LeetCode Dynamic Programming Tagged Solutions (50/424)

1. Longest Palindromic Substring
2. Regular Expression Matching
3. Generate Parentheses
4. Longest Valid Parentheses
5. Trapping Rain Water
6. Wildcard Matching
7. Jump Game II
8. Maximum Subarray
9. Jump Game
10. Unique Paths
11. Unique Paths II
12. Minimum Path
13. Climbing Stairs
14. Edit Distance
15. Maximal Rectangle
16. Scramble String
17. Decode Ways
18. Unique Binary Search Trees II
19. Unique Binary Search Trees
20. Interleaving String
21. Distinct Subsequences
22. Pascal's Triangle
23. Pascal's Triangle II
24. Triangle
25. Best Time to Buy and Sell Stock
26. Best Time to Buy and Sell Stock II
27. Best Time to Buy and Sell Stock III
28. Binary Tree Maximum Path Sum
29. Palindrome Partitioning
30. Palindrome Partitioning II
31. Word Break
32. Word Break II
33. Maximum Product Subarray
34. Dungeon Game
35. Best Time to Buy and Sell Stock IV
36. House Robber
37. House Robber II
38. Maximal Square
39. Number of Digit One
40. Different Ways to Add Parentheses
41. Paint House
42. Ugly Number II
43. Paint House II
44. Paint Fence
45. Perfect Squares
46. Flip Game II
47. Longest Increasing Subsequence
48. Best Time to Buy and Sell Stock CD..
49. Burst Balloons
50. Super Ugly Number (Q. 313)

1. Longest Palindromic Substring

```java
1 usage
static boolean isPalindrome(String s) {
    int low = 0, high = s.length() - 1;
    while (low <= high) {
        if (s.charAt(low) != s.charAt(high))
            return false;
        low++;
        high--;
    }
    return true;
}

1 usage
static void bruteForce(String s) {
    String res = "";
    int maxLen = Integer.MIN_VALUE;
    int n = s.length();
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j <= n; j++) {
            if (j - i >= maxLen) {
                String substr = s.substring(i, j);
                if (isPalindrome(substr)) {
                    int l = substr.length();
                    if (l > maxLen) {
                        maxLen = l;
                        res = substr;
                    }
                }
            }
        }
    }
    System.out.println(res);
}
```

dp[i][j] = whether the substring from index i to j is a palindrome or not

	b(0)	a(1)	b(2)	a(3)	d(4)
b(0)	true	false	true	false	false
a(1)	false	true	false	true	false
b(2)	false	false	true	false	false
a(3)	false	false	false	true	false
d(4)	false	false	false	false	true

```
1 usage
static void dynamicProgramming(String s) {
    int n = s.length();
    int x = 0, y = 0, max = Integer.MIN_VALUE;
    boolean dp[][] = new boolean[n][n];
//      dp[i][j] = whether the substring from index i to j is a palindrome or not
    for (int i = n - 1; i >= 0; i--) {
        for (int j = i; j <= n - 1; j++) {
//              i==j
//              Every single character of a string is a palindrome. Therefore dp[i][j] is true.
            if (i == j)
                dp[i][j] = true;
//              j-i=1
//              We're checking two characters at a time, if s[i]==s[j], then dp[i][j] is true.
            else if (s.charAt(i) == s.charAt(j)) {
                if (j - i == 1) dp[i][j] = true;
                else
//              j-i>=2 => j >= 2 + i
//              Consider "aba" s[0]=s[2], therefore dp[i][j] will be true.
//              If s[i]==s[j], but j-i>=2, dp[i][j] = dp[i+1][j-1].
//              Now the i+1,j-1 coordinates are literally eliminating the first and last characters
//              Since they are already the same, we want to know if the string without them is still
//              a palindrome or no? This result will in turn be any of the above cases or this case, nevertheless,
//              the result has already been calculated.
                    dp[i][j] = dp[i + 1][j - 1];
            }
            if (dp[i][j] && j - i >= max) {
                max = j - i;
                x = i;
                y = j;
            }
        }
    }
    System.out.println(Arrays.deepToString(dp));
    System.out.println(s.substring(x, y - x + 1));
}
```

2. Regular Expression Matching

```java
public class RegularExpressionMatching {

    /*
A regular expression consists of:

Letters A-Z
Numbers 0-9
'*': Matches 0 or more characters
'.': Matches one character
    */

    1 usage
    public static void matchesDP(String S, String R) {
        int m = S.length();
        int n = R.length();
        boolean[][] dp = new boolean[m + 1][n + 1];
        dp[0][0] = true;
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (S.charAt(i - 1) == R.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else if (R.charAt(j - 1) == '.') {
                    dp[i][j] = dp[i - 1][j - 1];
                } else if (R.charAt(j - 1) == '*') {
                    dp[i][j] = dp[i - 1][j] || dp[i][j - 1];
                }
            }
        }
        System.out.println(Arrays.deepToString(dp));
        System.out.println(dp[m][n]);
    }

    public static void main(String[] args) {
        matchesDP( S: "GREATS",  R: "G*T*S");
    }
}
```

		G	*	T	*	S
	true	false	false	false	false	false
G	false	true	true	false	false	false
R	false	false	true	false	false	false
E	false	false	true	false	false	false
A	false	false	true	false	false	false
T	false	false	true	true	true	false
S	false	false	true	false	true	true

Bottom-up equation

Dp[i][j] = true, if i = 0, j=0

Dp[i][j] = false, if i = 0 or j = 0

Dp[i][j] = dp[i-1][j-1], if S[i-1] = R[j-1],
if the characters from both the strings are same

Dp[i][j] = dp[i-1][j-1], if R[j-1] = .
'.' matches one character

Dp[i][j] = dp[i-1][j] or dp[i][j-1] if R[j-1] = *
'*' matches 0 or more characters

*****

If it matches with one character then,

matches(i, j) = matches(i-1, j)

If it matches with no character then,

matches(i, j) = matches(i, j-1)

