Dynamic Programming Introduction**

12 October 2024 15:48

```
class Solution {
    static int mod = (int)1e9 + 7;

    static long fiboTopDown(int n, long[] dp){
        if(n<=1) return n;
        if(dp[n] != -1) return dp[n];
        return dp[n] = (fiboTopDown(n-1,dp) + fiboTopDown(n-2,dp))%mod;
    }
    static long topDown(int n) {
        // code here
        long dp[] = new long[n+1];
        Arrays.fill(dp,-1);
        return fiboTopDown(n,dp);
    }

    static long  (int n) {
        if(n<=1) return n;
        // code here
        long dp[] = new long[n+1];
        Arrays.fill(dp,-1);
        dp[0] = 0;
        dp[1] = 1;
        for(int i=2; i<=n; i++){
            dp[i] = (dp[i-1] +dp[i-2])%mod;
        }
        return dp[n];

    }
}
```

Darde sir
Aafate jaan
Qayamate khas
Mallikae fasad
Sheere zeher
Zehni khalal

Climbing Stars

12 October 2024 16:19

```java
class Solution {
    public int climbStairs(int n)
    {
        if(n<=3)
        {
            return n;
        }
        int dp[]=new int[n+1];
        Arrays.fill(dp,-1);
        dp[0]=0;
        dp[1]=1;
        dp[2]=2;
        dp[3]=3;
        for(int i=4;i<=n;i++)
        {
            dp[i]=dp[i-1]+dp[i-2];
        }
        return dp[n];

    }
}
```

Frog Jump(DP-3)

12 October 2024 20:26

```
class Solution{
   public int minimumEnergy(int arr[],int N)
   {
      if(N==1)
      {
         return 0;
      }

      if(N==2)
      {
         return arr[1]-arr[0];
      }
      if(N==3)
      {
         return Math.max(arr[2]-arr[0],arr[2]-arr[1]);
      }
      int dp[]=new int[N+1];
      Arrays.fill(dp,0);
      dp[0]=0;
      dp[1]=0;
      dp[2]=Math.abs(arr[1]-arr[0]);
      dp[3]=Math.min(dp[1]+Math.abs(arr[2]-arr[0]),dp[2]+Math.abs(arr[2]-arr[1]));

      for(int i=4;i<=N;i++)
      {
         int a=dp[i-1]+Math.abs(arr[i-1]-arr[i-2]);
         int b=dp[i-2]+Math.abs(arr[i-1]-arr[i-3]);

         dp[i]=Math.min(a,b);
      }
      return dp[N];

   }
}
```

Use a and b instead of dp to optimize space

Frog Jump with k distances(DP-4) ***

12 October 2024 20:27

```
public class Solution {

    public static int solve(int idx, int[] height, int k, int[] dp) {
        if (idx == 0) return 0;
        if (dp[idx] != -1) return dp[idx];

        int minSteps = Integer.MAX_VALUE;

        for (int j = 1; j <= k; j++) {
            if (idx - j >= 0) {
                int jump = solve(idx - j, height, k, dp) + Math.abs(height[idx] - height[idx - j]);
                minSteps = Math.min(minSteps, jump);
            }
        }

        return dp[idx] = minSteps;
    }

    public static int frogJump(int n, int k, int[] height) {
        int[] dp = new int[n];
        Arrays.fill(dp, -1);
        return solve(n - 1, height, k, dp);
    }
}

{10, 30, 40, 50, 20};

public static int frogJump(int n, int k, int[] height) {
    int[] dp = new int[n];
    dp[0] = 0;

    for (int i = 1; i < n; i++) {
        int minSteps = Integer.MAX_VALUE;

        for (int j = 1; j <= k; j++) {
            if (i - j >= 0) {
                int jump = dp[i - j] + Math.abs(height[i] - height[i - j]);
                minSteps = Math.min(minSteps, jump);
            }
        }

        dp[i] = minSteps;
    }

    return dp[n - 1];
}
```

Maximum sum of non-adjacent elements (DP 5)

12 October 2024 20:27

```
class Solution {
    public int rob(int[] nums)
    {
        int n=nums.length;
        if(n==0)
        {
            return 0;
        }
        if(n==1)
        {
            return nums[0];
        }
        if(n==2)
        {
            return Math.max(nums[0],nums[1]);
        }
        int[]dp=new int[n+1];
        Arrays.fill(dp,-1);
        dp[0]=0;
        dp[1]=nums[0];
        dp[2]=nums[1];
        dp[3]=dp[1]+nums[2];

        for(int i=4;i<=n;i++)
        {
            dp[i]=Math.max(dp[i-2],dp[i-3])+nums[i-1];
        }
        int ans=nums[0];
        for(int i=0;i<=n;i++)
        {
            ans=Math.max(ans,dp[i]);
        }
        return ans;
    }
}
TC ans SC is O(N) and O(1) isme try kro
```

#RECURSIVE SOLN

```
public class Solution {
   public int rob(int[] nums) {
      int n = nums.length;
      int[] dp = new int[n + 1];
      Arrays.fill(dp, -1);
      return helper(n - 1, nums, dp);
   }

   private int helper(int i, int[] nums, int[] dp) {
      if (i < 0) return 0;
      if (dp[i] != -1) return dp[i];

      int robFromTwoBack = helper(i - 2, nums, dp);
      int robFromThreeBack = helper(i - 3, nums, dp);

      dp[i] = nums[i] + Math.max(robFromTwoBack, robFromThreeBack);
      return dp[i];
```

```
    }
}
```

