北京郵電大學

Beijing University of Posts and Telecommunications

程序设计竞赛基础

第九周作业

第 23 次 CCF 计算机

软件能力认证 解题报告

课程: 程序设计竞赛基础

姓名: 卢安来

学号: 2022212720

时间: 2023 年 4 月 23 日

目录

提交结果

3632318	卢安来	卢安来	箱根山岳险天下	04-20 17:49	4.299KB	CPP14	正确	100	734ms	22.90MB
3632317	卢安来	卢安来	收集卡牌	04-20 17:49	559B	CPP14	正确	100	93ms	43.37MB
3632316	卢安来	卢安来	脉冲神经网络	04-20 17:49	2.339KB	CPP14	正确	100	890ms	7.609MB
3632315	卢安来	卢安来	非零段划分	04-20 17:49	827B	CPP14	正确	100	78ms	13.36MB
3632314	卢安来	卢安来	数组推导	04-20 17:49	314B	CPP14	正确	100	15ms	2.894MB

图 1: 提交结果

1 202109-1 数组推导

- 时间限制：1.0 s.
- 空间限制：512.0 MiB.

1.1 题目

1.1.1 题目描述

$A_1, A_2, \cdots, A_n$ 是一个由 n 个**自然数**（即非负整数）组成的数组。在此基础上，我们用数组 $B_1 \cdots B_n$ 表示 A 的前缀最大值。

$$B_i = \max\{A_1, A_2, \cdots, A_i\}$$

如上所示，B_i 定义为数组 A 中前 i 个数的最大值。

根据该定义易知 $A_1 = B_1$，且随着 i 的增大，B_i 单调不降。

此外，我们用 $\text{sum} = A_1 + A_2 + \cdots + A_n$ 表示数组 A 中 n 个数的总和。

现已知数组 B，我们想要根据 B 的值来反推数组 A。

显然，对于给定的 B，A 的取值可能并不唯一。试计算，在数组 A 所有可能的取值情况中，sum 的最大值和最小值分别是多少?

1.1.2 子任务

50% 的测试数据满足数组 B 单调递增，即 $0 < B_1 < B_2 < \cdots < B_n < 10^5$；

全部的测试数据满足 $n \leq 100$ 且数组 B 单调不降，即 $0 \leq B_1 \leq B_2 \leq \cdots \leq B_n \leq 10^5$。

1.2 解法 A（50 分）

1.2.1 原理阐释

当满足 B 单调递增时，我们可以证明 $A_i = B_i (\forall i)$，过程如下。

由 $B_{i+1} = \max\limits_{j=1}^{i+1}\{A_j\}$，有 $A_{i+1} \leq B_{i+1}$。

假设 $A_{i+1} < B_{i+1}$，则 $B_{i+1} = \max\limits_{j=1}^{i+1}\{A_j\} = \max\{B_i, A_{i+1}\} < \max\{B_{i+1}, B_{i+1}\} = B_{i+1}$，导出矛盾。从而假设不成立，我们证明了 $A_{i+1} = B_{i+1}$。