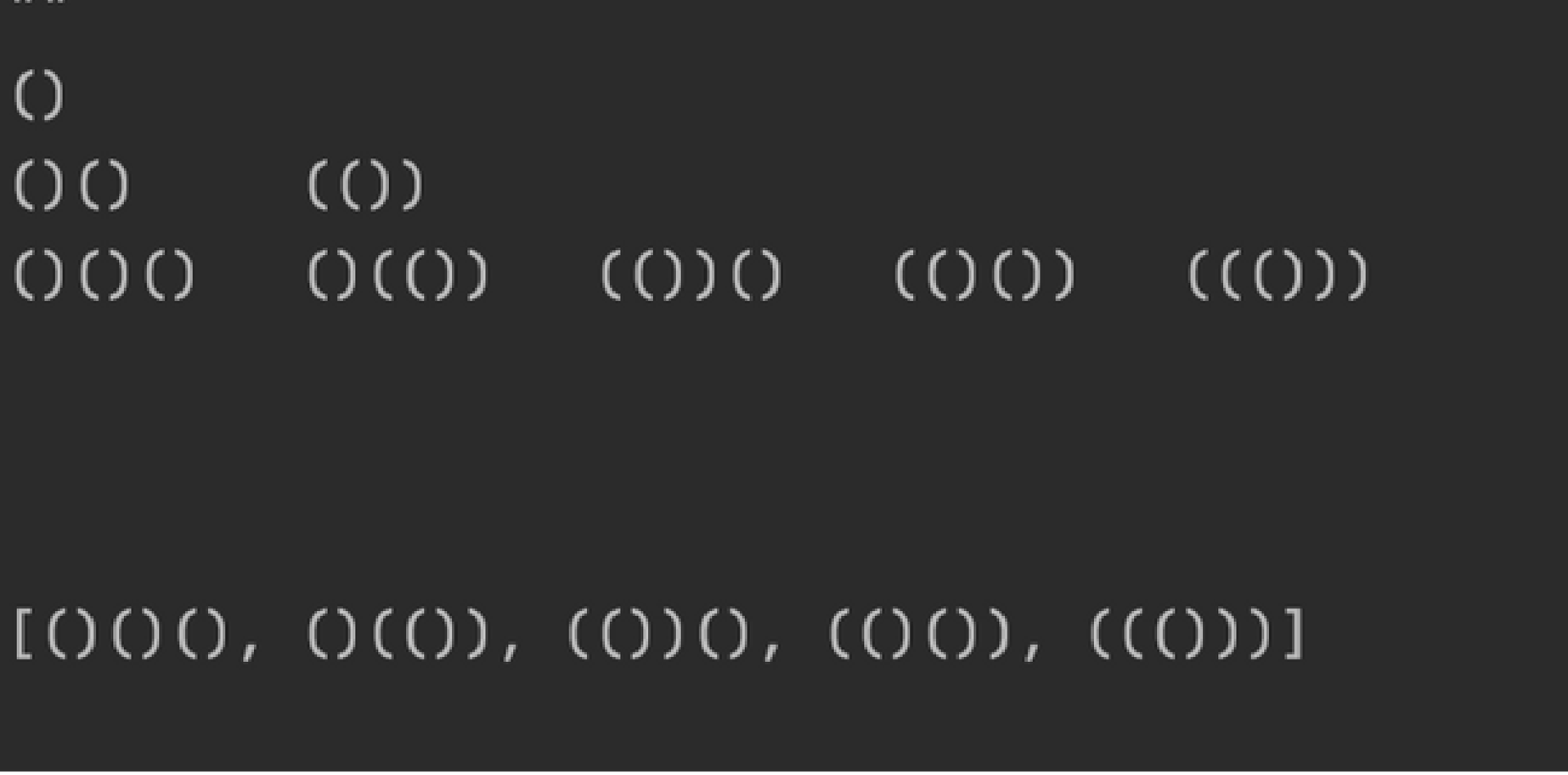
3. Generate Parentheses

```java
package com.sai;

import java.util.ArrayList;
import java.util.List;

public class GenerateParentheses {
    1 usage
    public static List<String> generateParenthesis(int n) {
        List<List<String>> dp = new ArrayList<>();
        List<String> dp0 = new ArrayList<>();
        dp0.add("");
        dp.add(dp0);
        for (int i = 1; i <= n; i++) {
            List<String> cur = new ArrayList<>();
            for (int j = 0; j < i; j++) {
                List<String> str1 = dp.get(j);
                List<String> str2 = dp.get(i - j - 1);
                for (String s1 : str1) {
                    for (String s2 : str2) {
                        cur.add("(" + s1 + ")" + s2);
                    }
                }
            }
            dp.add(cur);
        }
        return dp.get(n);
    }

    public static void main(String[] args) {
        System.out.println(generateParenthesis( n: 5));
    }
}
```



Better Solution Using BackTracking:

```java
public List < String > generateParenthesis(int n) {
    List < String > result = new LinkedList < String > ();
    dfs("", n, n, result);
    return result;
}
private void process(String prefix, int left, int right, List < String > result) {
    if (left == 0 && right == 0) {
        result.add(prefix);
        return;
    }
    if (left > 0) {
        dfs(prefix + '(', left - 1, right, result);
    }
    if (left < right) {
        dfs(prefix + ')', left, right - 1, result);
    }
}
public static void main(String...s) {
    Main sol = new Main();
    System.out.println(sol.generateParenthesis(3));
```

4. Longest Valid Parentheses

```java
public class LongestValidParentheses {
    1 usage
    static void lvp(String s) {
        String s2 = ")" + s;
        int n = s.length();
        int[] dp = new int[n];
        Arrays.fill(dp, val: 0);
        for (int i = 1; i < n; ++i)
            if (s2.charAt(i) == ')' && s2.charAt(i - dp[i - 1] - 1) == '(')
                dp[i] = dp[i - 1] + dp[i - dp[i - 1] - 2] + 2;
        System.out.println(Arrays.stream(dp).max().getAsInt());
    }

    public static void main(String[] args) {
        lvp( s: "()()()("); // 6
    }
}
```

dp[i] would store the length of the longest valid substring ending at s[i].

```
[0, 0, 2, 0, 4, 0, 6]
[(, ), (, ), (, ), (]
```

If s[i]==')',

- If s[i-1]==')', the string is like ".....))", then, this s[i] would make a valid substring only if s at i-dp[i-1]-1 is '(' because the substring from i-dp[i-1] to i-1 will itself be a valid substring and we can only add on it from both the ends. So, if s[i-dp[i-1]-1]=='(' them, dp[i] = dp[i-1]+2+dp[i-dp[i-1]-2].

5. Trapping Rain Water

```java
public class TrappingRainWater {

    1 usage
    public static int trap(int[] height) {
        int max1 = 0;
        int left[] = new int[height.length];
        for (int i = 0; i < height.length; i++) {
            if (max1 < height[i]) {
                max1 = height[i];
            }
            left[i] = max1;
        }
        int max2 = 0;
        int right[] = new int[height.length];
        for (int i = height.length - 1; i >= 0; i--) {
            if (max2 < height[i]) {                          left
                max2 = height[i];                            [2, 2, 2]
            }
            right[i] = max2;                                 right
        }                                                    [2, 2, 2]
        int trap = 0;
        for (int i = 0; i < height.length; i++) {
            trap += Math.min(left[i], right[i]) - height[i];
        }
        return trap;
    }

    public static void main(String[] args) {
        System.out.println(trap(new int[]{2, 0, 2})); // 2
    }
}
```

6. Wildcard Matching

```java
public class WildCartMatching {

    1 usage
    public static void isMatch(String s, String p) {
        int m = s.length();
        int n = p.length();
        boolean dp[][] = new boolean[m + 1][n + 1];
        dp[0][0] = true;
        for (int j = 1; j <= n; j++) {
            if (p.charAt(j - 1) == '*') {
                dp[0][j] = dp[0][j - 1];
            }
        }
        for (int i = 1; i < m + 1; i++) {
            for (int j = 1; j < n + 1; j++) {
                if (p.charAt(j - 1) == '?' || p.charAt(j - 1) == s.charAt(i - 1)) {
                    dp[i][j] = dp[i][j] || dp[i - 1][j - 1];
                } else if (p.charAt(j - 1) == '*') {
                    dp[i][j] = dp[i - 1][j] || dp[i][j - 1];
                }
            }
        }
        System.out.println(dp[m][n]);
    }

    public static void main(String[] args) {
        isMatch( s: "aa",  p: "a"); // false
    }
}
```

```
true   false

false  true

false  false
```

