]=board[i][j]+max(dp[i-1][j],dp[i][j-1]);
        }
    }
    return dp[n-1][n-1];
```

```
}
int main(){
    int n;
    scanf("%d",&n);
    int board[n][n];
    for (int i=0;i<n;i++){
        for (int j=0;j<n;j++){
            scanf("%d",&board[i][j]);
        }
    }
int result=maxMonetaryPath(n,board);
    printf("%d\n",result);
}
```

OUTPUT:

	Input	Expected	Got	
✔	3 1 2 4 2 3 4 8 7 1	19	19	✔
✔	3 1 3 1 1 5 1 4 2 1	12	12	✔
✔	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✔

RESULT :

Hence the above program has been executed successfully..

Ex. No. : 5.3 **Date: 10.09.24**

Register No.: 230701385 **Name: S VISHWAK**

AIM:

Given two strings, find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

s1: ggtabe

s2: tgatasb

s1	a	g	**g**	**t**	**a**	**b**	
s2	**g**	x	**t**	**x**	**a**	**y**	**b**

The length is 4

Solving it using Dynamic Programming

For example:

Input	Result
aab azb	2

ALGORITHM:

Step 1: Start

Step 2: Declare s1 and s2 as character arrays and read the input values

Step 3: Calculate the lengths of s1 and s2, and create a 2D array dp of size (len1 + 1) x (len2 + 1)

Step 4: Initialize the first row and column of dp to 0, then iterate through the arrays to fill dp by comparing characters of s1 and s2 and taking the maximum of the adjacent values

Step 5: Print dp[len1][len2]

PROGRAM:

```
#include<stdio.h>

#include<string.h>

int main()

{

    char s1[10],s2[10];

    scanf("%s",s1);

    scanf("%s",s2);

    int len1=strlen(s1);

    int len2=strlen(s2);

    int dp[len1 + 1][len2 + 1];
```

```
    for(int i=0;i<=len1;i++)
    {
        for(int j=0;j<=len2;j++)
        {
            if(i==0||j==0)
            {
                dp[i][j]=0;
            }
            else if(s1[i-1]==s2[j-1]){
                dp[i][j]=dp[i-1][j-1]+1;
            }
            else{
                if(dp[i][j-1]>dp[i-1][j])
                    dp[i][j]=dp[i][j-1];
                else
                    dp[i][j]=dp[i-1][j];
                }  }
    }
    printf("%d",dp[len1][len2]);
}
```

OUTPUT:

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

RESULT :

Hence the above program has been executed successfully..

Ex. No. : 5.4 **Date: 10.09.24**

Register No.: 230701385 **Name: S VISHWAK**

AIM:

Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Input:9

Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

ALGORITHM:

Step 1: Start

Step 2: Declare n, read the input value, and create an array arr of size n to store its values

Step 3: Initialize a 1D array dp of size n with all elements set to 1, and a variable maxlen to 1

Step 4: Iterate through the array arr to fill dp by comparing elements, updating dp[i] if arr[i] is greater than or equal to arr[j] and dp[i] < dp[j] + 1, also update maxlen.

Step 5: Print maxlen.

PROGRAM:

```
#include <stdio.h>
int subsequence(int arr[],int n){
    int dp[n];
    int maxlen=1;
    for (int i=0;i<n;i++){
        dp[i]=1;
    }
    for (int i=1;i<n;i++){
        for(int j=0;j<i;j++){
            if(arr[i]>=arr[j] && dp[i]<dp[j]+1){
                dp[i]=dp[j]+1;
            }
        }
        if(maxlen<dp[i]){
            maxlen=dp[i];
        }
    }
    return maxlen;
```

```
}

int main(){
    int n;
    scanf("%d",&n);
    int arr[n];
    for (int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    int result=subsequence(arr,n);
    printf("%d",result);
}
```

OUTPUT:

	Input	Expected	Got	
✔	9 -1 3 4 5 2 2 2 2 3	6	6	✔
✔	7 1 2 2 4 5 7 6	6	6	✔

RESULT :

Hence the above program has been executed successfully..