综合 $A_1 = B_1$，我们知道在此条件下 A 的取值唯一，即为 $A_i = B_i(\forall i)$，答案即为数组 B 的和。

对数组 B 进行求和输出，时间复杂度为 $\Theta(n)$，空间复杂度为 $\Theta(1)$。

1.2.2 C++ 代码实现

202109-1-subtask1.cpp

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

int main(){
        istream_iterator<int> in{cin};
        [[maybe_unused]] int n=*in++;
        int ans{accumulate(in,istream_iterator<int>{},0)};
        cout<<ans<<endl<<ans<<endl;
        return 0;
}
```

1.3 解法 B（100 分）

1.3.1 原理阐释

与解法 A 相比，多出了 $B_{i+1} = B_i$ 的情形，此时不难看出，A_{i+1} 可取 0 到 B_{i+1} 之间的任何值，从而由自然数加法的保序性，可以推出，欲使答案最大，需取 $A_{i+1} = B_{i+1}$，欲使得答案最小，需取 $A_{i+1} = 0$。

进一步整理，即得最大答案为数组 B 的和，最小答案为数组 B 中不同元素的和。

时间复杂度为 $\Theta(n)$，空间复杂度按照去重方法可以有所不同，常见的有 $\Theta(n)$ 和 $\Theta(1)$。

1.3.2 C++ 代码实现

202109-1.cpp

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

int main(){
```

```
5     istream_iterator<int> in{cin};
6     [[maybe_unused]] int n=*in++;
7     vector<int> v(in,istream_iterator<int>{});
8     int ans1{accumulate(v.begin(),v.end(),0)};
9     int ans2{accumulate(v.begin(),unique(v.begin(),v.end()),0)};
10    cout<<ans1<<endl<<ans2<<endl;
11    return 0;
12 }
```

1.3.3 提交结果

3632318	卢安来	卢安来	箱根山岳险天下	04-20 17:49	4.299KB	CPP14	正确	100	734ms	22.90MB
3632317	卢安来	卢安来	收集卡牌	04-20 17:49	559B	CPP14	正确	100	93ms	43.37MB
3632316	卢安来	卢安来	脉冲神经网络	04-20 17:49	2.339KB	CPP14	正确	100	890ms	7.609MB
3632315	卢安来	卢安来	非零段划分	04-20 17:49	827B	CPP14	正确	100	78ms	13.36MB
3632314	卢安来	卢安来	数组推导	04-20 17:49	314B	CPP14	正确	100	15ms	2.894MB

图 2: 提交结果

2 202109-2 非零段划分

- 时间限制：1.0 s.
- 空间限制：512.0 MiB.

2.1 题目

2.1.1 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。

2.1.2 子任务

70% 的测试数据满足 $n \le 1000$；

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

2.2 解法 A（70 分）

2.2.1 原理阐释

考虑一个朴素的办法，枚举 p 的取值，每次用 $\Theta(n)$ 的时间完成修改操作并统计非零段的个数，考虑到 A 的值域有限，这个方法可以取得不错的速度。

时间复杂度为 $\Theta(n\,|A|)$，空间复杂度为 $\Theta(n)$。

2.2.2 C++ 代码实现

202109-2-subtask1.cpp

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

int getCnt(const vector<int>& v){
        int cnt=0;
        for(size_t i=0;i<v.size();++i)
                if((!i||v[i-1]==0)&&v[i])
                        ++cnt;
        return cnt;
}

int main(){
        ios::sync_with_stdio(false);
        istream_iterator<int> in{cin};
        [[maybe_unused]] int n{*in++};
        vector<int> v(in,istream_iterator<int>{});
        int V{*max_element(v.begin(),v.end())};
        int val{getCnt(v)};
        int ans{val};
        for(int p=2;p<=V;++p){
                for(auto &i:v)
                        if(i<p)
                                i=0;
                ans=max(ans,getCnt(v));
        }
        cout<<ans<<endl;
        return 0;
}
```

2.3 解法 B（100 分）

2.3.1 原理阐释

注意到若 $p+1$ 在 A 中没有出现，则取 p 和取 $p+1$ 后 A 的状态没有改变，从而我们得知只需要处理 A 中出现了的值即可。

更进一步地，若我们知道了需要修改的值和其在 A 中的位置，我们可以在常数级别的时间内快速维护答案。从而时间复杂度均摊为 $\Theta(n)$。

总而言之，我们只需要简单维护每一个值出现的位置，并利用其快速维护答案即可。时间复杂度为 $\Theta(n+|A|)$，空间复杂度为 $\Theta(n+|A|)$。

2.3.2 C++ 代码实现

202109-2.cpp

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

int getCnt(const vector<int>& v){
    int cnt=0;
    for(size_t i=0;i<v.size();++i)
        if((!i||v[i-1]==0)&&v[i])
            ++cnt;
    return cnt;
}

int main(){
    ios::sync_with_stdio(false);
    istream_iterator<int> in{cin};
    [[maybe_unused]] int n{*in++};
    vector<int> v(in,istream_iterator<int>{});
    int V{*max_element(v.begin(),v.end())};
    using vit=vector<int>::iterator;
    vector<vector<vit>> bucket(V+1);
    for(auto it=v.begin();it!=v.end();++it)
        bucket[*it].emplace_back(it);
    bucket[0].clear();
    int val{getCnt(v)};
    int ans{val};
    for(const auto& buc:bucket){
```

```
26          for(const auto& it:buc){
27              int vLef{it==v.begin()?0:*(it-1)},vRig{(it+1)==v.end()?0:*(it+1)};
28              if(vLef<*it&&*it>vRig) --val;
29              if(vLef>*it&&*it<=vRig) ++val;
30              *it=0;
31          }
32          ans=max(ans,val);
33      }
34      cout<<ans<<endl;
35      return 0;
36  }
```

