8，o(N)时间复杂度解决面试中遇到的问题中

2018年4月27日 19:50

- 再谈O(n)
- 一些例题
  - 例1 下一个排列
  - 例2 均分01
  - 例3 X的个数
  - 例4 PAT的个数
  - 例5 最小平均值子数组
  - 例6 环形最大子数组和
  - 例7 允许交换一次的最大子数组和
- 总结
- 再谈O(n)
- 组合数学
  - 下一个排列（上一个排列）
  - 巧妙地证明
  - 计数 != 枚举
- 动态规划

- 一些例题
  - 例1 下一个排列

```
/*
问题：1 (C++ STL) Next Permutation 找到字典序里的下一个排列。
12345的下一个是12354,而54321的下一个认为是12345。 (Leetcode 31)
https://leetcode.com/problems/next-permutation/description/

分析:如果目前排列为(A)a[x](B),那么下一个排列为 (A)a[y](B'),中间a为数组
其中A是一个序列，它尽可能长，中间明显要a[x]大于a[y]的，后面b‘几乎是B里面所有数排好序的结果

    那么如何确定x呢？
    一个位置，只要右边有数比它大，它就是候选的x。
```

同时，a[x]一定是最后一个这样的数
1,a[x]右边的数，是按照降序or不升序排列的

算法：二找，一交换，一翻转
1,逆序从右到左，找到第一个不严格升序的首位，定义为x。x处以后从左往右都是降序的。x+1处最大
2,找到y>x,a[y]>a[x],且a[y]最小
因为a[x]后面的数都是降序，因此，从后往前找到第一个大于a[x]的位置就是y了
3,交换a[x],a[y]处元素。
4,交换完,新的a[x+1,...,n-1]仍然是降序，对这部分进行逆序，变成严格升序即可。

拓展：找上一个排序，
从右到左，找第一个非严格降序的首位，
然后从右到左，找到第一个a[y]<a[x]的位置
交换
对x+1后面的部分逆序即可
*/

```
class Solution {
public:
    void nextPermutation(vector<int>& nums) {
        int n = nums.size();
        int x;
        for(x = n-2;(x>=0)&&nums[x]>=nums[x+1];--x);//从右往左查找x，最后得到的x是从右往左，第一个不是严格升序的字母，后面x+1到最后都是严格递减的
        if(x<0)//特殊情况，说明整个串从左到右都是递减的，比如4321，那么整体逆序即可
        {
            reverse(nums.begin(),nums.end());
            return;
        }
        int y;
        for(y = n-1;nums[y]<=nums[x];--y);//从右往左，找到第一个大于a[x]的位置
        swap(nums[x],nums[y]);//交换
        //交换完，将x后面的部分整体逆序变成升序即可
        reverse(nums.begin()+x+1,nums.end());
        return;

    }
};
```

来自 <http://tool.oschina.net/highlight>

-
  - ## 例2 均分01

```
/*
问题：给定一个01串，恰好包含2n个0和2n个1，你可以把它切成若干段，再把它们任意拼接，要求拼接出两部分，每部分恰好包含n个0，n个1,
如何使得切的段数最少？

比如0101,中间切，切为01,01,分别作为两部分
0011,切成0 01 1,把0 1组合形成一部分，剩下的是一部分

分析：下标从0开始，f(x),x=0,1,...,2n.表示原串在[x,x+2n-1]这个窗口（大小为2n)里，0和1个数差.(注意，串总长为4n)
首先，f0+f2n = 0.为什么呢，它前后分别表示串前半段和后半段，整个串01之差自然为0
```

1,如果f0 = f2n = 0,那么自然的，从中间切一刀即可。答案为2
2,如果一个小于0一个大于0,那么首先它是奇数还是偶数呢?首先一定是偶数，不信可以试一下
然后窗口滑动是如何滑动的呢?fx变成fx+1时，有三种可能，
去掉1个0,补一个1,-2
or去掉一个1,补一个0,+2
or去0补0,去1补1,不变

因此，窗口在滑动时，偶数由负数变正数or正变负过程中，一定能存在f（y）=0的点，那么把[y,2n+y-1]作为一段，它包含n个0n个1,剩下的[0,y-1]和[2n+y,4n-1]合并起来作为一段，
因此，切三段即可

总之，三段or两段即可
*/

```
class Solution
{
    int solution(string s)
    {
        int n = s.size();
        int d = 0;
        for(int i = 0;i<n/2;i++)//窗口大小为总长的一半,这里相当于上面判断f0和fn是否相等。
        {
            if(s[i] == '0') ++d;
            else if(s[i] == '1') --d;
        }
        if(d == 0) return 2;
        else return 3;

    }
}
```