7. Jump Game 2

```java
public class JumpGame2 {

    1 usage
    public static int jump(int[] nums) {
        int n = nums.length;
        int[] dp = new int[n + 1];
        Arrays.fill(dp, Integer.MAX_VALUE);
        dp[0] = 0;
        for (int i = 1; i < n; i++) {
            for (int j = 0; j < i; j++) {
                if (j + nums[j] >= i) {
                    dp[i] = Math.min(dp[i], dp[j] + 1);
                }
            }
        }
        return dp[n - 1];
    }

    public static void main(String[] args) {
        System.out.println(jump(new int[]{2, 3, 1, 1, 4})); // 2
    }
```

```
[0, 1, 1, 2, 2, 2147483647]
```

8. Maximum Subarray
(Can be solved using Kadane Algorithm)

```java
public class MaximumSubArray {
    1 usage
    public static int maxSubArray(int[] a) {
        int n = a.length;
        int dp[] = new int[n];
        dp[0] = a[0];
        int big = dp[0];
        for (int i = 1; i < n; i++) {
            dp[i] = Math.max(dp[i - 1] + a[i], a[i]);
            if (dp[i] > big) big = dp[i];
        }
        return big;
    }

    public static void main(String[] args) {
        System.out.println(maxSubArray(new int[]{-2, 1, -3, 4, -1, 2, 1, -5, 4})); // 6
    }
}
```

[-2, 1, -2, 4, 3, 5, 6, 1, 5]

10. Unique Paths

```java
public class UniquePaths {

    1 usage
    static int uniquePaths(int m, int n) {
        int[][] dp = new int[m][n];
        for (int[] dp1 : dp) Arrays.fill(dp1, val: 1);
        for (int i = 1; i < m; i++)
            for (int j = 1; j < n; j++)
                dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
        return dp[m - 1][n - 1];
    }

    1 usage
    static int uniquePaths1D(int m, int n) {
        int[] dp = new int[n];
        Arrays.fill(dp, val: 1);
        for (int i = 1; i < m; i++)
            for (int j = 1; j < n; j++)
                dp[j] += dp[j - 1];
        return dp[n - 1];
    }

    public static void main(String[] args) {
        System.out.println(uniquePaths( m: 3,  n: 2)); // 3
        System.out.println(uniquePaths1D( m: 3,  n: 2)); // 3
    }

}
```

```
1   1
1   2
1   3
```

```
[1, 3]
```

11. Unique Paths 2

```java
public class UniquePaths2 {

    1 usage
    static int uniquePathsWithObstacles1(int[][] obstacleGrid) {
        int m = obstacleGrid.length;
        int n = obstacleGrid[0].length;
        int[][] dp = new int[m + 1][n + 1];
        dp[0][1] = 1;
        // or dp[1][0] = 1
        for (int i = 1; i <= m; ++i)
            for (int j = 1; j <= n; ++j)
                if (obstacleGrid[i - 1][j - 1] == 0)
                    dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
        return dp[m][n];
    }

    1 usage
    static int uniquePathsWithObstacles(int[][] obstacleGrid) {
        int m = obstacleGrid.length;
        int n = obstacleGrid[0].length;
        int[] dp = new int[n];
        dp[0] = 1;
        for (int i = 0; i < m; ++i)
            for (int j = 0; j < n; ++j)
                if (obstacleGrid[i][j] == 1)
                    dp[j] = 0;
                else if (j > 0)
                    dp[j] += dp[j - 1];
        return dp[n - 1];
    }

    public static void main(String[] args) {
        System.out.println(uniquePathsWithObstacles1(new int[][]{
                {0, 0, 0},
                {0, 1, 0},
                {0, 0, 0}
        })); // 2
        System.out.println(uniquePathsWithObstacles(new int[][]{
                {0, 0, 0},
                {0, 1, 0},
                {0, 0, 0}
        })); // 2
    }
}
```

```
0 1 0 0
0 1 1 1
0 1 0 1
0 1 1 2
####
1 1 2
```

12. Minimum Path

```java
public class MinimumPath {

    1 usage
    static int minPathSum(int[][] dp) {
        int m = dp.length;
        int n = dp[0].length;
        for (int i = 0; i < m; ++i)
            for (int j = 0; j < n; ++j)
                if (i > 0 && j > 0)
                    dp[i][j] += Math.min(dp[i - 1][j], dp[i][j - 1]);
                else if (i > 0)
                    dp[i][0] += dp[i - 1][0];
                else if (j > 0)
                    dp[0][j] += dp[0][j - 1];
        return dp[m - 1][n - 1];
    }

    public static void main(String[] args) {
        System.out.println(minPathSum(new int[][]{
                {1, 3, 1},
                {1, 5, 1},
                {4, 2, 1}
        })); // 7
    }

}
```

```
1 3 1
1 5 1
4 2 1
#####
1 4 5
2 7 6
6 8 7
7
```

13. Climbing Stairs

```java
public class ClimbingStairs {

    1 usage
    static int climbStairs(int n) {
        int[] dp = new int[n + 1];
        dp[0] = 1;
        dp[1] = 1;
        for (int i = 2; i <= n; ++i)
            dp[i] = dp[i - 1] + dp[i - 2];
        return dp[n];
    }

    public static void main(String[] args) {
        System.out.println(climbStairs( n: 2));
    }

}
```

```
[ 1, 1, 2 ]
###
2
```

14. Edit Distance

```java
public class EditDistance {

    1 usage
    static int minDistance(String word1, String word2) {
        int m = word1.length();
        int n = word2.length();
        if (m == 0) return n;
        if (n == 0) return m;
        int[][] dp = new int[m + 1][n + 1];
        for (int i = 1; i <= m; ++i)
            dp[i][0] = i;
        for (int j = 1; j <= n; ++j)
            dp[0][j] = j;
        for (int i = 1; i <= m; ++i)
            for (int j = 1; j <= n; ++j)
                if (word1.charAt(i - 1) == word2.charAt(j - 1))
                    dp[i][j] = dp[i - 1][j - 1];
                else
                    dp[i][j] = Math.min(dp[i - 1][j - 1], Math.min(dp[i - 1][j], dp[i][j - 1])) + 1;
        return dp[m][n];
    }

    public static void main(String[] args) {
        System.out.println(minDistance("horse", "ros")); // 3
    }

}
```

		r	o	s
h	0	1	2	
o	1	1	2	
r	2	2	1	
s	3	2	2	
e	4	3	3	

15. Maximal Rectangle

```java
public class MaximalRectangle {

    1 usage
    static int maximalRectangle(char[][] matrix) {
        if (matrix.length == 0) return 0;
        int n = matrix.length;
        int m = matrix[0].length;
        int[][] dp = new int[n][m];
        int maxArea = 0;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                if (i == 0)
                    dp[i][j] = matrix[i][j] == '1' ? 1 : 0;
                else
                    dp[i][j] = matrix[i][j] == '1' ? (dp[i - 1][j] + 1) : 0;
                int min = dp[i][j];
                for (int k = j; k >= 0; k--) {
                    if (min == 0) break;
                    if (dp[i][k] < min) min = dp[i][k];
                    maxArea = Math.max(maxArea, min * (j - k + 1));
                }
            }
        }
        return maxArea;
    }

    public static void main(String[] args) {
        System.out.println(maximalRectangle(
                new char[][]{
                        {'1', '0', '1', '0', '0'},
                        {'1', '0', '1', '1', '1'},
                        {'1', '1', '1', '1', '1'},
                        {'1', '0', '0', '1', '0'}
                }
        )); // 6
    }

}
```