House Robber (DP 6) ***

12 October 2024 20:27

```java
public int rob1(int[] nums)
    {
        int n=nums.length;
        if(n==0)
        {
            return 0;
        }
        if(n==1)
        {
            return nums[0];
        }
        if(n==2)
        {
            return Math.max(nums[0],nums[1]);
        }
        int[]dp=new int[n+1];
        Arrays.fill(dp,-1);
        dp[0]=0;
        dp[1]=nums[0];
        dp[2]=nums[1];
        dp[3]=dp[1]+nums[2];

        for(int i=4;i<=n;i++)
        {
            dp[i]=Math.max(dp[i-2],dp[i-3])+nums[i-1];
        }
        int ans=nums[0];
        for(int i=0;i<=n;i++)
        {
            ans=Math.max(ans,dp[i]);
        }
        return ans;
    }
    public int rob(int[] nums)
    {
        int n=nums.length;
        if(n==0)
        {
            return 0;
        }
        if(n==1)
        {
            return nums[0];
        }
        if(n==2)
        {
            return Math.max(nums[0],nums[1]);
        }
        int[]temp=new int[nums.length-1];
        for(int i=0;i<nums.length-1;i++)
        {
            temp[i]=nums[i];
        }
        int[]res=new int[nums.length-1];
        for(int i=1;i<nums.length;i++)
        {
            res[i-1]=nums[i];
        }
        return Math.max(rob1(temp),rob1(res));

    }
```

Ninja's Training (DP 7)

13 October 2024 12:03

```
class Solution {
   public int maximumPoints(int arr[][], int N)
   {
      int m=arr.length;
      int n=arr[0].length;
      int[][] ans=new int[m][n];
      for(int i=0;i<n;i++)
      {
         ans[0][i]=arr[0][i];
      }


      for(int i=1;i<m;i++)
      {
         for(int j=0;j<n;j++)
         {
            int res=0;
            for(int k=0;k<n;k++)
            {
               if(k!=j)
               {
                  res=Math.max(res,ans[i-1][k]);
               }


            }
            ans[i][j]=arr[i][j]+res;
         }
      }

      int sum=0;
      for(int i=0;i<n;i++)
      {
         sum=Math.max(sum,ans[m-1][i]);
      }
      return sum;
   }
}
```

Grid Unique Paths : DP on Grids (DP8)

13 October 2024 12:03

```
class Solution {
    public int uniquePaths(int m, int n)
    {
       int[][]arr=new int[m][n];
       for(int i=0;i<n;i++)
       {
        arr[m-1][i]=1;
       }
       for(int i=0;i<m;i++)
       {
        arr[i][n-1]=1;
       }
       for(int i=m-2;i>=0;i--)
       {
        for(int j=n-2;j>=0;j--)
        {
            arr[i][j]=arr[i][j+1]+arr[i+1][j];
        }
       }
       return arr[0][0];

    }
}
```

Grid Unique Paths 2 (DP 9)

13 October 2024 12:03

```
class Solution {
    public int uniquePathsWithObstacles(int[][] nums)
    {
        int m=nums.length;
        int n=nums[0].length;
        int[][]arr=new int[m][n];
        if(nums[m-1][n-1]==1||nums[0][0]==1)
        {
            return 0;
        }
        int found=1;
        for(int i=n-1;i>=0;i--)
        {
            if(nums[m-1][i]!=1&&found==1)
            {
                arr[m-1][i]=1;
            }
            else
            {
                found=0;
                arr[m-1][i]=0;
            }

        }
        int found1=1;
        for(int i=m-1;i>=0;i--)
        {
            if(nums[i][n-1]!=1&&found1==1)
            {
                arr[i][n-1]=1;
            }
            else
            {
                found1=0;
                arr[i][n-1]=0;
            }
        }
        for(int i=m-2;i>=0;i--)
        {
            for(int j=n-2;j>=0;j--)
            {
                if(nums[i][j]!=1)
                {
                    arr[i][j]=arr[i][j+1]+arr[i+1][j];
                }
                else
                {
                    arr[i][j]=0;
                }

            }
        }
       return arr[0][0];
    }
}

// found1 ka koi Zaroorat nahi tha vaise socho ek baar
```

Minimum path sum in Grid (DP 10)

13 October 2024 12:04

```java
class Solution {
    public int minPathSum(int[][] nums)
    {
        int m=nums.length;
        int n=nums[0].length;
        int[][]arr=new int[m][n];
        // if(nums[m-1][n-1]==0||nums[0][0]==0)
        // {
        //     return 0;
        // }
        arr[m-1][n-1]=nums[m-1][n-1];

        for(int i=n-2;i>=0;i--)
        {
            arr[m-1][i]=arr[m-1][i+1]+nums[m-1][i];
        }
        for(int i=m-2;i>=0;i--)
        {
            arr[i][n-1]=arr[i+1][n-1]+nums[i][n-1];
        }
        for(int i=m-2;i>=0;i--)
        {
            for(int j=n-2;j>=0;j--)
            {
                arr[i][j]=Math.min(arr[i][j+1],arr[i+1][j])+nums[i][j];
            }
        }
        return arr[0][0];
    }
}
```

Minimum path sum in Triangular Grid (DP 11)

