CCF考试准备

考试时间：2021-12-05

考试环境：

C/C++：Dev-CPP 5.4.0 (Min GW 4.7.2)

Dev选择**C++11**标准：

https://blog.csdn.net/IT_flying625/article/details/89255376?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216374722101678036654833 4%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163747221016780366548334&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-89255376.first_rank_v2_pc_rank_v29&utm_term=Dev%E5%A6%82%E4%BD%95%E9%80%89%E6%8B%A9%E4%BD%BF%E7%94%A8std11&spm=1018.2226.3001.4187

可以点击菜单栏的“工具”-»“编译选项”进入如下界面，勾选“编译时加入以下指令”，填入"-std=c++11"

头文件：

- 建议打印常用头文件的列表

```
#include<bits/stdc++.h>
using namespace std; //万能头文件

#include <iostream>
#include <cstdio>
#include <fstream>
#include <algorithm>
#include <cmath>
#include <deque>
#include <vector>
#include <queue>
#include <string>
#include <cstring>
#include <map>
#include <stack>
#include <set>
```

2021_9_2试题

相关博客地址：

https://blog.csdn.net/qq_43211230/article/details/121177529?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216374235521678036654041O%2522%252C%2522scm%2522%253A%25222014071 3.130102334..%2522%257D&request_id=163742355216780366540410&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-121177529.first_rank_v2_pc_rank_v29&utm_term=ccf202109&spm=1018.2226.3001.4187

题目描述：

题目描述

$A_1, A_2, \cdots, A_n$ 是一个由 n 个自然数（非负整数）组成的数组。我们称其中 $A_i, \cdots, A_j$ 是一个非零段，当且仅当以下条件同时满足：

- $1 \le i \le j \le n$；
- 对于任意的整数 k，若 $i \le k \le j$，则 $A_k > 0$；
- $i = 1$ 或 $A_{i-1} = 0$；
- $j = n$ 或 $A_{j+1} = 0$。

下面展示了几个简单的例子：

- $A = [3, 1, 2, 0, 0, 2, 0, 4, 5, 0, 2]$ 中的 4 个非零段依次为 $[3, 1, 2]$、$[2]$、$[4, 5]$ 和 $[2]$；
- $A = [2, 3, 1, 4, 5]$ 仅有 1 个非零段；
- $A = [0, 0, 0]$ 则不含非零段（即非零段个数为 0）。

现在我们可以对数组 A 进行如下操作：任选一个正整数 p，然后将 A 中所有小于 p 的数都变为 0。试选取一个合适的 p，使得数组 A 中的非零段个数达到最大。若输入的 A 所含非零段数已达最大值，可取 $p = 1$，即不对 A 做任何修改。

输入格式

从标准输入读入数据。

输入的第一行包含一个正整数 n。

输入的第二行包含 n 个用空格分隔的自然数 $A_1, A_2, \cdots, A_n$。

输出格式

输出到标准输出。

仅输出一个整数，表示对数组 A 进行操作后，其非零段个数能达到的最大值。

题目分析：

- 理解非零段的意义：连续的数字中没有**0**
- 求解最大非零段的数量：没有捷径可求，暴力模拟+优化
- 我的思路：根据**p**的取值从小到大排序，逐一测试，每次测试时都缩短数组的长度

```
//#include<vector>
//#include<iostream>
#include<bits/stdc++.h>
using namespace std;

int getnum(vector<int>& data, int size) {
    int res = 0;
```

```
    int i = 0;
    while (i < size) {
        while (i<size&&data[i] == 0)++i;
        if (i >= size)break;
        while (i<size&&data[i] != 0)++i;
        ++res;
    }
    return res;

}
int main() {
    //暴力+优化
    //处理输入
    int n;
    cin >> n;
    //n可能为0
    if (n == 0)return 0;
    vector<int>data1(n, 0);
    vector<int>data2(n, 0);
    vector<int>s(n, 0);
    for (int i = 0; i < n; i++) {
        cin >> data1[i];
        s[i] = data1[i];
    }
    int n1 = n, n2 = n;

    //p的取值优化
    sort(s.begin(), s.end());
    vector<int>p;
    p.push_back(s[0]);
    for (int i = 1; i < n; i++) {
        if (s[i] != s[i - 1])p.push_back(s[i]);
    }
    int m = p.size();

    int res = getnum(data1,n);
    for (int i = 0; i < m; i++) {
        int pp = p[i];
        int tag = 0;
        if (i % 2 == 0) {
            //除去0，压缩
            //除去前导0
            int j = 0;
            while (j<n1&&data1[j] == 0)++j;
            for (j; j < n1; j++) {
                if (data1[j] < pp)data1[j] = 0;
                if (data1[j] != 0||(j > 0 && data1[j - 1] != 0))
                    data2[tag++] = data1[j];
            }
```

```
            n2 = tag;//更新数组大小
            res = max(res, getnum(data2, n2));

        }
        else {
            //除去0，压缩
            //除去前导0
            int j = 0;
            while (j < n2 && data2[j] == 0)++j;
            for (j; j < n2; j++) {
                if (data2[j] < pp)data2[j] = 0;
                if (data2[j] != 0 || (j > 0 && data2[j - 1] != 0))
                    data1[tag++] = data2[j];
            }
            n1 = tag;//更新数组大小
            res = max(res, getnum(data1, n1));
        }
    }
    cout << res;
    return 0;

}
```

子任务

70% 的测试数据满足 $n \leq 1000$;