1	0	1	0	0
2	0	2	1	1
3	1	3	2	2
4	0	0	3	0

height of rectangle we're increasing here as we can see

16. Scramble String

```java
public class ScrambleString {

    1 usage
    public static boolean isScramble(String s1, String s2) {
        int n = s1.length();
        if (n != s2.length()) {
            return false;
        }
        boolean[][][] dp = new boolean[n + 1][n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (s1.charAt(i) == s2.charAt(j)) {
                    dp[1][i][j] = true;
                }
            }
        }
        for (int l = 2; l <= n; l++) {
            for (int i = 0; i <= n - l; i++) {
                for (int j = 0; j <= n - l; j++) {
                    for (int k = 1; k < l; k++) {
                        if ((dp[k][i][j] && dp[l - k][i + k][j + k]) ||
                                (dp[k][i][j + l - k] && dp[l - k][i + k][j])) {
                            dp[l][i][j] = true;
                            break;
                        }
                    }
                }
            }
        }
        return dp[n][0][0];
    }

    public static void main(String[] args) {
        System.out.println(isScramble("great", "rgeat")); // true
    }

}
```

```
[[[false, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false]],
 [[false, true, false, false, false],
 [true, false, false, false, false],
 [false, false, true, false, false],
 [false, false, false, true, false],
 [false, false, false, false, true]],
 [[true, false, false, false, false],
 [false, false, false, false, false],
 [false, false, true, false, false],
 [false, false, false, true, false],
 [false, false, false, false, false]],
 [[true, false, false, false, false],
 [false, false, false, false, false],
 [false, false, true, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false]],
 [[true, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false]],
 [[true, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false],
 [false, false, false, false, false]]]
```

17. Decode Ways

```java
public class DecodeWays {

    2 usages
    static boolean isSafe(char c1) {
        return c1 != '0';
    }

    1 usage
    static boolean isSafe(char c1, char c2) {
        return c1 == '1' || (c1 == '2' && c2 <= '6');
    }

    1 usage
    static int numDecodings(String s1) {
        int n = s1.length();
        int[] dp = new int[n + 1];
        dp[n] = 1;
        char[] s = s1.toCharArray();
        if (isSafe(s[n - 1])) dp[n - 1] = 1;
        for (int i = n - 2; i >= 0; --i) {
            if (isSafe(s[i]))
                dp[i] += dp[i + 1];
            if (isSafe(s[i], s[i + 1]))
                dp[i] += dp[i + 2];
        }
        return dp[0];
    }

    public static void main(String[] args) {
        System.out.println(numDecodings( s1: "12")); // 2
    }

}
```

```
[ 2, 1, 1 ]
2
```

18. Unique Binary Search Trees 2

```
1 usage
public List<TreeNode> generateTrees(int n) {
    List<TreeNode>[] result = new List[n + 1];
    result[0] = new ArrayList<>();
    if (n == 0) return result[0];
    result[0].add(null);
    for (int i = 1; i <= n; i++) {
        result[i] = new ArrayList<>();
        for (int j = 0; j < i; j++) {
            List<TreeNode> lefts = result[j];
            List<TreeNode> rights = result[i - 1 - j];
            for (TreeNode left : lefts) {
                for (TreeNode right : rights) {
                    TreeNode root = new TreeNode( val: j + 1);
                    root.left = left;
                    root.right = copyTree(right,  offset: j + 1)
                    result[i].add(root);
                }
            }
        }
    }
    return result[n];
}

3 usages
private TreeNode copyTree(TreeNode root, int offset) {
    if (root == null) return null;
    TreeNode node = new TreeNode( val: root.val + offset);
    node.left = copyTree(root.left, offset);
    node.right = copyTree(root.right, offset);
    return node;
}

1 usage
void print(List<TreeNode> nodes) {
    for (TreeNode node : nodes) {
        preOrder(new TreeNode(node.val, node.left, node.right));
        System.out.println();
    }
```

```
1 2 3
1 3 2
2 1 3
3 1 2
3 2 1
```

19. Unique Binary Search Trees

```java
public class UniqueBinarySearchTrees {

    4 usages
    class TreeNode {
        2 usages
        int val;
        1 usage
        TreeNode left;
        1 usage
        TreeNode right;

        TreeNode() {
        }

        TreeNode(int val) {
            this.val = val;
        }

        TreeNode(int val, TreeNode left, TreeNode right) {
            this.val = val;
            this.left = left;
            this.right = right;
        }
    }

    1 usage
    static int numTrees(int n) {
        int[] dp = new int[n + 1];
        dp[0] = 1;
        dp[1] = 1;
        for (int i = 2; i <= n; ++i)
            for (int j = 0; j < i; ++j)
                dp[i] += dp[j] * dp[i - j - 1];
        return dp[n];
    }

    ;

    public static void main(String[] args) {
        System.out.println(numTrees( n: 3)); // 5
    }
}
```

```
[ 1, 1, 2, 5 ]
5
```

20. Interleaving String

```java
public class InterleavingString {

    1 usage
    static boolean isInterleave(String s1, String s2, String s3) {
        int m = s1.length();
        int n = s2.length();
        if (m + n != s3.length()) return false;
        boolean[] dp = new boolean[n + 1];
        for (int i = 0; i <= m; ++i)
            for (int j = 0; j <= n; ++j)
                if (i == 0 && j == 0)
                    dp[j] = true;
                else if (i == 0)
                    dp[j] = dp[j - 1] && s2.charAt(j - 1) == s3.charAt(j - 1);
                else if (j == 0)
                    dp[j] = dp[j] && s1.charAt(i - 1) == s3.charAt(i - 1);
                else
                    dp[j] = dp[j] && s1.charAt(i - 1) == s3.charAt(i + j - 1) ||
                            dp[j - 1] && s2.charAt(j - 1) == s3.charAt(i + j - 1);
        return dp[n];
    }

    ;

    public static void main(String[] args) {
        System.out.println(isInterleave("aabcc", "dbbca", "aadbbcbcac")); // true
    }

}
```

```
[ false, false, false, true, false, true ]
true
```

21. Distinct Subsequences

```java
class DistinctSubsequences {

    1 usage
    static int numDistinct(String s, String t) {
        int m = s.length();
        int n = t.length();
        int[] dp = new int[n + 1];
        dp[0] = 1;
        for (int i = 1; i <= m; ++i)
            for (int j = n; j >= 1; --j)
                if (s.charAt(i - 1) == t.charAt(j - 1))
                    dp[j] += dp[j - 1];
        return dp[n];
    }

    ;

    public static void main(String[] args) {
        System.out.println(numDistinct( s: "rabbbit",  t: "rabbit")); // 3
    }

}
```

```
[1, 1, 1, 3, 3, 3, 3]
3
```