13 October 2024 12:04

```
class Solution {
    public int minimumTotal(List<List<Integer>> list)
    {
        int m=list.size();
        int n=list.get(m-1).size();
        int[][]arr=new int[m][n];
        int[][]nums=new int[m][n];
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<list.get(i).size();j++)
            {
                nums[i][j]=list.get(i).get(j);
            }
        }
        for(int i=0;i<n;i++)
        {
            arr[m-1][i]=nums[m-1][i];
        }
        for(int i=m-2;i>=0;i--)
        {
            for(int j=n-2;j>=0;j--)
            {
                arr[i][j]=nums[i][j]+Math.min(arr[i+1][j],arr[i+1][j+1]);
            }
        }
        return arr[0][0];

    }
}
```

```
//COPIED CODE
// Approach - 1 Memorization
class Solution {
    public int minimumTotal(List<List<Integer>> triangle) {

        int n = triangle.size();
        int [][] dp =  new int [n][n];
        for(int [] i: dp) Arrays.fill(i,-1);
        return solve (0,0,n,triangle,dp);
    }
    int solve(int i, int j,int n, List<List<Integer>> triangle,int [][] dp){
        if(i>=n||j>=n) return (int)(1e9);
        if(i==n-1) return triangle.get(i).get(j);
        if(dp[i][j]!=-1) return dp[i][j];
        int down = triangle.get(i).get(j)+ solve(i+1,j,n,triangle,dp);
        int diagonal = triangle.get(i).get(j)+ solve(i+1,j+1,n,triangle,dp);
        return dp[i][j] = Math.min(down,diagonal);
    }
}
```

```
import java.util.*;

class TUF {
    // Function to find the minimum path sum in the triangle using dynamic programming
    static int minimumPathSum(int[][] triangle, int n) {
        // Create two arrays to store intermediate results: front and cur
```

```
        int[] front = new int[n]; // Stores the results for the current row
        int[] cur = new int[n];   // Stores the results for the next row

        // Initialize the front array with the values from the bottom row of the triangle
        for (int j = 0; j < n; j++) {
            front[j] = triangle[n - 1][j];
        }

        // Starting from the second to last row, calculate the minimum path sum for each element
        for (int i = n - 2; i >= 0; i--) {
            for (int j = i; j >= 0; j--) {
                // Calculate the two possible paths: moving down or moving diagonally
                int down = triangle[i][j] + front[j];
                int diagonal = triangle[i][j] + front[j + 1];

                // Store the minimum of the two paths in the cur array
                cur[j] = Math.min(down, diagonal);
            }

            // Update the front array with the values from the cur array for the next row
            front = cur.clone();
        }

        // The result is stored at the top of the front array
        return front[0];
    }

    public static void main(String args[]) {
        int triangle[][] = {{1},
                            {2, 3},
                            {3, 6, 7},
                            {8, 9, 6, 10}};

        int n = triangle.length;

        // Call the minimumPathSum function and print the result
        System.out.println(minimumPathSum(triangle, n));
    }
}
```

Minimum/Maximum Falling Path Sum (DP-12)

13 October 2024 12:04

```
class Solution {
    public int minFallingPathSum(int[][] nums)
    {
        int m=nums.length;
        int n=nums[0].length;
        int[][]arr=new int[m][n];
        arr[m-1][n-1]=nums[m-1][n-1];
        //Arrays.fill(arr,0); //Learn how to fill 2D array using arrays.fill()
        for(int i=0;i<n;i++)
        {
            arr[0][i]=nums[0][i];
        }
        int sum=0;
        if(n==1)
        {
            for(int i=0;i<m;i++)
            {
                sum+=nums[i][0];
            }
            return sum;
        }
        for(int i=1;i<m;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(j==0)
                {
                    arr[i][j]=nums[i][j]+Math.min(arr[i-1][j],arr[i-1][j+1]);
                }
                else if(j==n-1)
                {
                    arr[i][j]=nums[i][j]+Math.min(arr[i-1][j],arr[i-1][j-1]);
                }
                else
                {
                    arr[i][j]=nums[i][j]+Math.min(Math.min(arr[i-1][j-1],arr[i-1][j]),arr[i-1][j+1]);
                }
            }
        }
        int res=arr[m-1][0];
        for(int i=0;i<n;i++)
        {
            res=Math.min(res,arr[m-1][i]);
        }
        return res;
    }
}
```

3-d DP : Ninja and his friends (DP-13)***

13 October 2024 12:04

Longest Common Subsequence | (DP - 25)

13 October 2024 16:03

```
class Solution {
    public int longestCommonSubsequence(String text1, String text2)
    {
        int m=text1.length();
        int n=text2.length();
        int[][]arr=new int[m+1][n+1];
        for(int i=0;i<=m;i++)
        {
            for(int j=0;j<=n;j++)
            {
                if(i==0||j==0)
                {
                    arr[i][j]=0;
                }
                else
                {
                    if(text1.charAt(i-1)==text2.charAt(j-1))
                    {
                        arr[i][j]=1+arr[i-1][j-1];
                    }
                    else
                    {
                        arr[i][j]=Math.max(arr[i-1][j],arr[i][j-1]);
                    }
                }
            }
        }
        return arr[m][n];

    }
}
```

Print Longest Common Subsequence | (DP - 26)***

13 October 2024 16:03

```
class Solution {
   public List<String> all_longest_common_subsequences(String text1, String text2)
   {
      int m=text1.length();
      int n=text2.length();
      int[][]arr=new int[m+1][n+1];
      for(int i=0;i<=m;i++)
      {
         for(int j=0;j<=n;j++)
         {
            if(i==0||j==0)
            {
               arr[i][j]=0;
            }
            else
            {
               if(text1.charAt(i-1)==text2.charAt(j-1))
               {
                  arr[i][j]=1+arr[i-1][j-1];
               }
               else
               {
                  arr[i][j]=Math.max(arr[i-1][j],arr[i][j-1]);
               }
            }
         }
      }
      String s="";
      List<String>s1=new ArrayList<>();

      int a=m;
      int b=n;

      while(a>0&&b>0)
      {
         if(text1.charAt(a-1)==text2.charAt(b-1))
         {
            s+=text1.charAt(a-1);
            a--;
            b--;
         }
         else if(arr[a-1][b]>arr[a][b-1])
         {
            a--;
         }
         else
         {
            b--;
         }
      }
      s1.add(s);
```

```
        //String result = new StringBuffer(s1).reverse().toString();
        return s1;

    }
}
```

Longest Common Substring | (DP - 27)

