Product of Array Except Self

Given an array of n integers where $n > 1$, `nums`, return an array `output` such that `output[i]` is equal to the product of all the elements of `nums` except `nums[i]`.

Solve it **without division** and in $O(n)$.

For example, given `[1,2,3,4]`, return `[24,12,8,6]`.

Follow up:
Could you solve it with constant space complexity? (Note: The output array **does not** count as extra space for the purpose of space complexity analysis.)

Solution 1

```
public class Solution {
public int[] productExceptSelf(int[] nums) {
    int n = nums.length;
    int[] res = new int[n];
    res[0] = 1;
    for (int i = 1; i < n; i++) {
        res[i] = res[i - 1] * nums[i - 1];
    }
    int right = 1;
    for (int i = n - 1; i >= 0; i--) {
        res[i] *= right;
        right *= nums[i];
    }
    return res;
}
```

}

written by lycjava3 original link here

Solution 2

Use `tmp` to store temporary multiply result by two directions. Then fill it into `result`. Bingo!

```
public int[] productExceptSelf(int[] nums) {
    int[] result = new int[nums.length];
    for (int i = 0, tmp = 1; i < nums.length; i++) {
        result[i] = tmp;
        tmp *= nums[i];
    }
    for (int i = nums.length - 1, tmp = 1; i >= 0; i--) {
        result[i] *= tmp;
        tmp *= nums[i];
    }
    return result;
}
```

written by xcv58 original link here

Solution 3

First, consider O(n) time and O(n) space solution.

```
class Solution {
public:
    vector<int> productExceptSelf(vector<int>& nums) {
        int n=nums.size();
        vector<int> fromBegin(n);
        fromBegin[0]=1;
        vector<int> fromLast(n);
        fromLast[0]=1;

        for(int i=1;i<n;i++){
            fromBegin[i]=fromBegin[i-1]*nums[i-1];
            fromLast[i]=fromLast[i-1]*nums[n-i];
        }

        vector<int> res(n);
        for(int i=0;i<n;i++){
            res[i]=fromBegin[i]*fromLast[n-1-i];
        }
        return res;
    }
};
```

We just need to change the two vectors to two integers and note that we should do multiplying operations for two related elements of the results vector in each loop.

```
class Solution {
public:
    vector<int> productExceptSelf(vector<int>& nums) {
        int n=nums.size();
        int fromBegin=1;
        int fromLast=1;
        vector<int> res(n,1);

        for(int i=0;i<n;i++){
            res[i]*=fromBegin;
            fromBegin*=nums[i];
            res[n-1-i]*=fromLast;
            fromLast*=nums[n-1-i];
        }
        return res;
    }
};
```

written by zhaoqiang original link here