22. Pascal's Triangle

```java
public class PascalsTriangle {

    1 usage
    public static List<int[]> generate(int numRows) {
        int[][] dp = new int[numRows][];
        for (int i = 0; i < numRows; i++) {
            int[] row = new int[i + 1];
            row[0] = 1;
            row[i] = 1;
            for (int j = 1; j < i; j++) {
                row[j] = dp[i - 1][j - 1] + dp[i - 1][j];
            }
            dp[i] = row;
        }
        return Arrays.asList(dp);
    }

    public static void main(String[] args) {
        List<int[]> x = generate( numRows: 3);
        x.stream().forEach(y -> {
            System.out.print(Arrays.toString(y) + "  ");
            System.out.println();
        });

    }

}
```

```
[1]
[1, 1]
[1, 2, 1]
```

23. Pascal's Triangle 2

```java
public class PascalsTriangle2 {

    1 usage
    static int[] getRow(int rowIndex) {
        int[] dp = new int[rowIndex + 1];
        Arrays.fill(dp, val: 1);
        for (int i = 2; i < rowIndex + 1; ++i)
            for (int j = 1; j < i; ++j)
                dp[i - j] += dp[i - j - 1];
        return dp;
    }

    public static void main(String[] args) {
        System.out.println(Arrays.toString(getRow( rowIndex: 3)));
//          [1, 3, 3, 1]
    }

}
```

24. Triangle

```java
public class Triangle {

    1 usage
    static int minimumTotal(int[][] triangle) {
        for (int i = triangle.length - 2; i >= 0; --i)
            for (int j = 0; j <= i; ++j)
                triangle[i][j] += Math.min(triangle[i + 1][j],
                        triangle[i + 1][j + 1]);
        return triangle[0][0];
    }

    public static void main(String[] args) {
        System.out.println(minimumTotal(new int[][]{
                {2},
                {3, 4},
                {6, 5, 7},
                {4, 1, 8, 3}
        }));
        // 11
    }
}
```

```
[[11],
 [9, 10],
 [7, 6, 10],
 [4, 1, 8, 3]]
```

25. Best Time to Buy and Sell Stock

```java
public class BestTimeToBuyAndSellStock {

    1 usage
    public static int maxProfit(int[] prices) {
        int res = 0;
        int[] dp = new int[prices.length];
        dp[0] = prices[0];
        for (int i = 1; i < prices.length; i++) {
            dp[i] = Math.min(dp[i - 1], prices[i]);
        }
        for (int i = 0; i < prices.length; i++) {
            res = Math.max(res, prices[i] - dp[i]);
        }
        return res;
    }

    public static void main(String[] args) {
        System.out.println(maxProfit(new int[]{7, 1, 5, 3, 6, 4})); // 5
    }

}
```

[7, 1, 1, 1, 1, 1]

26. Best Time to Buy and Sell Stock 2

```java
// dp[i] represents the maximum money can make until index i
// 0 -> buy
// 1 -> sell

                                                                    [[-7, 0],
    1 usage                                                          [-6, 0],
    public static int maxProfit(int[] prices) {                      [-4, 0],
        int[][] dp = new int[prices.length][2];                      [-3, 0],
                                                                     [-1, 0]]
        dp[0][0] = -prices[0];
        dp[0][1] = 0;

        for (int i = 1; i < prices.length; i++) {
            dp[i][0] = Math.max(dp[i - 1][0], dp[i - 1][1] - prices[i]);
            dp[i][1] = Math.max(dp[i - 1][1], dp[i - 1][0] + prices[i]);
        }

        return Math.max(dp[prices.length - 1][0], dp[prices.length - 1][1]);
    }


    public static void main(String[] args) {
        System.out.println(maxProfit(new int[]{7, 6, 4, 3, 1})); // 0
    }

}
```

27. Best Time to Buy and Sell Stock 3

```java
public class BestTimeToBuyAndSellStock3 {

    1 usage
    public static int maxProfit(int[] prices) {
        int n = prices.length;
        int dp[][] = new int[3][n];

        for (int i = 1; i <= 2; i++) {
            int maxDiff = dp[i - 1][0] - prices[0];
            for (int j = 1; j < n; j++) {
                dp[i][j] = Math.max(dp[i][j - 1], prices[j] + maxDiff);
                maxDiff = Math.max(maxDiff, dp[i - 1][j] - prices[j]);
            }
        }

        return dp[2][n - 1];
    }

    public static void main(String[] args) {
        System.out.println(maxProfit(new int[]{7, 6, 4, 3, 1})); // 0
    }

}
```

28. Binary Tree Maximum Path Sum

```java
public class BinaryTreeMaximumPathSum {

    3 usages
    static int res = Integer.MIN_VALUE;

    1 usage
    public static int maxPathSum(TreeNode root) {
        solve(root);
        return res;
    }

    3 usages
    public static int solve(TreeNode root) {
        if (root == null) return 0;
        int left = solve(root.left);
        int right = solve(root.right);
        int temp = Math.max(Math.max(left, right) + root.val, root.val);
        int ans = Math.max(temp, left + right + root.val);
        res = Math.max(ans, res);
        return temp;
    }

    public static void main(String[] args) {
        TreeNode t = new TreeNode( val: 1);
        t.left = new TreeNode( val: 2);
        t.right = new TreeNode( val: 3);
        System.out.println(maxPathSum(t)); // 6
    }

}
```

29. Palindrome Partitioning

```
1 usage
public static List<List<String>> partition(String s) {
    List<List<String>> res = new ArrayList<>();
    boolean[][] dp = new boolean[s.length()][s.length()];
    for (int i = 0; i < s.length(); i++) {
        for (int j = 0; j <= i; j++) {
            if (s.charAt(i) == s.charAt(j) && (i - j <= 2 || dp[j + 1][i - 1])) {
                dp[j][i] = true; // palindromic string from j to i
            }
        }
    }
    helper(res, new ArrayList<>(), dp, s,  pos: 0);
    return res;
}

2 usages
private static void helper(List<List<String>> res, List<String> path, boolean[][] dp, String s, int pos) {
    if (pos == s.length()) {
        res.add(new ArrayList<>(path));
        return;
    }

    for (int i = pos; i < s.length(); i++) {
        if (dp[pos][i]) {
            path.add(s.substring(pos, i + 1));
            helper(res, path, dp, s,  pos: i + 1);
            path.remove( index: path.size() - 1);
        }
    }
}

public static void main(String[] args) {
    System.out.println(partition( s: "aab")); // [[a, a, b], [aa, b]]
}
```

```
[[true, true, false],
 [false, true, false],
 [false, false, true]]
```

