

DP Solutions

Solution 1:

Time Complexity : $O(n)$

Space Complexity: $O(n)$

```
import java.io.*;

class Solution {

    static void printTrib(int n){
        int dp[] = new int[n];
        dp[0] = dp[1] = 0;
        dp[2] = 1;

        for (int i = 3; i < n; i++)
            dp[i] = dp[i - 1] + dp[i - 2] + dp[i - 3];

        for (int i = 0; i < n; i++)
            System.out.print(dp[i] + " ");
    }

    public static void main(String args[]){
        int n = 10;
        printTrib(n);
    }
}
```

Solution 2 :

Time Complexity : $O(n)$

Space Complexity: $O(n)$

```
import java.io.*;

class Solution{
```

er.dm.zer03@gmail.com

```
static void _printParenthesis(char str[], int pos, int n, int open, int close) {  
    if(close == n) {  
        for(int i=0;i<str.length;i++)  
            System.out.print(str[i]);  
        System.out.println();  
        return;  
    }  
    else{  
        if(open > close) {  
            str[pos] = ')';  
            _printParenthesis(str, pos+1, n, open, close+1);  
        }  
        if(open < n) {  
            str[pos] = '(';  
            _printParenthesis(str, pos+1, n, open+1, close);  
        }  
    }  
}
```

```
static void printParenthesis(char str[], int n) {  
    if(n > 0)  
        _printParenthesis(str, 0, n, 0, 0);  
    return;  
}
```

```
public static void main (String[] args) {  
    int n = 3;  
    char[] str = new char[2 * n];  
    printParenthesis(str, n);  
}
```

Solution 3 :

Time Complexity : $O(n^2)$
Space Complexity: $O(1)$

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

er.dm.zer03@gmail.com

```
class Solution{

    static int max_profit(int a[],int b[],int n,int fee){
        int i, j, profit;
        int l, r, diff_day = 1, sum = 0;

        b[0]=0;
        b[1]=diff_day;
        for(i=1;i<n;i++){
            l=0;
            r=diff_day;
            sum=0;

            for(j=n-1;j>=i;j--){
                profit=(a[r]-a[l])-fee;
                if(profit>0){
                    sum=sum+profit;
                }
                l++;
                r++;
            }
            if(b[0] < sum){
                b[0] = sum;
                b[1] = diff_day;
            }
            diff_day++;
        }
        return 0;
    }

    public static void main(String args[]){
        int arr[] = { 6, 1, 7, 2, 8, 4 };
        int n = arr.length;
        int[] b = new int[2];
        int tranFee = 2;

        max_profit(arr, b, n, tranFee);
        System.out.println(b[0]+", "+b[1]);
    }
}
```



er.dm.zer03@gmail.com

}

Solution 4 :Time Complexity : $O(n^2)$ Space Complexity: $O(n^2)$

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

```
class Solution {
```

```
    static int LIP(int dp[][], int mat[][], int n,
                  int m, int x, int y){
        if (dp[x][y] < 0) {
            int result = 0;
            if (x == n - 1 && y == m - 1)
                return dp[x][y] = 1;
            if (x == n - 1 || y == m - 1)
                result = 1;
            if (x + 1 < n && mat[x][y] < mat[x + 1][y])
                result = 1 + LIP(dp, mat, n, m, x + 1, y);
            if (y + 1 < m && mat[x][y] < mat[x][y + 1])
                result = Math.max(result, 1 + LIP(dp, mat, n, m, x, y + 1));
            dp[x][y] = result;
        }
        return dp[x][y];
    }
```

```
    static int wrapper(int mat[][], int n, int m){
        int dp[][] = new int[10][10];
        for (int i = 0; i < 10; i++)
            Arrays.fill(dp[i], -1);

        return LIP(dp, mat, n, m, 0, 0);
    }
```

```
    public static void main(String[] args){
        int mat[][] = {
```

er.dm.zer03@gmail.com

```
{ 1, 2, 3, 4 },  
{ 2, 2, 3, 4 },  
{ 3, 2, 3, 4 },  
{ 4, 5, 6, 7 },  
};  
int n = 4, m = 4;  
System.out.println(wrapper(mat, n, m));  
}  
}
```

Solution 5 :

```
public static int helper(int left, int right) {  
    if (left == 0 && right == 0){  
        ans++;  
    }  
    if (left > right){  
        return 0;  
    }  
    if (left > 0){  
        helper(left-1, right);  
    }  
    if (right > 0){  
        helper(left, right-1);  
    }  
    return ans;  
}  
  
// Find possible ways for balanced parentheses  
private static int countWays(int n){  
    // If n is odd no possible valid parentheses  
    if ((n & 1) != 0)  
        return 0;  
    return helper(n / 2, n / 2);  
}
```



er.dm.zer03@gmail.com