553. Optimal Division

Add to List

Description HintsSubmissionsSolutions

- Total Accepted: **2583**
- Total Submissions: **4804**
- Difficulty: **Medium**
- Contributors:fallcreek

Given a list of **positive integers**, the adjacent integers will perform the float division. For example, [2,3,4] -> 2 / 3 / 4.

However, you can add any number of parenthesis at any position to change the priority of operations. You should find out how to add parenthesis to get the **maximum** result, and return the corresponding expression in string format. **Your expression should NOT contain redundant parenthesis.**

Example:

Input: [1000,100,10,2] **Output:** "1000/(100/10/2)" **Explanation:**

1000/(100/10/2) = 1000/((100/10)/2) = 200

However, the bold parenthesis in "1000/((100/10)/2)" are redundant,
since they don't influence the operation priority. So you should return "1000/(100/10/2)".

Other cases:

1000/(100/10)/2 = 50

1000/(100/(10/2)) = 50

1000/100/10/2 = 0.5

```
1000/100/(10/2) = 2
```

Note:

1. The length of the input array is [1, 10].
2. Elements in the given array will be in range [2, 1000].
3. There is only one optimal division for each test case.

Subscribe to see which companies asked this question.

```
#include<iostream>
#include<sstream>
#include<stdio.h>
#include<vector>
#include<unordered_set>
#include<unordered_map>
#include<limits.h>
#include<set>
#include<random>
#include<ctime>
using namespace std;
template<typename T>
string to_string(T t)
{
    stringstream ss;
    ss<<t;
    string res = "";
    ss>>res;
    return res;
}

string optimalDivision(vector<int>& nums) {
    stringstream ss;
    string result = "";
    if (nums.size()==1)
    {
        ss<<nums[0];
        ss>>result;
        return result;
    }
    if(nums.size()==2)
    {
```

```
        double ans_temp = (double)nums[0]/nums[1];
        ss<<ans_temp;
        ss>>result;
        return result;
    }
    int sz = nums.size();
    vector<vector<double>>                    d_grid                    =
vector<vector<double>>(sz,vector<double>(sz,0));
    vector<vector<string>>                    s_grid                    =
vector<vector<string>>(sz,vector<string>(sz,""));
    for(int row=sz-2;row>=0;row--)
    {
        for(int col=row+1;col<sz;col++)
        {
            if(col==row+1)
            {
                d_grid[row][col] = (double)nums[row]/nums[col];
                string tp1=""; string tp2="";
                ss<<nums[row];ss>>tp1;ss.clear();
                ss<<nums[col];ss>>tp2;ss.clear();
                s_grid[row][col] = tp1 + "/" + tp2;
                continue;
            }
            if(row==0 && col==sz-1)
            {
                //max

    if((double)nums[row]/d_grid[row+1][col]>=(double)d_grid[row][col-1]/
nums[col])
                {
                    d_grid[row][col]                                    =
(double)nums[row]/d_grid[row+1][col];
                    string tp1=""; ss << nums[row]; ss >> tp1;ss.clear();
                    s_grid[row][col] = tp1 + "/" + "(" + s_grid[row+1][col]
+ ")";
                }else{
                    d_grid[row][col]                                    =
(double)d_grid[row][col-1]/nums[col];
                    string tp1=""; ss << nums[row]; ss >> tp1;ss.clear();
                    s_grid[row][col] = tp1 + "/" +  s_grid[row+1][col];
                }
            }else
            {
                //min
```

```
    if((double)nums[row]/d_grid[row+1][col]<(double)d_grid[row][col-1]/n
ums[col])
                {
                    d_grid[row][col]                                    =
(double)nums[row]/d_grid[row+1][col];
                    string tp1=""; ss << nums[row]; ss >> tp1;ss.clear();
                    s_grid[row][col] = tp1 + "/" + "(" + s_grid[row+1][col]
+ ")";
                }else{
                    d_grid[row][col]                                    =
(double)d_grid[row][col-1]/nums[col];
                    string tp1=""; ss << nums[row]; ss >> tp1;ss.clear();
                    s_grid[row][col] = tp1 + "/" +  s_grid[row+1][col];
                }
            }
        }
    }
    double final_ans = d_grid[0][sz-1];
    //cout << final_ans<<endl;
    //cout << s_grid[0][sz-1] << endl;
    result = s_grid[0][sz-1];
    return result;
}

int main(int argc,char *argv[])
{
    vector<int> nums = {1000,100,10,2};
    string ans = optimalDivision(nums);
    cout << ans;
    string s = to_string<int>(11);
    cout << s;
    return 0;
}
```