30. Palindrome Partitioning 2

```
1 usage
static int minCut(String s) {
    int n = s.length();
    int[] cut = new int[n];
    boolean[][] dp = new boolean[n][n];
    for (int i = 0; i < n; i++) {
        int min = i;
        for (int j = 0; j < i + 1; j++) {
            if (s.charAt(j) == s.charAt(i) && (j + 1 > i - 1 || dp[j + 1][i - 1])) {
                dp[j][i] = true;
                if (j == 0) min = 0;
                else min = Math.min(min, cut[j - 1] + 1);
            }
        }                                                    [0, 0, 1]
        cut[i] = min;
    }
    return cut[n - 1];
}

public static void main(String[] args) {
    System.out.println(minCut( s: "aab")); // 1
}
```

31. Word Break

```
1 usage
static boolean wordBreak(String s, Set<String> wordDict) {
    int n = s.length();
    Set<String> wordSet = new HashSet<>(wordDict);
    boolean[] dp = new boolean[n + 1];
    dp[0] = true;
    for (int i = 1; i <= n; ++i)
        for (int j = 0; j < i; ++j)
            if (dp[j] && wordSet.contains(s.substring(j, i))) {
                dp[i] = true;
                break;
            }
    return dp[n];
}

public static void main(String[] args) {
    System.out.println(wordBreak( s: "leetcode",
            new HashSet<>(Arrays.asList("leet", "code"))
    ));
}
```

```
[
true, false, false,
false, true, false,
false, false, true
]
true
```

32. Word Break 2

```java
public class WordBreak2 {

    1 usage
    public static List<String> wordBreak(String s, List<String> wordDict) {
        HashSet set = new HashSet();
        boolean[] dp = new boolean[s.length() + 1];
        dp[0] = true;
        for (int i = 0; i < wordDict.size(); i++) {
            set.add(wordDict.get(i));
        }
        List result = new ArrayList();
        dfs( path: "", result, index: 0, s, dp, set);
        return result;
    }

    2 usages
    static void dfs(String path, List<String> result, int index, String s, boolean[] dp, Set set) {
        if (index == s.length()) {
            result.add(path);
            return;
        }
        for (int j = index; j <= s.length(); j++) {
            if (dp[index] && set.contains(s.substring(index, j))) {
                dp[j] = true;
                String temp = path;
                path += s.substring(index, j);
                if (j != s.length()) path += " ";
                dfs(path, result, j, s, dp, set);
                path = temp;
            }
        }
    }

    public static void main(String[] args) {
        System.out.println(wordBreak( s: "catsanddog", Arrays.asList("cat", "cats", "and", "sand", "dog")));
//        [cat sand dog, cats and dog]
    }

}
```

[true, false, false, true, true, false, false, true, false, false, true]

33. Maximum Product Subarray

```java
public class MaximumProductSubarray {

//    the maximum product will always lie either from the starting of the array or from the end of the array.

    1 usage
    static int maxProduct(int[] nums) {
        int ans, prevMin, prevMax;
        ans = prevMin = prevMax = nums[0];
        for (int i = 1; i < nums.length; i++) {
            int mini = prevMin * nums[i];
            int maxi = prevMax * nums[i];
            prevMin = Math.min(nums[i], Math.min(mini, maxi));
            prevMax = Math.max(nums[i], Math.max(mini, maxi));
            ans = Math.max(ans, prevMax);
        }
        return ans;
    }


    public static void main(String[] args) {
        System.out.println(maxProduct(new int[]{2, 3, -2, 4})); // 6
    }

}
```

34. Dungeon Game

```
1 usage
static int calculateMinimumHP(int[][] dungeon) {
    int m = dungeon.length;
    int n = dungeon[0].length;
    int[] dp = new int[n + 1];
    Arrays.fill(dp, Integer.MAX_VALUE);
    dp[n - 1] = 1;
    for (int i = m - 1; i >= 0; --i)
        for (int j = n - 1; j >= 0; --j) {
            dp[j] = Math.min(dp[j], dp[j + 1]) - dungeon[i][j];
            dp[j] = Math.max(dp[j], 1);
        }
    return dp[0];                                [7, 5, 2, 2147483647]
}

public static void main(String[] args) {
    System.out.println(calculateMinimumHP(new int[][]{
            {-2, -3, 3},
            {-5, -10, 1},
            {10, 30, -5}
    })); //  7
}
```

35. Best Time to Buy and Sell Stock IV

```
1 usage
static int maxProfit(int k, int[] prices) {
    if (k >= (prices.length / 2)) {
        int sell = 0;
        int hold = Integer.MIN_VALUE;
        for (int p : prices) {
            sell = Math.max(sell, hold + p);
            hold = Math.max(hold, sell - p);
        }
        return sell;
    }
    int[] sell = new int[k + 1];
    int[] hold = new int[k + 1];
    Arrays.fill(hold, Integer.MAX_VALUE);
    for (int price : prices)
        for (int i = k; i > 0; --i) {
            sell[i] = Math.max(sell[i], hold[i] + price);
            hold[i] = Math.max(hold[i], sell[i - 1] - price);
        }
    return sell[k];
}

public static void main(String[] args) {
    System.out.println(maxProfit( k: 2, new int[]{2, 4, 1})); // 2
}
```

36. House Robber

```java
static int rob(int[] nums) {
    if (nums == null || nums.length == 0) {
        return 0;
    }
    int[] dp = new int[nums.length + 1];
    dp[0] = 0;
    dp[1] = nums[0];
    for (int i = 2; i <= nums.length; i++) {
        dp[i] = Math.max(nums[i - 1] + dp[i - 2], dp[i - 1]);
    }
    return dp[nums.length];
}

static int robSpaceOptimized(int[] nums) {
    int a = 0, b = nums[0];
    for (int i = 1; i < nums.length; ++i) {
        int c = Math.max(nums[i] + a, b);
        a = b;
        b = c;
    }
    return b;
}

public static void main(String[] args) {
    System.out.println(rob(new int[]{1, 2, 3, 1})); // 4
}
```

[0, 1, 2, 4, 4]

37. House Robber 2

```java
2 usages
static int robH(int[] nums, int l, int r) {
    int prev1 = 0; // dp[i - 1]
    int prev2 = 0; // dp[i - 2]
    for (int i = l; i <= r; ++i) {
        int dp = Math.max(prev1, prev2 + nums[i]);
        prev2 = prev1;
        prev1 = dp;
    }
    return prev1;
}

1 usage
static int rob(int[] nums) {
    if (nums.length == 0) return 0;
    if (nums.length < 2) return nums[0];
    return Math.max(robH(nums, l: 0, r: nums.length - 2), robH(nums, l: 1, r: nums.length - 1));
}

public static void main(String[] args) {
    System.out.println(rob(new int[]{1, 2, 1})); // 2
}
```

```java
public static int rob1(int[] nums) {
    if (nums == null || nums.length == 0) {
        return 0;
    }
    if (nums.length == 1) {                                    [0, 1, 2, 2]
        return nums[0];                                        [0, 0, 2, 2]
    }                                                          2
    int n = nums.length;

    // case-1: Rob the first house, not the last one.
    int[] dp = new int[n + 1];
    dp[0] = 0;
    dp[1] = nums[0];
    for (int i = 2; i < n; i++) {
        dp[i] = Math.max(dp[i - 1], dp[i - 2] + nums[i - 1]);
    }
    dp[n] = dp[n - 1]; // not robbing last one

    // case-2: Not rob the first, might or may not rob  the last one
    int[] dp2 = new int[n + 1];
    dp2[0] = 0;
    dp2[1] = 0;
    for (int i = 2; i < n + 1; i++) {
        dp2[i] = Math.max(dp2[i - 1], dp2[i - 2] + nums[i - 1]);
    }
    return Math.max(dp[n], dp2[n]);
}
```