13 October 2024 16:03

```
class Solution {
   public int longestCommonSubstr(String text1, String text2)
   {

      int m=text1.length();
      int n=text2.length();

      int[][]arr=new int[m+1][n+1];

      for(int i=0;i<=m;i++)
      {
         for(int j=0;j<=n;j++)
         {
            if(i==0||j==0)
            {
               arr[i][j]=0;
            }
            else
            {
               if(text1.charAt(i-1)==text2.charAt(j-1))
               {
                  arr[i][j]=1+arr[i-1][j-1];
               }
               else
               {
                  arr[i][j]=0;
               }
            }
         }
      }
      int res=0;
      for(int i=0;i<=m;i++)
      {
         for(int j=0;j<=n;j++)
         {
            res=Math.max(res,arr[i][j]);
         }
      }
      return res;
   }
}
```

```
import java.util.*;

class TUF {
   // Function to find the length of the Longest Common Substring (LCS)
   static int lcs(String s1, String s2) {
```

```
        int n = s1.length();
        int m = s2.length();

        // Create arrays to store LCS lengths
        int prev[] = new int[m + 1];
        int cur[] = new int[m + 1];

        int ans = 0; // Initialize a variable to store the maximum LCS length

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= m; j++) {
                // If the characters at the current indices are the same, extend the LCS
                if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    int val = 1 + prev[j - 1];
                    cur[j] = val;
                    ans = Math.max(ans, val); // Update the maximum LCS length
                } else {
                    cur[j] = 0; // Reset LCS length if characters don't match
                }
            }
            // Update the 'prev' array to the values of 'cur' for the next iteration
            prev = cur.clone();
        }

        return ans; // Return the length of the Longest Common Substring (LCS)
    }

    public static void main(String args[]) {
        String s1 = "abcjklp";
        String s2 = "acjkp";

        // Call the lcs function and print the result
        System.out.println("The Length of Longest Common Substring is " + lcs(s1, s2));
    }
}
```

Longest Palindromic Subsequence | (DP-28)

13 October 2024 16:03

```java
class Solution {
    public int longestPalindromeSubseq(String text1)
    {
        int m=text1.length();
        int n=text1.length();
        int[][]arr=new int[m+1][n+1];
        //WAY TO REVERSE ANY STRING
        StringBuilder text2 = new StringBuilder();
        text2.append(text1);
        text2.reverse();
        for(int i=0;i<=m;i++)
        {
            for(int j=0;j<=n;j++)
            {
                if(i==0||j==0)
                {
                    arr[i][j]=0;
                }
                else
                {
                    if(text1.charAt(i-1)==text2.charAt(j-1))
                    {
                        arr[i][j]=1+arr[i-1][j-1];
                    }
                    else
                    {
                        arr[i][j]=Math.max(arr[i-1][j],arr[i][j-1]);
                    }
                }
            }
        }
        return arr[m][n];
    }
}
```

Minimum insertions to make string palindrome | DP-29

13 October 2024 16:03

```
class Solution {
    public int minInsertions(String text1)
    {
        int m=text1.length();
        int n=text1.length();
        int[][]arr=new int[m+1][n+1];
        //WAY TO REVERSE ANY STRING
        StringBuilder text2 = new StringBuilder();
        text2.append(text1);
        text2.reverse();
        for(int i=0;i<=m;i++)
        {
            for(int j=0;j<=n;j++)
            {
                if(i==0||j==0)
                {
                    arr[i][j]=0;
                }
                else
                {
                    if(text1.charAt(i-1)==text2.charAt(j-1))
                    {
                        arr[i][j]=1+arr[i-1][j-1];
                    }
                    else
                    {
                        arr[i][j]=Math.max(arr[i-1][j],arr[i][j-1]);
                    }
                }
            }
        }
        return text1.length()-arr[m][n];

    }
}
```

Minimum Insertions/Deletions to Convert String | (DP- 30)

13 October 2024 16:04

```
class Solution {
    public int minDistance(String text1, String text2)
    {
        int m=text1.length();
        int n=text2.length();
        int[][]arr=new int[m+1][n+1];
        //WAY TO REVERSE ANY STRING
        // StringBuilder text2 = new StringBuilder();
        // text2.append(text1);
        // text2.reverse();
        for(int i=0;i<=m;i++)
        {
            for(int j=0;j<=n;j++)
            {
                if(i==0||j==0)
                {
                    arr[i][j]=0;
                }
                else
                {
                    if(text1.charAt(i-1)==text2.charAt(j-1))
                    {
                        arr[i][j]=1+arr[i-1][j-1];
                    }
                    else
                    {
                        arr[i][j]=Math.max(arr[i-1][j],arr[i][j-1]);
                    }
                }
            }
        }
        return text1.length()-arr[m][n]+text2.length()-arr[m][n];

    }
}
```

Shortest Common Supersequence | (DP - 31)

13 October 2024 16:04

return sb.reverse().toString();

// WHY MY CODE IS NOT WORKING

```
class Solution {
    public String shortestCommonSupersequence(String text1, String text2)
    {
        int m=text1.length();
        int n=text2.length();
        int[][]arr=new int[m+1][n+1];
        for(int i=0;i<=m;i++)
        {
            for(int j=0;j<=n;j++)
            {
                if(i==0||j==0)
                {
                    arr[i][j]=0;
                }
                else
                {
                    if(text1.charAt(i-1)==text2.charAt(j-1))
                    {
                        arr[i][j]=1+arr[i-1][j-1];
                    }
                    else
                    {
                        arr[i][j]=Math.max(arr[i-1][j],arr[i][j-1]);
                    }
                }
            }
        }

        //String s="";
        StringBuilder sb =new StringBuilder();
        int a=m;
        int b=n;

        while(a>0&&b>0)
        {
            if(text1.charAt(a-1)==text2.charAt(b-1))
            {
                //s+=text1.charAt(a-1);
                sb.append(text1.charAt(a-1));
                a--;
                b--;
            }
            else if(arr[a-1][b]>arr[a][b-1])
            {
                //s+=text1.charAt(a-1);
                sb.append(text1.charAt(a-1));
                a--;
            }
            else
            {
                //s+=text2.charAt(b-1);
                sb.append(text2.charAt(b-1));
                b--;
```

```
            }
        }


            while(a>0)
            {
                //s+=text1.charAt(a-1);
                sb.append(text1.charAt(a-1));
                a--;
            }


            while(b>0)
            {
                //s+=text2.charAt(b-1);
                sb.append(text2.charAt(b-1));
                b--;
            }

        //return s;
        return sb.reverse().toString();

    }
}
```

Distinct Subsequences| (DP-32)***

13 October 2024 16:04

Edit Distance | (DP-33)

