# 题目

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。

**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] 。

**提示：**

1 <= len(A), len(B) <= 1000

0 <= A[i], B[i] < 100

注：对比最长回文子串（都可以采用动态规划的算法）

# 分析

## 方法一：动态规划

**思路：**

**代码：**

class Solution {

public:

int findLength(vector<int>& A, vector<int>& B) {

int n = A.size(), m = B.size();

vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));

int ans = 0;

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

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

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

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

}

}

return ans;

}

};

**复杂度分析：**

时间复杂度：O(N×M)。

空间复杂度：O(N×M)。

N表示数组A的长度，M表示数组B的长度。

空间复杂度还可以再优化，利用滚动数组可以优化到O(min(N,M))。

**或（推荐）：**

class Solution {

public:

int findLength(vector<int>& A, vector<int>& B) {

int n = A.size(), m = B.size();

vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));

int ret = 0;

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

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

if(A[i] == B[j])

dp[i][j] = dp[i + 1][j + 1] + 1;

else

dp[i][j] = 0;

ret = max(ret, dp[i][j]);

}

}

return ret;

}

};

## 方法二：滑动窗口

## 方法三：二分查找+哈希表