Optimal Division

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.

Solution 1

```
public class Solution {
    class Result {
        String str;
        double val;
    }

    public String optimalDivision(int[] nums) {
        int len = nums.length;
        return getMax(nums, 0, len - 1).str;
    }

    private Result getMax(int[] nums, int start, int end) {
        Result r = new Result();
        r.val = -1.0;

        if (start == end) {
            r.str = nums[start] + "";
            r.val = (double)nums[start];
        }
        else if (start + 1 == end) {
            r.str = nums[start] + "/" + nums[end];
            r.val = (double)nums[start] / (double)nums[end];
        }
        else {
            for (int i = start; i < end; i++) {
                Result r1 = getMax(nums, start, i);
                Result r2 = getMin(nums, i + 1, end);
                if (r1.val / r2.val > r.val) {
                    r.str = r1.str + "/" + (end - i >= 2 ? "(" + r2.str + ")" : r2.
str);
                    r.val = r1.val / r2.val;
                }
            }
        }

        //System.out.println("getMax " + start + " " + end + "->" + r.str + ":" + r
.val);
        return r;
    }

    private Result getMin(int[] nums, int start, int end) {
        Result r = new Result();
        r.val = Double.MAX_VALUE;

        if (start == end) {
            r.str = nums[start] + "";
            r.val = (double)nums[start];
        }
        else if (start + 1 == end) {
            r.str = nums[start] + "/" + nums[end];
            r.val = (double)nums[start] / (double)nums[end];
        }
        else {
            for (int i = start; i < end; i++) {
                Result r1 = getMin(nums, start, i);
                Result r1 = getMin(nums, start, i);
                Result r2 = getMax(nums, i + 1, end);
                if (r1.val / r2.val < r.val) {
                    r.str = r1.str + "/" + (end - i >= 2 ? "(" + r2.str + ")" : r2.
str);
                    r.val = r1.val / r2.val;
                }
            }
        }

        //System.out.println("getMin " + start + " " + end + "->" + r.str + ":" + r
.val);
        return r;
    }
}
```

written by shawngao original link here

Solution 2

```java
public class Solution {
    public String optimalDivision(int[] nums) {
        StringBuilder builder = new StringBuilder();
        builder.append(nums[0]);
        for (int i = 1; i < nums.length; i++) {
            if (i == 1 && nums.length > 2) {
                builder.append("/(").append(nums[i]);
            } else {
                builder.append("/").append(nums[i]);
            }
        }

        return nums.length > 2 ? builder.append(")").toString() : builder.toString(
);
    }
}
```

written by aaddccadc original link here

Solution 3

Regardless of parentheses, every element is either in the numerator or denominator of the final fraction. The expression A[0] / **(** A[1] / A[2] / ... / A[N-1]**)** has every element in the numerator except A[1], and it is impossible for A[1] to be in the numerator, so it is the largest. We must also be careful with corner cases.

```
def optimalDivision(self, A):
    A = map(str, A)
    if len(A) <= 2: return '/'.join(A)
    return '{}/({})'.format(A[0], '/'.join(A[1:]))
```

written by awice original link here