13 October 2024 16:04

```
class Solution {
    public int minDistance(String text1, String text2)
    {
        int m=text1.length();
        int n=text2.length();
        int[][]arr=new int[m+1][n+1];
        for(int i=0;i<=m;i++)
        {
            arr[i][0]=i;
        }
        for(int j=0;j<=n;j++)
        {
            arr[0][j]=j;
        }
        for(int i=1;i<=m;i++)
        {
            for(int j=1;j<=n;j++)
            {

                if(text1.charAt(i-1)==text2.charAt(j-1))
                {
                    arr[i][j]=arr[i-1][j-1];
                }
                else
                {
                    arr[i][j]=1+Math.min(arr[i-1][j],Math.min(arr[i-1][j-1],arr[i][j-1]));
                }

            }
        }
        return arr[m][n];
    }
}
```

//
a=self.findans(s,t,m-1,n-1) #replace
b=self.findans(s,t,m,n-1) #insert
c=self.findans(s,t,m-1,n) #delete

From <https://www.youtube.com/watch?v=3_KL0hiPsNE&t=56s>

Wildcard Matching | (DP-34)***

13 October 2024 16:04

Longest Increasing Subsequence |(DP-41)

13 October 2024 21:57

#CORRECT CODE

```
class Solution {
    public int lengthOfLIS(int[] nums)
    {
        int n=nums.length;
        int[]arr=new int[n];
        Arrays.fill(arr,1);
        //arr[0]=1;
        for(int i=1;i<n;i++)
        {
            int res=0;
            for(int j=i-1;j>=0;j--)
            {
                if(nums[i]>nums[j]                    res=Math.max(res,arr[j]);
                }

            }
            arr[i]=res+1;
        }
        int ans=0;
        for(int i=0;i<n;i++)
        {
            ans=Math.max(ans,arr[i]);
        }
        return ans;


    }
}
```

Printing Longest Increasing Subsequence|(DP-42)***

13 October 2024 21:57

```
class Solution {
   public ArrayList<Integer> longestIncreasingSubsequence(int n, int nums[])
   {
     int[] arr = new int[n];
      int[] hash = new int[n];
      Arrays.fill(arr, 1);
      Arrays.fill(hash, -1);

      for (int i = 0; i < n; i++) {
         hash[i] = i; // Important: initialize each hash[i] to itself
         for (int j = 0; j < i; j++) {
            if (nums[i] > nums[j] && arr[i] < arr[j] + 1) {
               arr[i] = arr[j] + 1;
               hash[i] = j;
            }
         }
      }

      int ans = 0;
      int index = 0;
      for (int i = 0; i < n; i++) {
         if (arr[i] > ans) {
            ans = arr[i];
            index = i;
         }
      }

      ArrayList<Integer> ans1 = new ArrayList<>();
      while (hash[index] != index) {
         ans1.add(nums[index]);
         index = hash[index];
      }
      ans1.add(nums[index]);  // don't forget to add the starting element!

      Collections.reverse(ans1);
      return ans1;

   }
}
```

Longest Increasing Subsequence |(DP-43)

13 October 2024 21:57

```
class Solution {
   // Function to find length of longest increasing subsequence.
   static int longestSubsequence(int n, int nums[])
   {

      //int n=nums.length;
      int[]arr=new int[n];
      Arrays.fill(arr,1);
      //arr[0]=1;
      for(int i=1;i<n;i++)
      {
         int res=0;
         for(int j=i-1;j>=0;j--)
         {
            if(nums[i]>nums[j])
            {
               res=Math.max(res,arr[j]);
            }

         }
         arr[i]=res+1;
      }
      int ans=0;
      for(int i=0;i<n;i++)
      {
         ans=Math.max(ans,arr[i]);
      }
      return ans;

   }
}
```

Largest Divisible Subset|(DP-44) ***

13 October 2024 21:58

nums =
[3,4,16,8]
Use Testcase
Output
[4,16]
Expected
[4,8,16]

```java
public List<Integer> largestDivisibleSubset(int[] nums) {
        int n=nums.length;
        int[] arr = new int[n];
        int[] hash = new int[n];
        Arrays.fill(arr, 1);
        Arrays.fill(hash, -1);
        Arrays.sort(nums);
        for (int i = 0; i < n; i++) {
            hash[i] = i; // Important: initialize each hash[i] to itself
            for (int j = 0; j < i; j++) {
                if (nums[i] % nums[j]==0 && arr[i] < arr[j] + 1) {
                    arr[i] = arr[j] + 1;
                    hash[i] = j;
                }
            }
        }
         int ans = 0;
        int index = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] > ans) {
                ans = arr[i];
                index = i;
            }
        }

        ArrayList<Integer> ans1 = new ArrayList<>();
        while (hash[index] != index) {
            ans1.add(nums[index]);
            index = hash[index];
        }
        ans1.add(nums[index]);  // don't forget to add the starting element!

        Collections.reverse(ans1);
        return ans1;

    }
```

Longest String Chain | (DP-45)

13 October 2024 21:58

```
public class Solution {
    public boolean checkPossibility(String longer, String shorter) {
        if (longer.length() != shorter.length() + 1) return false;

        int i = 0, j = 0;
        while (i < longer.length()) {
            if (j < shorter.length() && longer.charAt(i) == shorter.charAt(j)) {
                i++;
                j++;
            } else {
                i++;  // allow skipping only 1 character from the longer string
            }
        }
        return j == shorter.length();
    }

    public int longestStrChain(String[] words) {
        Arrays.sort(words, Comparator.comparingInt(String::length));  // sort by length

        int n = words.length;
        int[] dp = new int[n];
        Arrays.fill(dp, 1);

        int maxLen = 1;

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < i; j++) {
                if (checkPossibility(words[i], words[j]) && dp[i] < dp[j] + 1) {
                    dp[i] = dp[j] + 1;
                }
            }
            maxLen = Math.max(maxLen, dp[i]);
        }

        return maxLen;
    }
}
```

Longest Bitonic Subsequence |(DP-46)