来自 <http://tool.oschina.net/highlight>

-

- ## 例3 X的个数

```
/*
问题：给定一个长度为n的整数数组a，下标从0开始，再给定一个元素X，求一个位置m, 满足0<=m<=n,
且a[0..m - 1]中X的个数（如果m = 0表示空数组）和a[m..n - 1]中非X的个数(如果m == n, 表示空数组）相等。

分析：假定a中有x个X，给定m，假设a[0,m-1]中有y个X，那么a[m,n-1]中非X的个数应为(n-m)-(x-y)（前面为元素个数，后面为x的个数）
,且根据题意，其还等于y
那么有n-m-x=0.解得m = n-x。其中x为X个数。因此m存在，且唯一

因此，只需循环一边，统计X的个数，然后返回n-num即可

思考：10个硬币，有4个是正面的，在不开灯的情况下，将其分成两组，希望正面个数相等
```

解法：分成前6个一组，后4个一组，把后4个翻面。这也是个很神奇的思路
*/

来自 <http://tool.oschina.net/highlight>

例4 PAT的个数

/*
问题：（PAT：Programming Ability Test）给定一个只包含P,A,T的串，求一共出现多少个“PAT” 子序列？

分析：这里只需要计数不需要统计，因此N时间复杂度就可以做。怎么做呢？
p，pa，pat表示之前出现过得“P”，“PA",”PAT"的个数，即
s[i] == 'P',++p;
s[i] == 'A',pa+=p;//这时因为之前出现过得所有”P",都可以与之形成"PA",因此计数加上之前p的个数
s[i] == 'T',pat+=pa;

思考题：leetcode115
https://leetcode.com/problems/distinct-subsequences/description/
分析：设dp[i+1][j+1]表示t[0,i]在s[0,j]中出现的次数。易得最后返回的应是dp[t.size()][s.size()]
首先第一行，代表任意串包含空串的次数，均设为1.即dp[0][j] = 1;//这里如果t存在特殊情况，即空串，也能正确返回。
然后第一列，代表空串包含任意串的次数，均设为0,即dp[i][0] = 0，注意dp[0][0] = 1.

首先，如果t[i]!=s[j]，那么不使用s[j],dp[i+1][j+1] =dp[i+1][j];
否则，使用s[j],dp[i+1][j+1] = dp[i+1][j]+dp[i][j]//注意到了没，后面这里相当于计算pat时，加上pa的统计
*/

```
class Solution {
public:
    int numDistinct(string s, string t) {
        vector<vector<int> > dp(t.size()+1,vector<int>(s.size()+1,0));
        for(int j = 0;j<=s.size();j++)//初始化第一行
            dp[0][j] = 1;


        for(int i = 0;i<t.size();i++)
        {
            for(int j = 0;j<s.size();j++)
            {
                if(t[i] == s[j])
                    dp[i+1][j+1] = dp[i+1][j]+dp[i][j];
                else
                    dp[i+1][j+1] = dp[i+1][j];
            }
        }
```

```
        return dp[t.size()][s.size()];
    }


};
```

来自 <http://tool.oschina.net/highlight>

例5 最小平均值子数组

```
/*
问题：最小平均值子数组
 给定一个数组，求一个至少包含两个元素的子数组，满足平均值最小。输出子数组的起点，多个的时候输出最小的。



分析：我们先从结果倒推，如果最优解的长度为偶数，那么我们可以将最优解拆成长度为2的若干段，
    否则，如果是奇数，那么将其拆成若干年长度为2的和一个长度为3的
    最优解中，每一段的平均值都必须相等，为什么呢？
    如果有一段平均值比最优解小，那么肯定有一段比最优解大，那么为什么还要哪些比最优值大的部分？


    因此，我们要看最优解，只要看长度为2or3的段就可以了。
*/
#include<bits/stdc++.h>
using namespace std;
class Solution
{
public:
    int solution(vector<int> a)
    {
        int n= a.size();
        if(n<2) return -1;
        int total2 = a[1]+a[2];//2滑动窗口初始化。为了之后滑动的时候统一从3开始，这里将total2初始化为了1,2,并根据012处大小初始化了best和start
        int best2;
        int start2;
        if(a[0]<=a[2])
        {
            start2 = 0;
            best2 = a[0]+a[1];
        }
        else
        {
            start2 = 1;
            best2 = a[1]+a[2];
        }
        int total3 = a[0]+a[1]+a[2];//3大小滑动窗口初始化
        int best3 = total3;
```

```
        int start3 = 0;


        for(int i = 3;i<n;i++)
        {
            total2 += a[i] - a[i-2];//相当于窗口后移一位
            if(total2<best2)//这里发现了没，等于的时候并未修改start值
            {
                best2 = total2;
                start2 = i-1;//窗口开始位置
            }
            total3 += a[i]-a[i-3];
            if(total2<best3)
            {
                best3 = total3;
                start3 = i-2;
            }
        }
        //不要忘了，这里我们得到的best2,best3为累加值，而非平均。
        //因此我们需要对best2/2,best3/3,但这里可以用乘法代替除法，即比较 best2*3 和 best3*2
        int cmp = best2*3-best3*2;
        if(cmp == 0)
            return min(start2,start3);
        return cmp<0?start2:start3;
    }
};
int main()
{
    int a[7] = {1,3,-8,-9,-10,11,2};
    vector<int> arr(a,a+7);
    Solution s;
    cout<<s.solution(arr)<<endl;
    return 0;
}
```

