1. Multiply Strings

Given two non-negative integers num1 and num2 represented as strings, return the product of num1 and num2, also represented as a string.

**Example 1:**

Input: num1 = "2", num2 = "3"  
Output: "6"

**Example 2:**

Input: num1 = "123", num2 = "456"  
Output: "56088"

**Note:**

1. The length of both num1 and num2 is < 110.
2. Both num1 and num2 contain only digits 0-9.
3. Both num1 and num2 do not contain any leading zero, except the number 0 itself.
4. You **must not use any built-in BigInteger library** or **convert the inputs to integer** directly.

**解**

思路和列竖式计算相同，以123 \* 456为例

解法1 分解为 123 \* （400 + 50 + 6），将每次计算的结果累加。但是运行很慢，占据很大内存

class Solution {  
public:  
 string multiply(string num1, string num2) {  
 int len1 = num1.size(), len2 = num2.size();  
 string ans = "";  
 // num1 x num2  
 for(int j = len2 - 1; j >= 0; --j){  
 int sum = 0, add2 = num2[j] - '0';  
 string tmp = "";  
 for(int i = len1 - 1; i >= 0; --i){  
 sum += (num1[i] - '0') \* add2;  
 tmp = to\_string(sum % 10) + tmp;  
 sum /= 10;  
 }  
 while(sum){  
 tmp = to\_string(sum % 10) + tmp;  
 sum /= 10;  
 }  
 ans = add(ans, tmp + string(len2 - j - 1, '0'));  
 }  
 while(ans[0] == '0' && ans.size() > 1)ans.erase(ans.begin());  
 return ans;  
 }  
 string add(string num1, string num2){  
 int i = num1.size() - 1, j = num2.size() - 1;  
 int sum = 0;  
 string res = "";  
 while(i >= 0 && j >= 0){  
 sum += (num1[i] - '0') + (num2[j] - '0');  
 res = to\_string(sum % 10) + res;  
 sum /= 10;  
 i--, j--;  
 }  
 while(i >= 0){  
 sum += (num1[i] - '0');  
 res = to\_string(sum % 10) + res;  
 sum /= 10;  
 i--;  
 }  
 while(j >= 0){  
 sum += (num2[j] - '0');  
 res = to\_string(sum % 10) + res;  
 sum /= 10;  
 j--;  
 }  
 if(sum != 0)res = to\_string(sum) + res;  
 return res;  
 }  
};

解法2 分解为(100 + 20 + 3) \* (400 + 50 + 6)，每次计算之后直接累加结果

class Solution {  
public:  
 string multiply(string num1, string num2) {  
 string res = "";  
 int m = num1.size(), n = num2.size();  
 vector<int> vals(m + n);  
 for (int i = m - 1; i >= 0; --i) {  
 for (int j = n - 1; j >= 0; --j) {  
 int mul = (num1[i] - '0') \* (num2[j] - '0');  
 int p1 = i + j, p2 = i + j + 1, sum = mul + vals[p2];  
 vals[p1] += sum / 10;  
 vals[p2] = sum % 10;  
 }  
 }  
 for (int val : vals) {  
 if (!res.empty() || val != 0) res.push\_back(val + '0');  
 }  
 return res.empty() ? "0" : res;  
 }  
};