13 October 2024 21:58

```
static int longestBitonicSequence(int[] arr, int n) {
        // Arrays to store lengths of increasing and decreasing subsequences
        int[] dp1 = new int[n];
        int[] dp2 = new int[n];
// Initialize both arrays with 1, as each element itself is a subsequence of length 1
        Arrays.fill(dp1, 1);
        Arrays.fill(dp2, 1);
// Calculate the lengths of increasing subsequences
        for (int i = 0; i < n; i++) {
            for (int prevIndex = 0; prevIndex < i; prevIndex++) {
                if (arr[prevIndex] < arr[i]) {
                    dp1[i] = Math.max(dp1[i], 1 + dp1[prevIndex]);
                }
            }
        }
// Reverse the direction of nested loops and calculate the lengths of decreasing subsequences
        for (int i = n - 1; i >= 0; i--) {
            for (int prevIndex = n - 1; prevIndex > i; prevIndex--) {
                if (arr[prevIndex] < arr[i]) {
                    dp2[i] = Math.max(dp2[i], 1 + dp2[prevIndex]);
                }
            }
        }
int maxi = -1;
// Calculate the length of the longest bitonic subsequence
        for (int i = 0; i < n; i++) {
            maxi = Math.max(maxi, dp1[i] + dp2[i] - 1);
        }
return maxi;
    }
```

From <https://takeuforward.org/data-structure/longest-bitonic-subsequence-dp-46/>

Number of Longest Increasing Subsequences|(DP-47)

13 October 2024 21:58

Subset sum equal to target (DP- 14)

14 October 2024 19:38

```
static Boolean isSubsetSum(int n, int arr[], int sum)
  {

    boolean[][]dp=new boolean[n+1][sum+1];
    for(int i=0;i<=n;i++)
    {
      dp[i][0]=true;
    }
    for(int i=1;i<=sum;i++)
    {
      dp[0][i]=false;
    }


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

          if(arr[i-1]<=j)
          {
            dp[i][j]=dp[i-1][j-arr[i-1]]||dp[i-1][j];
          }
          else
          {
            dp[i][j]=dp[i-1][j];
          }

      }
    }
    return dp[n][sum];
  }
```

Partition Equal Subset Sum (DP- 15)

14 October 2024 19:38

```
class Solution {
    public boolean canPartition(int[] arr)
    {
        int sum=0;
        for(int i=0;i<arr.length;i++)
        {
            sum+=arr[i];
        }

        if(sum%2!=0)
        {
            return false;
        }
        sum=sum/2;
        int n=arr.length;
        boolean[][]dp=new boolean[n+1][sum+1];
        for(int i=0;i<=n;i++)
        {
            dp[i][0]=true;
        }
        for(int i=1;i<=sum;i++)
        {
            dp[0][i]=false;
        }

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

                    if(arr[i-1]<=j)
                    {
                        dp[i][j]=dp[i-1][j-arr[i-1]]||dp[i-1][j];
                    }
                    else
                    {
                        dp[i][j]=dp[i-1][j];
                    }

            }
        }
        return dp[n][sum];

    }
}
```

Partition Set Into 2 Subsets With Min Absolute Sum Diff (DP-16)***

14 October 2024 19:39

Count Subsets with Sum K (DP - 17)

14 October 2024 19:39

Question =>
Given an array **arr** of size **n** of non-negative integers{0 bhi aaega ismebor } and an integer **sum**, the task is to count all subsets of the given array with a sum equal to a given **sum**.

arr = {0, 1, 2}
sum = 3

(Meaning two subsets: {0,1,2} and {1,2} both sum to 3 if you optionally include 0.)

```
public int perfectSum(int arr[],int n, int sum)
        {
            int modulo = 1000000007;
       int zero=0;
       for(int i=0;i<n;i++)
       {
          if(arr[i]==0)
          {
             zero++;

          }

       }
            int[][]dp=new int[n+1][sum+1];
       for(int i=0;i<=n;i++)
       {
          dp[i][0]=1;
       }
       for(int i=1;i<=sum;i++)
       {
          dp[0][i]=0;
       }



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

                if(arr[i-1]<=j&&arr[i-1]!=0)
                {
                   dp[i][j]=(dp[i-1][j-arr[i-1]]+dp[i-1][j])%modulo;
                }
                else
                {
                   dp[i][j]=dp[i-1][j]%modulo;
                }

          }
       }
       return ((1<<zero)*dp[n][sum])%modulo;

         }
```

Count Partitions with Given Difference (DP - 18)

14 October 2024 19:40

```
class Solution {
   public static int countPartitions(int n, int d, int[] arr)
   {
       //int mod = 1000000007;
      int sum=0;
      for(int i=0;i<arr.length;i++)
      {
         sum+=arr[i];
      }
      //System.out.println(sum);
      sum=sum+d;
      if(sum%2!=0)
      {
         return 0;
      }
      sum=sum/2;


      int modulo = 1000000007;

           int[][]dp=new int[n+1][sum+1];
      for(int i=0;i<=n;i++)
      {
         dp[i][0]=1;
      }

      for(int i=1;i<=sum;i++)
      {
         dp[0][i]=0;
      }



      for(int i=1;i<=n;i++)
      {
         for(int j=0;j<=sum;j++)  //Why j=0 why not j=1??
         {

               if(arr[i-1]<=j)
               {
                  dp[i][j]=(dp[i-1][j-arr[i-1]]+dp[i-1][j])%modulo;
               }
               else
               {
                  dp[i][j]=dp[i-1][j]%modulo;
               }

         }
      }
      return dp[n][sum];

   }
}
```