全部的测试数据满足 $n \leq 5 \times 10^5$，且数组 A 中的每一个数均不超过 10^4。

只能通过**70%**的用例

模拟+优化（更为精确的寻找要修改的位置并进行修改）

- 使用**set**来获取数组**num**的有序排列（从小到大）;
- 使用**map**来获取数组**num**中元素对索引的映射，这里一个元素可以对应多个索引;
- 遍历**set**集合并通过**map**来对数组**num**进行判断, 并设置两个辅助的索引**begin, end;**

1. 如果**num[end]==0**,更改**num[end]=0**;
2. 判断辅助的索引**begin, end**位置来判断非零段的增加还是减少;
3. 获取最大的非零段大小

```
#include<iostream>
#include<unordered_map>
#include<bits/stdc++.h>

using namespace std;

int main() {
```

```
    int n;
    cin>>n;

    vector<int> Num;
    set<int> Diffset;
    unordered_map<int, vector<int>> Num_index;

    Num.push_back(0);//增加边界

    for(int i=1; i<=n; i++) {
        int num;
        cin>>num;

        Num.push_back(num);
        Diffset.insert(num);
        Num_index[num].push_back(i);//相同数字的索引是以小到大排列
    }

    Num.push_back(0);//增加边界
    int res=0;
    for(int i=1; i<=Num.size(); i++) {
        if(Num[i+1] == 0 && Num[i]!=0) res++;
    }

    int maxres = res;
    for(int num: Diffset) {

        if(num == 0) continue;

        for(int i=0; i<Num_index[num].size(); i++) {

            int begin=Num_index[num][i], end=begin;

            if(Num[begin] == 0) continue; //在之前已经将 begin变成了零

            while(Num[end] == num) {
                Num[end] = 0;
                end++;
            }
            end-=1;
            //判断非零段
            //在 begin的左边只能出现比num大的数字或者0
            if(Num[begin-1] == 0 && Num[end+1] == 0) {
                res = res -1;
            } else {
            if(Num[begin-1] != 0 && Num[end+1] != 0) {
                res = res + 1;
            }
            }
```

```
                //其他情况非零区间不变
                //maxres = max(res, maxres) ;    将这一段放在这里出错，不知道什么原因
            }
            maxres = max(res, maxres) ;
        }
        cout<<maxres;
        return 0;
    }
```

2021_9_4:

相关博客地址：

https://blog.csdn.net/weixin_51554954/article/details/120559907?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216378563731678025527711 2%2522%252C%2522scm%2522%253A%25222014071 3.130102334.pc%255Fall.%252 2%257D&request_id=163785637316780255277112&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-4-120559907.first_rank_v2_pc_rank_v29&utm_term=csp2021-9-4&spm=1018.2226.3001.4187

题目描述：

题目描述

小林在玩一个抽卡游戏，其中有 n 种不同的卡牌，编号为 1 到 n。每一次抽卡，她获得第 i 种卡牌的概率为 p_i。如果这张卡牌之前已经获得过了，就会转化为一枚硬币。可以用 k 枚硬币交换一张没有获得过的卡。

小林会一直抽卡，直至集齐了所有种类的卡牌为止，求她的期望抽卡次数。如果你给出的答案与标准答案的绝对误差不超过 10^{-4}，则视为正确。

提示：聪明的小林会把硬币攒在手里，等到通过兑换就可以获得剩余所有卡牌时，一次性兑换并停止抽卡。

输入格式

从标准输入读入数据。

输入共两行。第一行包含两个用空格分隔的正整数 n, k，第二行给出 $p_1, p_2, \ldots, p_n$，用空格分隔。

输出格式

输出到标准输出。

输出共一行，一个实数，即期望抽卡次数。

样例1输入

```
2 2
0.4 0.6
```

子任务

对于 20% 的数据，保证 $1 \le n, k \le 5$。

对于另外 20% 的数据，保证所有 p_i 是相等的。

对于 100% 的数据，保证 $1 \le n \le 16, 1 \le k \le 5$，所有的 p_i 满足 $p_i \ge \frac{1}{10000}$，且 $\sum_{i=1}^{n} p_i = 1$。

题目分析：

- 很显然：需要回溯，但是会超时
- 回溯+记忆化搜索（动态规划）
- 如何漂亮的写出记忆化搜索？

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<vector>

using namespace std;

typedef long long LL;
typedef pair<int,int> PII;
const int N=2e6+10,mod=998244353;
bool dis[1<<17][100];
double g[17],dp[1<<17][100];//dp[i][j]标识的是从这个状态出发到最终状态抽卡数量的期望值
int n,k;

double dfs(int t,int cnt,int time,int sum,double l)
{
    /*
        t:位向量
        cut:硬币数
        time:抽卡次数
        sum:已经抽到的卡牌数量
        l: 已经抽到的卡牌数量的概率和
    */
    if(dis[t][time])return dp[t][time];//如果已经遍历过了，则直接返回

    dis[t][time]=1;//更改存在性
    if(sum==n||(cnt>=(n-sum)*k))return dp[t][time]=time;//在末梢处返回卡牌总数

    for(int i=1;i<=n;i++)
        if(!((t>>i)&1))
            dp[t][time]+=g[i]*dfs(t|(1<<i),cnt,time+1,sum+1,l+g[i]);

    if(t)dp[t][time]+=l*dfs(t,cnt+1,time+1,sum,l);
```

```
    return dp[t][time];
}

int main()
{
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++)scanf("%lf",g+i); //输出lf,防止因为精度而报错
    printf("%.10lf\n",dfs(0,0,0,0,0));
    return 0;
}
```

需要深刻理解！！！