2.3.3 提交结果

3632318	卢安来	卢安来	箱根山岳险天下	04-20 17:49	4.299KB	CPP14	正确	100	734ms	22.90MB
3632317	卢安来	卢安来	收集卡牌	04-20 17:49	559B	CPP14	正确	100	93ms	43.37MB
3632316	卢安来	卢安来	脉冲神经网络	04-20 17:49	2.339KB	CPP14	正确	100	890ms	7.609MB
3632315	卢安来	卢安来	非零段划分	04-20 17:49	827B	CPP14	正确	100	78ms	13.36MB
3632314	卢安来	卢安来	数组推导	04-20 17:49	314B	CPP14	正确	100	15ms	2.894MB

图 3: 提交结果

3 202109-3 脉冲神经网络

- 时间限制：1.0 s.
- 空间限制：512.0 MiB.

3.1 题目

3.1.1 题目背景

在本题中，你需要实现一个 SNN（spiking neural network，脉冲神经网络）的模拟器。一个 SNN 由以下几部分组成：

1. 神经元：按照一定的公式更新内部状态，接受脉冲并可以发放脉冲
2. 脉冲源：在特定的时间发放脉冲
3. 突触：连接神经元-神经元或者脉冲源-神经元，负责传递脉冲

3.1.2 题目描述

神经元会按照一定的规则更新自己的内部状态。本题中，我们对时间进行离散化处理，即设置一个时间间隔 Δt，仅考虑时间间隔整数倍的时刻 $t = k\Delta t(k \in \mathbb{Z}_+)$，按照下面的公式，从 $k-1$ 时刻的取值计算 k 时刻变量的取值：

$$v_k = v_{k-1} + \Delta t(0.04v_{k-1}^2 + 5v_{k-1} + 140 - u_{k-1}) + I_k$$

$$u_k = u_{k-1} + \Delta ta(bv_{k-1} - u_{k-1})$$

其中 v 和 u 是神经元内部的变量，会随着时间而变化，a 和 b 是常量，不会随着时间变化；其中 I_k 表示该神经元在 k 时刻接受到的所有脉冲输入的强度之和，如果没有接受到脉冲，那么 $I_k = 0$。当进行上面的计算后，如果满足 $v_k \geq 30$，神经元会发放一个脉冲，脉冲经过突触传播到其他神经元；同时，v_k 设为 c 并且 u_k 设为 $u_k + d$，其中 c 和 d 也是常量。图 1 展示了一个神经元 v 变量随时间变化的曲线。

突触表示的是神经元-神经元、脉冲源-神经元的连接关系，包含一个入结点和一个出结点（可能出现自环和重边）。当突触的入结点（神经元或者脉冲源）在 k 时刻发放一个脉冲，那么在传播延迟 $D(D > 0)$ 个时刻以后，也就是在 $k + D$ 时刻突触的出结点（神经元）会接受到一个强度为 w 的脉冲。

脉冲源在每个时刻以一定的概率发放一个脉冲，为了模拟这个过程，每个脉冲源有一个参数 r ，并统一采用以下的伪随机函数：

C++ 版本：

```cpp
static unsigned long next = 1;

/* RAND_MAX assumed to be 32767 */
int myrand(void) {
    next = next * 1103515245 + 12345;
    return((unsigned)(next/65536) % 32768);
}
```