Assign Cookies

14 October 2024 19:40

#WE CAN ALSO SOLVE USING POINTER APPROCH

#BELOW I USE MIN PRIORITY

```
class Solution {
    public int findContentChildren(int[] g, int[] s)
    {
        PriorityQueue<Integer> pq1 = new PriorityQueue<>();//MIN HEAP
        PriorityQueue<Integer> pq2 = new PriorityQueue<>();//MIN HEAP
        int n=g.length;
        for(int i=0;i<n;i++)
        {
            pq1.add(g[i]);
        }
        int m=s.length;
        for(int i=0;i<m;i++)
        {
            pq2.add(s[i]);
        }int cnt=0;
        while(pq2.size()>0&&pq1.size()>0)
        {
            int a=pq1.peek();
            int b=pq2.peek();
            if(a<=b)
            {
                cnt++;
                pq1.poll();
                pq2.poll();
            }
            else
            {
                while(pq2.size()>0&&pq1.peek()>pq2.peek())
                {
                    pq2.poll();
                }
            }
        }
        return cnt;
    }
}
```

Minimum Coins (DP - 20)***

14 October 2024 19:40

Target Sum (DP - 21)

14 October 2024 19:40

WHAT IF ARR={100} AND TARGET = -200?

```
class Solution {
    public int findTargetSumWays(int[] arr, int d)
    {

        int n=arr.length;
        int sum=0;
        if(d<0)
        {
            d=-1*d;
        }
        for(int i=0;i<arr.length;i++)
        {
            sum+=arr[i];
        }
        //System.out.println(sum);
        sum=sum+d;
        if(sum%2!=0)
        {
            return 0;
        }
        sum=sum/2;

        int modulo = 1000000007;

        int[][]dp=new int[n+1][sum+1];
        for(int i=0;i<=n;i++)
        {
            dp[i][0]=1;
        }

        for(int i=1;i<=sum;i++)
        {
            dp[0][i]=0;
        }


        for(int i=1;i<=n;i++)
        {
            for(int j=0;j<=sum;j++)  //Why j=0 why not j=1??
            {

                    if(arr[i-1]<=j)
                    {
                        dp[i][j]=(dp[i-1][j-arr[i-1]]+dp[i-1][j])%modulo;
                    }
                    else
                    {
                        dp[i][j]=dp[i-1][j]%modulo;
                    }

            }
        }
        return dp[n][sum];


    }
}
```

Coin Change 2 (DP - 22)

14 October 2024 19:40

```java
class Solution {
    public int change(int sum, int[] arr)
    {
        int n=arr.length;
        int[][]dp=new int[n+1][sum+1];
        for(int i=0;i<=n;i++)
        {
            dp[i][0]=1;
        }
        for(int i=1;i<=sum;i++)
        {
            dp[0][i]=0;
        }

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

                    if(arr[i-1]<=j)
                    {
                        dp[i][j]=dp[i][j-arr[i-1]]+dp[i-1][j];
                    }
                    else
                    {
                        dp[i][j]=dp[i-1][j];
                    }

            }
        }
        return dp[n][sum];

    }
}
```

Unbounded Knapsack (DP - 23)

14 October 2024 19:40

```
class Solution{
   static int knapSack(int n, int sum, int val[], int arr[])
   {
      //int n=arr.size();
      int[][]dp=new int[n+1][sum+1];
      for(int i=0;i<=n;i++)
      {
         dp[i][0]=0;
      }
      for(int i=1;i<=sum;i++)
      {
         dp[0][i]=0;
      }

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

               if(arr[i-1]<=j)
               {
                  dp[i][j]=Math.max(val[i-1]+dp[i][j-arr[i-1]],dp[i-1][j]);
               }
               else
               {
                  dp[i][j]=dp[i-1][j];
               }

         }
      }
      return dp[n][sum];

   }
}
```

Rod Cutting Problem | (DP - 24)

14 October 2024 19:41

```
class Solution{
   public int cutRod(int val[], int sum)
   {

      int n=val.length;
      int[]arr=new int[n];
      for(int i=0;i<n;i++)
      {
         arr[i]=i+1;
      }
      int[][]dp=new int[n+1][sum+1];
      for(int i=0;i<=n;i++)
      {
         dp[i][0]=0;
      }
      for(int i=1;i<=sum;i++)
      {
         dp[0][i]=0;
      }

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

               if(arr[i-1]<=j)
               {
                  dp[i][j]=Math.max(val[i-1]+dp[i][j-arr[i-1]],dp[i-1][j]);
               }
               else
               {
                  dp[i][j]=dp[i-1][j];
               }

         }
      }
      return dp[n][sum];

   }
}
```

Best Time to Buy and Sell Stock |(DP-35)

16 October 2024 21:34

Mine Solution

```
class Solution {
    public int maxProfit(int[] arr)
    {
        int mn=arr[0];
        int n=arr.length;
        int ans=0;
        for(int i=1;i<n;i++)
        {
            ans=Math.max(arr[i]-mn,ans);
            if(arr[i]<mn)
            {
                mn=arr[i];
            }
        }
        return ans;
    }
}
```

Buy and Sell Stock - II|(DP-36)

16 October 2024 21:34

#JITNA MAN UTNA KHAREED BECH SAKTE HO

```
class Solution {
    public int maxProfit(int[] arr)
    {
        int ans=0;
        int n=arr.length;
        for(int i=1;i<n;i++)
        {
            if(arr[i]-arr[i-1]>0)
            {
                ans+=arr[i]-arr[i-1];
            }
        }
        return ans;

    }
}
```

#STRIVER KA CODE USING DP

```
class Solution {
    int find(int[] arr,int idx,int n,int buy,int[][]dp)
    {
        if(idx==n)
        {
            return 0;
        }
        if(dp[idx][buy]!=-1)
        {
            return dp[idx][buy];
        }
        int profit=0;
        if(buy==1)
        {
            profit=Math.max(-arr[idx]+find(arr,idx+1,n,0,dp),find(arr,idx+1,n,1,dp));  //0 means no
more buy -arr[idx] means we are investing money to buy stocks
        }
        else
        {
            profit=Math.max(arr[idx]+find(arr,idx+1,n,1,dp),find(arr,idx+1,n,0,dp));
        }
        return dp[idx][buy]=profit;
    }
    public int maxProfit(int[] arr)
    {
        int ans=0;
        int n=arr.length;
        int[][]dp=new int[n+1][2];
        for (int[] row: dp)
        {
            Arrays.fill(row, -1);
        }

        return find(arr,0,n,1,dp);

    }
```

```
}
```