38. Maximal Square

```java
public class MaximalSquare {

    1 usage
    static int maximalSquare(char[][] matrix) {
        if (matrix.length == 0) return 0;
        int m = matrix.length;
        int n = matrix[0].length;
        int[] dp = new int[n + 1];
        int maxLength = 0;
        int prev = 0;
        for (int i = 0; i < m; ++i)
            for (int j = 0; j < n; ++j) {
                int cache = dp[j];
                if (i == 0 || j == 0 || matrix[i][j] == '0')
                    dp[j] = matrix[i][j] == '1' ? 1 : 0;
                else
                    dp[j] = Math.min(prev, Math.min(dp[j], dp[j - 1])) + 1;
                maxLength = Math.max(maxLength, dp[j]);
                prev = cache;
            }
        return maxLength * maxLength;
    }

    public static void main(String[] args) {
        System.out.println(maximalSquare(new char[][]{
                {'1', '0', '1', '0', '0'},
                {'1', '0', '1', '1', '1'},
                {'1', '1', '1', '1', '1'},
                {'1', '0', '0', '1', '0'}
        })); // 4
    }

}
```

[1, 0, 0, 1, 0, 0]

39. Number of Digit One

```java
private static int[] getLimitArray(int n) {
    String num = n + "";
    int[] limits = new int[num.length()];
    for (int i = 0; i < num.length(); i++) {
        limits[i] = Character.getNumericValue(num.charAt(i));
    }
    return limits;
}

1 usage
public static int countDigitOne(int n) {
    int[] limits = getLimitArray(n);
    int[][][] dp = new int[limits.length + 1][2][limits.length + 2];

    for (int index = limits.length; index >= 0; index--) {
        for (int flag = 0; flag <= 1; flag++) {
            for (int count = limits.length; count >= 0; count--) {
                if (index == limits.length) {
                    dp[index][flag][count] = count;
                } else {
                    int result = 0;
                    if (flag == 1) {
                        for (int i = 0; i <= limits[index]; i++) {
                            int newCount = count;
                            if (i == 1)
                                newCount++;
                            if (i == limits[index]) {
                                result += dp[index + 1][1][newCount];
                            } else {
                                result += dp[index + 1][0][newCount];
                            }
                        }
                    } else {
                        for (int i = 0; i <= 9; i++) {
                            int newCount = count;
                            if (i == 1)
                                newCount++;
                            result += dp[index + 1][0][newCount];
                        }
                    }
                    dp[index][flag][count] = result;
                }
            }
        }
    }
```

```
[[[20, 117, 162, 0],
  [6, 17, 18, 0]],
 [[1, 11, 18, 0],
  [1, 5, 6, 0]],
 [[0, 1, 2, 0],
  [0, 1, 2, 0]]]
```

```
        return dp[0][1][0];
    }

    public static void main(String[] args) {
        System.out.println(countDigitOne( n: 13)); // 6
    }

}
```

40. Different Ways to Add Parentheses

```
public class DifferentWaysToAddParentheses {
    3 usages
    public static List<Integer> fun(String str) {
        List<Integer> ans = new ArrayList<>();
        for (int i = 1; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch == '*' || ch == '-' || ch == '+') {
                List<Integer> left = fun(str.substring(0, i));
                List<Integer> right = fun(str.substring( beginIndex: i + 1));
                for (int j = 0; j < left.size(); j++) {
                    for (int k = 0; k < right.size(); k++) {
                        if (ch == '+') ans.add(left.get(j) + right.get(k));
                        else if (ch == '-') ans.add(left.get(j) - right.get(k));
                        else if (ch == '*') ans.add(left.get(j) * right.get(k));
                    }
                }
            }
        }
        if (ans.size() == 0) ans.add(Integer.parseInt(str));
        return ans;
    }

    1 usage
    public static List<Integer> diffWaysToCompute(String expression) { return fun(expression); }

    public static void main(String[] args) { System.out.println(diffWaysToCompute( expression: "2-1-1")); } // [2, 0]

}
```

41. Paint House

```java
public class PaintHouse {

    1 usage
    public static int minCost(int[][] costs) {
        int len = costs.length;
        if (costs != null && len == 0)
            return 0;

        int[][] dp = costs;
        // dp[i][j], for house-i, three colors j (0 or 1 or 2)

        // no need dp[m+1][n+1], just i=1 is good enough
        for (int i = 1; i < len; i++) {
            dp[i][0] = costs[i][0] + Math.min(costs[i - 1][1], costs[i - 1][2]);
            dp[i][1] = costs[i][1] + Math.min(costs[i - 1][0], costs[i - 1][2]);
            dp[i][2] = costs[i][2] + Math.min(costs[i - 1][0], costs[i - 1][1]);
        }
        return Math.min(dp[len - 1][0], Math.min(dp[len - 1][1], dp[len - 1][2]));
    }

    public static void main(String[] args) {
        System.out.println(minCost(new int[][]{
                {17, 2, 17},
                {16, 16, 5},
                {14, 3, 19}
        }));
//        10
//        Paint house 0 into blue, paint house 1 into green, paint house 2 into blue.
//             Minimum cost: 2 + 5 + 3 = 10.
    }

}
```

```
[[17, 2, 17],
 [18, 33, 7],
 [21, 10, 37]]
10
```

42. Ugly Number II

```java
public class UglyNumber2 {

    1 usage
    public static int nthUglyNumber(int n) {
        int[] dp = new int[n];
        dp[0] = 1;
        int p2 = 0, p3 = 0, p5 = 0;
        for (int i = 1; i < n; i++) {
            dp[i] = Math.min(Math.min(dp[p2] * 2, dp[p3] * 3), dp[p5] * 5);
            p2 = (dp[i] == dp[p2] * 2) ? p2 + 1 : p2;
            p3 = (dp[i] == dp[p3] * 3) ? p3 + 1 : p3;
            p5 = (dp[i] == dp[p5] * 5) ? p5 + 1 : p5;
        }
        return dp[n - 1];
    }

    public static void main(String[] args) {
        System.out.println(nthUglyNumber( n: 10)); // 12
    }

}
```

[1, 2, 3, 4, 5, 6, 8, 9, 10, 12]

