

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Experiment-10

Student Name: Kanishk Soni
Branch: BE-CSE
Semester: 6th
Subject Code: 20CSP-351

UID: 20BCS9398
Section/Group: 20BCS_DM-708B
Subject Name: Competitive Coding-II

AIM: To demonstrate the concept of Dynamic Programming.

Problem1: Climbing Stairs

<https://leetcode.com/problems/climbing-stairs/>

Program Code:

```
class Solution {  
public:  
    int climbStairs(int n) {  
        if(n<=2) {  
            return n;  
        }  
        vector<int> dp(n+1);  
        dp[0]=0;  
        dp[1]=1;  
        dp[2]=2;  
        for(int i=3;i<=n;i++) {  
            dp[i]=dp[i-1]+dp[i-2];  
        }  
        return dp[n];  
    }  
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

};

Output:

The screenshot displays a LeetCode interface for the 'Climbing Stairs' problem. On the left, there's a sidebar with 'Accepted' status, 'Next question' (71. Simplify Path), and 'More challenges' (746. Min Cost Climbing Stairs, 509. Fibonacci Number, 1137. N-th Tribonacci Number). The main area shows the problem title 'Climbing Stairs', a 'C++' language tag, and a 'Details' button. Below this, there's a performance summary: 'Runtime 0 ms', 'Beats 100%', 'Memory 6.1 MB', and 'Beats 39.10%'. A 'Notes' section is present with a text input field. The 'Related Tags' section shows 'Select tags' and '0/5'. The solution code is displayed in a dark-themed editor, showing a C++ class 'Solution' with a public method 'climbStairs' that uses dynamic programming to calculate the number of ways to climb stairs.

Problem2: Longest Palindromic Substring

<https://leetcode.com/problems/longest-palindromic-substring/>

Program Code:

```
class Solution {  
  
public:  
  
    void chk(int l,int r,string &s,int &max,string &ans){  
  
        while(l>=0 && r<s.size() && s[l]==s[r]){  
  
            l--;  
  
            r++;  
  
        }  
  
        l++;  
  
        r--;  
  
        int a=r-l+1;  
  
        if(a>max){
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
        ans=s.substr(l,a);

        max=a;
    }
}

string longestPalindrome(string s) {
    int n=s.size();
    int max=0;
    string ans = "";
    for(int i=0;i<n;i++){
        chk(i,i,s,max,ans);
        if(i==n-1){
            break;
        }
        chk(i,i+1,s,max,ans);
        if(max==n){
            return s;
        }
    }
    return ans;
}

};
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Output:

Accepted

Next question

6. Zigzag Conversion

More challenges

214. Shortest Palindrome

266. Palindrome Permutation

336. Palindrome Pairs

All statuses

All languages

Accepted

a few seconds ago

C++

Accepted

a few seconds ago

C++

Kanishk

May 09, 2023 21:32

Details

+ Solution

C++

Runtime 13 ms

Beats 88.57%

Memory 11.2 MB

Beats 55.91%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags

0/5

```
class Solution {
public:
    void chk(int l,int r,string &s,int &max,string &ans){
        while(l<=r && r<s.size() && s[l]==s[r]){
            l--;
            r++;
        }
        l++;
        r--;
        int a=r-l+1;
        if(a>max){
            ans=s.substr(l,a);
            max=a;
        }
    }
    string longestPalindrome(string s) {
        int n=s.size();
```