Buy and Sell Stocks III|(DP-37)

16 October 2024 21:34

```
class Solution {
    int find(int[] arr,int idx,int n,int buy,int[][][]dp,int cap)
    {
        if(idx==n||cap==0)
        {
            return 0;
        }
        if(dp[idx][buy][cap]!=-1)
        {
            return dp[idx][buy][cap];
        }
        int profit=0;
        if(buy==1)
        {
            profit=Math.max(-arr[idx]+find(arr,idx+1,n,0,dp,cap),find(arr,idx+
1,n,1,dp,cap));  //0 means no more buy -arr[idx] means we are investing money to buy
stocks
        }
        else
        {
            profit=Math.max(arr[idx]+find(arr,idx+1,n,1,dp,cap-1),find(arr,idx+
1,n,0,dp,cap));
        }
        return dp[idx][buy][cap]=profit;
    }
    public int maxProfit(int[] arr)
    {
        int ans=0;
        int n=arr.length;
        int[][][]dp=new int[n+1][2][3];  //0-n-1,{0,1},{0,1,2 //0 transaction remaining
1 remaining 0r 2 remaining}
        for (int[][] rowouter: dp)
        {
            for(int[]rowinner:rowouter)
            {

                    Arrays.fill(rowinner, -1);


            }

        }

        return find(arr,0,n,1,dp,2);

    }
}
```

Tabulation-----

```
int maxProfit(vector<int>& prices, int n)
{
   // Write your code here.
   vector<vector<int>>dp(n+1,vector<int>(5,-1));
   for(int i=0;i<=4;i++)
```

```
    {
        dp[n][i] = 0;
    }
    for(int i=0;i<=n;i++)
    {
        dp[i][4]=0;
    }
    for(int ind=n-1;ind>=0;ind--)
    {
        for(int ts = 3;ts>=0;ts--)
        {
            if(ts%2==0)
            {
                dp[ind][ts] = max(-prices[ind]+dp[ind+1][ts+1],dp[ind+1][ts]);
            }
            else
            {
                dp[ind][ts] = max(prices[ind]+dp[ind+1][ts+1],dp[ind+1][ts]);
            }
        }
    }
    return dp[0][0];
}
```

Buy and Stock Sell IV |(DP-38)

16 October 2024 21:34

```
class Solution {
    int find(int[] arr,int idx,int n,int buy,int[][][]dp,int cap)
    {
        if(idx==n||cap==0)
        {
            return 0;
        }
        if(dp[idx][buy][cap]!=-1)
        {
            return dp[idx][buy][cap];
        }
        int profit=0;
        if(buy==1)
        {
            profit=Math.max(-arr[idx]+find(arr,idx+1,n,0,dp,cap),find(arr,idx+
1,n,1,dp,cap));  //0 means no more buy -arr[idx] means we are investing money to buy
stocks
        }
        else
        {
            profit=Math.max(arr[idx]+find(arr,idx+1,n,1,dp,cap-1),find(arr,idx+
1,n,0,dp,cap));
        }
        return dp[idx][buy][cap]=profit;
    }
    public int maxProfit(int k, int[] arr)
    {
        int ans=0;
        int n=arr.length;
        int[][][]dp=new int[n+1][2][k+1];  //0-n-1,{0,1},{0,1,2,……k+1 //0 transaction
remaining 1 remaining 0r 2 remaining}
        for (int[][] rowouter: dp)
        {
            for(int[]rowinner:rowouter)
            {

                    Arrays.fill(rowinner, -1);


            }

        }

        return find(arr,0,n,1,dp,k);

    }
}
```

Buy and Sell Stocks With Cooldown|(DP-39)

16 October 2024 21:34

```
class Solution {
    int find(int[] arr,int idx,int n,int buy,int[][]dp)
    {
        if(idx>=n)
        {
            return 0;  // ye kyuki idx+2 bhi ho rha hai bachhe
        }
        if(dp[idx][buy]!=-1)
        {
            return dp[idx][buy];
        }
        int profit=0;
        if(buy==1)
        {
            profit=Math.max(-arr[idx]+find(arr,idx+1,n,0,dp),find(arr,idx+1,n,1,dp));  //0
means no more buy -arr[idx] means we are investing money to buy stocks
        }
        else
        {
            profit=Math.max(arr[idx]+find(arr,idx+2,n,1,dp),find(arr,idx+1,n,0,dp));
//bechte time ek idx aur aage badha do
        }
        return dp[idx][buy]=profit;
    }
    public int maxProfit(int[] arr)
    {
        int ans=0;
        int n=arr.length;
        int[][]dp=new int[n+1][2];
        for (int[] row: dp)
        {
            Arrays.fill(row, -1);
        }

        return find(arr,0,n,1,dp);

    }
}
```

Buy and Sell Stocks With Transaction Fee|(DP-40)

16 October 2024 21:34

```
class Solution {
    int find(int[] arr,int idx,int n,int buy,int[][]dp,int fee)
    {
        if(idx==n)
        {
            return 0;
        }
        if(dp[idx][buy]!=-1)
        {
            return dp[idx][buy];
        }
        int profit=0;
        if(buy==1)
        {
            profit=Math.max(-arr[idx]-fee+find(arr,idx+1,n,0,dp,fee),find(arr,idx+
1,n,1,dp,fee));  //0 means no more buy -arr[idx] means we are investing money to buy
stocks
        }
        else
        {
            profit=Math.max(arr[idx]+find(arr,idx+1,n,1,dp,fee),find(arr,idx+
1,n,0,dp,fee));
        }
        return dp[idx][buy]=profit;
    }
    public int maxProfit(int[] arr, int fee)
    {
        int ans=0;
        int n=arr.length;
        int[][]dp=new int[n+1][2];
        for (int[] row: dp)
        {
            Arrays.fill(row, -1);
        }

        return find(arr,0,n,1,dp,fee);


    }
}
```