43. Paint House 2

```java
public class PaintHouse2 {

    1 usage
    public static int minCost(int[][] costs) {
        if (costs != null && costs.length == 0) {
            return 0;
        }
        int[][] dp = costs;
        int min1 = -1, min2 = -1;
        for (int i = 0; i < dp.length; i++) {
            int last1 = min1;
            int last2 = min2;
            min1 = -1;
            min2 = -1;
            for (int j = 0; j < dp[i].length; j++) {
                if (j != last1) {
                    dp[i][j] += last1 < 0 ? 0 : dp[i - 1][last1];
                } else {
                    dp[i][j] += last2 < 0 ? 0 : dp[i - 1][last2];
                }

                if (min1 < 0 || dp[i][j] < dp[i][min1]) {
                    min2 = min1;
                    min1 = j;
                } else if (min2 < 0 || dp[i][j] < dp[i][min2]) {
                    min2 = j;
                }
            }
        }
        return dp[dp.length - 1][min1];
    }

    public static void main(String[] args) {
        System.out.println(minCost(new int[][]{
                {1, 5, 3},
                {2, 9, 4}
        })); //  5
//  Paint house 0 into color 0, paint house 1 into color 2. Minimum cost: 1 + 4 = 5;
//  Or paint house 0 into color 2, paint house 1 into color 0. Minimum cost: 3 + 2 = 5.
    }

}
```

```
[[1, 5, 3],
 [5, 10, 5]]
5
```

44. Paint Fence

```java
public class PaintFence {

    1 usage
    public static int numWays(int n, int k) {
        if (n == 0 || k == 0)
            return 0;
        // `dp[i][0]` same color
        // `dp[i][1]` different color
        int[][] dp = new int[n][2];
        dp[0][0] = 0;
        dp[0][1] = k;
        for (int i = 1; i < n; i++) {
            dp[i][0] = dp[i - 1][1]; // or else 3-posts the same color
            dp[i][1] = (dp[i - 1][0] + dp[i - 1][1]) * (k - 1);
        }
        return dp[n - 1][0] + dp[n - 1][1];
    }

    public static void main(String[] args) {
        System.out.println(numWays( n: 3,  k: 2)); // 16
    }

}
```

```
[[0, 2],
 [2, 2],
 [2, 4]]
```

45. Perfect Squares

```java
public class PerfectSquares {

    1 usage
    static int numSquares(int n) {
        int[] dp = new int[n + 1];
        Arrays.fill(dp, n);
        dp[0] = 0;
        dp[1] = 1;
        for (int i = 2; i <= n; ++i)
            for (int j = 1; j * j <= i; ++j)
                dp[i] = Math.min(dp[i], dp[i - j * j] + 1);
        return dp[n];
    }

    public static void main(String[] args) {
        System.out.println(numSquares( n: 12)); // 3
    }
}
```

```
[0, 1, 2, 3, 1, 2, 3, 4, 2, 1, 2, 3, 3]
```

46. Flip Game 2

```java
public class FlipGame2 {

    4 usages
    private static Map<String, Boolean> memo = new HashMap<>();

    2 usages
    public static boolean canWin(String currentState) {
        if (memo.containsKey(currentState))
            memo.get(currentState);
        // If any of currentState[i:i + 2] == "++" and your friend can't win after
        // Changing currentState[i:i + 2] to "--" (or "-"), then you can win
        for (int i = 0; i + 1 < currentState.length(); ++i)
            if (currentState.charAt(i) == '+' && currentState.charAt(i + 1) == '+' &&
                    !canWin( currentState: currentState.substring(0, i) + "-" + currentState.substring( beginIndex: i + 2))) {
                memo.put(currentState, true);
                return true;
            }
        memo.put(currentState, false);
        return false;
    }


    public static void main(String[] args) {
        System.out.println(canWin( currentState: "++++")); // true
    }

}
```

{--=false, ++++=true, +-+=false, -++=true}

47. Longest Increasing Subsequence

```java
public class LongestIncreasingSubsequence {

    1 usage
    static int lengthOfLIS(int[] nums) {
        int n = nums.length;
        int[] dp = new int[n + 1];
        for (int i = 1; i < n; i++) {
            for (int j = 0; j <= i; j++) {
                if (nums[i] > nums[j] && dp[i] < dp[j] + 1)
                    dp[i] = dp[j] + 1;
            }
        }
        return Arrays.stream(dp).max().getAsInt();
    }

    public static void main(String[] args) {
        System.out.println(lengthOfLIS(new int[]{10, 9, 2, 5, 3, 7, 101, 18})); // 3
    }

}
```

[10, 9, 2, 5, 3, 7, 101, 18]

[0, 0, 0, 1, 1, 2, 3, 3, 0]

48. Best Time to Buy and Sell Stock with Cooldown

```java
public class BestTimeToBuyAndSellStockWithCoolDown {

    1 usage
    public static int maxProfit(int[] prices) {
        int n = prices.length;
        int[][] dp = new int[n + 2][2];
        for (int ind = n - 1; ind >= 0; ind--) {
            dp[ind][1] = Math.max(-prices[ind] + dp[ind + 1][0], dp[ind + 1][1]);
            dp[ind][0] = Math.max(prices[ind] + dp[ind + 2][1], dp[ind + 1][0]);
        }
        return dp[0][1];
    }

    public static void main(String[] args) {
        System.out.println(maxProfit(new int[]{1, 2, 3, 0, 2})); // 3
    }

}
```

```
[[4, 3],
 [4, 2],
 [3, 2],
 [2, 2],
 [2, 0],
 [0, 0],
 [0, 0]]
```

49. Burst Balloons

```
1 usage
static int maxCoins(int[] nums) {
    int n = nums.length;
    List<Integer> a = new ArrayList<>(Arrays.stream(nums).boxed().toList());
    a.add( index: 0,  element: 1);
    a.add(1);                                                    [[0, 0, 0, 0, 0, 0],
    int[][] dp = new int[n + 2][n + 2];                           [0, 3, 30, 159, 167, 0],
    for (int d = 0; d < n; ++d)                                   [0, 0, 15, 135, 159, 0],
        for (int i = 1; i < n - d + 1; ++i) {                     [0, 0, 0, 40, 48, 0],
            int j = i + d;                                        [0, 0, 0, 0, 40, 0],
            for (int k = i; k <= j; ++k)                          [0, 0, 0, 0, 0, 0]]
                dp[i][j] = Math.max(dp[i][j],
                        dp[i][k - 1] + dp[k + 1][j] + a.get(i - 1) * a.get(k) * a.get(j + 1));
        }
    return dp[1][n];
}

public static void main(String[] args) {
    System.out.println(maxCoins(new int[]{3, 1, 5, 8})); // 167
}
```

50. Super Ugly Number

```java
public class SuperUglyNumber {

    1 usage
    static int nthSuperUglyNumber(int n, int[] primes) {
        int k = primes.length;
        int[] indices = new int[k];
        int[] uglyNums = new int[n];
        uglyNums[0] = 1;
        for (int i = 1; i < n; ++i) {
            int[] nexts = new int[k];
            for (int j = 0; j < k; ++j)
                nexts[j] = uglyNums[indices[j]] * primes[j];
            int next = Arrays.stream(nexts).min().getAsInt();
            for (int j = 0; j < k; ++j)
                if (next == nexts[j])
                    ++indices[j];
            uglyNums[i] = next;
        }
        return uglyNums[n - 1];
    }

    public static void main(String[] args) {
        System.out.println(nthSuperUglyNumber( n: 12, new int[]{2, 7, 13, 19})); // 32
    }
}
```

Ugly Nums
[1, 2, 4, 7, 8, 13, 14, 16, 19, 26, 28, 32]

Indices
[8, 3, 2, 1]