1. 题目
2. 题干

给两个整数数组 A 和 B ，返回两个数组中公共的、长度最长的子数组的长度。

1. 示例

**输入：**

A: [1,2,3,2,1]

B: [3,2,1,4,7]

**输出：**3

**解释：**

长度最长的公共子数组是 [3, 2, 1] 。

1. 题解
2. 思路

比较两个整数数组，找最大的公共子数组。一开始想着用双指针的方法来比较，但是没办法不断更新比较的结果。之后想到用暴力解法，将A数组的每个数，与B数组的每个数进行比较，但是这样时间复杂度过高。一直都没意识到这种不断递推的题目可以用动态规划来做，之前做DP的题目都是在一个数组对象上操作，这种比较两个数组的题目就可以用一个二维DP数组来解决，dp[i][j]就代表着A[i]开头的子数组与B[j]开头的子数组之间的最长公共子数组长度。从两个数组的末尾开始，不断向前递推，如果两个数组的两个节点的数字相同，说明以dp[i][j]至少为1，其他的就需要看dp[i-1][j-1]的情况了。即dp[i][j] = dp[i-1][j-1] + 1,否则dp[i][j] = 0。

1. 代码实现

Java：

class Solution {

    public int findLength(int[] A, int[] B) {

        // dp思路

        int an = A.length;

        int bn = B.length;

        int[][] dp = new int[an+1][bn+1];

        int ans = 0;

        for(int i = an -1; i >= 0; --i){

            for(int j = bn -1; j >= 0; --j){

                dp[i][j] = A[i] == B[j] ? dp[i+1][j+1] + 1 : 0;

                ans =Math.max(ans, dp[i][j]);

            }

        }

        return ans;

    }

}