来自 <http://tool.oschina.net/highlight>

○ 例6 环形最大子数组和

```
/*
```

问题： 环形最大子数组和

（itint5 9)给定一个数组，是环形的，最后一个元素和第一个元素相接，求最大子数组和。

如：普通的：1 2 -4 5 6 -9 返回5 +6
环形的，1 2 -4 -5 -6 9返回 9 1 2

因此，环形的比起普通的，无非是先按普通的求一次，再求出普通的的最小子数组和，再用总和减去最小的，
两次比较下哪一个大，返回哪个就行。

求最小字数组和时，可以对元素统统取反，再计算一次最大字数组和即可。

如何计算最大子数组和呢？维持一个等长的数组f，计算在这之前（包括该数最大的字数组和，放在该位置）
如果上一个位置上的f值大于0,则累加，否则，置fi = ai即可。

最后再遍历一遍，求得f的最大值，即为最大子数组和。

分析：
*/

来自 <http://tool.oschina.net/highlight>

○ 例7 允许交换一次的最大子数组和

```
/*
问题：允许交换一次的最大子数组和
    （codility）给定一个数组，在允许交换两个数的前提下（只允许交换一次，可以不换），求最大子数组和。

分析：1,定义fi为两部分之和，以a[i]为结尾的最大字数组的和（可以为空），和a[0,i]里面任意一个单独元素的和。
注意，这两部分不能求交集,即fi其实是一段+一个孤立点，这个孤立点不能取在段上。
fi = max(fi-1 + ai,max(a[0,...,i]),这个表达式可以这样理解，fi可以取fi-1+ai,也可以取0+max(a[0,i]),两者取最大

这个fi有什么意义呢，这里面的孤立点，就是ai想要换掉的点

    2,定义gi为以a[i]开头的最大子数组和
    gi = max（gi+1,0）+ai

    3,如果ai和aj交换(j<i)，那么原来包含a[i]的最大字数组和变成g[i]-a[i]+f(i-1).
    即，要换掉的a[j]在f(i-1)里
    如果不交换，答案就是max{g[i]}

    4,我们值考虑j<i的情况，对于j>i，把a反转再做一次就好了

*/
#include<bits/stdc++.h>
using namespace std;
class Solution
{
public:
    int solution(vector<int> &a)
    {
        int res = help(a);
        reverse(a.begin(),a.end());
        res = max(res,help(a));
```

```
        return res;
    }
private:
    int help(vector<int> &a)
    {
        int n = a.size();
        vector<int> f(n),g(n);//分别是以ai结尾和开头的最大子序列和
        f[0] = a[0];
        int now = a[0];
        for(int i = 1;i<n;i++)//得到f，其含义就是a[i]要交换的位置的元素的值。（这个位置是在i左边的，因此后面发现要反转一次
        {
            now = max(now,a[i]);//a[i]之前包括a[i]的最大值
            f[i] = max(a[i]+f[i-1],now);//f取最大连续子序列和，和前面最大值中较大的那个
        }
        g[n-1] = a[n-1];
        int res = a[n-1];//统计g[i]最大值,即不交换的情况下的最大值
        for(int i = n-2;i>=0;i--)
        {
            g[i] = max(g[i+1],0)+a[i];
            res = max(res,g[i]);
        }
        for(int i = 1;i<n;i++)//交换情况下的最大值
        {
            res = max(res,g[i]-a[i]+f[i-1]);//这里为何是fi-1.是因为ai要和它之前的最大值交换
        }
        return res;
    }
};


int main()
{
    int a[4] = {10,-100,10,10};
    vector<int> arr(a,a+4);
    Solution s;
    cout<<s.solution(arr)<<endl;
    return 0;
}
```

来自 <http://tool.oschina.net/highlight>

- 总结
- 多思考，多练习
- 计数！= 枚举
- 没有讲到的问题

- $O(n^3)$优化到$O(n^2)$
- 序列相关的问题
  - 给定一个1-n的排列，每次只能把一个数放到序列开头，至少几次能排好顺序？
  - 给定一个1-n的排列，每次可以把一个数放到序列开头，也可以放到结尾，至少几次能排好序？
- 更多前缀、后缀的利用
- …