在每个时间刻，按照编号顺序从小到大，每个脉冲源调用一次上述的伪随机函数，当 $r > \text{myrand}()$ 时，在当前时间刻发放一次脉冲，并通过突触传播到神经元。

进行仿真的时候，已知 0 时间刻各个神经元的状态，从 1 时间刻开始按照上述规则进行计算，直到完成 T 时刻的计算，再输出 T 时刻神经元的 v 值和发放的脉冲次数分别的最小值和最大值。

规定输入数据中结点按如下方式顺序编号：$[0, N-1]$ 为神经元的编号，$[N, N+P-1]$ 为脉冲源的编号。

代码中请使用双精度浮点类型。

3.1.3 子任务

子任务	T	N	S	P	D	分值
1	10^2	10^2	10^2	10^2	10^2	30
2	10^3	10^3	10^3	10^3	10^3	40
3	10^5	10^3	10^3	10^3	10	30

3.2 解法 A（30 分）

3.2.1 原理阐释

按秒维护，每秒处理神经元时枚举邻接情况，用邻接矩阵处理，则处理单个神经元复杂度为 $\Theta((n+p)D)$。

从而总的时间复杂度为 $\Theta(Tn(n+p)D)$，空间复杂度为 $\Theta((n+p)^2)$。

3.3 解法 B（70 分）

3.3.1 原理阐释

与解法 A 相同的思路，但换用邻接表存图，此时每一秒处理每个神经元产生脉冲的总的复杂度就变为 $\Theta(sD)$。

从而总的时间复杂度为 $\Theta(T(n+sD+p))$，空间复杂度为 $\Theta(n+p+s)$。

实际上本解法因为题目时间限制过于紧张，只能拿到 66 分。

3.4 解法 C（100 分）

3.4.1 原理阐释

若每次都枚举 D 时刻之前各个神经元的状态，则代价十分昂贵。注意到 D 很小，则我们可以用滚动数组直接存储其余神经元对特定神经元在特定时间贡献的总和。

这个值每次处理神经元产生脉冲时直接简单相加，处理神经元新状态时读取并清空即可。

总的时间复杂度为 $\Theta(T(n+p+s))$，空间复杂度为 $\Theta(nD+n+p+s)$。

由于时间限制过于紧张，如果仅仅按照上面的思路编写代码难以通过，下面我将介绍几种在不使用其他条件的情况下加速程序的办法。

1. **取模优化**：若对于两个都在完全剩余类中的数，它们的加法取模可以转化为简单的加法和减法，见代码中的 “add” 函数；
2. **局部性优化**：增强代码的局部性，对于神经元的三个步骤（更新，清空滚动数组，脉冲）分开处理，增强代码的内存局部性从而利用 CPU 缓存加速。
3. **预处理优化**：简化题目给定的表达式，将冗余的计算参数提前算好，见代码中的 “A,B,C” 参数和 “ap”。
4. **浮点优化**：题目虽然要求使用 “double” 类型的浮点数，但可以发现，题目的算数有效位数很低，可以在某些精度要求不高的变量（除了 $u, v, \Delta t$ 都要求不高）上使用 “float” 类型，加速浮点运算。
5. **循环展开**：这题的计算稍有复杂之处，若欲有效循环展开则需要使用额外空间存储中间变量，内存限制稍紧，故不采用。

3.4.2 C++ 代码实现

这份代码使用了取模优化、局部性优化、预处理优化、浮点优化，其中取模优化更进一步，利用了位运算优化。可以轻松通过本题。

202109-3.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
#define reg register

namespace ProblemCode{
    static unsigned long next = 1;

    /* RAND_MAX assumed to be 32767 */
    inline int myrand(void) {
        next = next * 1103515245 + 12345;
        return((unsigned)(next>>16)&32767);
    }
}

constexpr int MAXN=1e3;
constexpr int MAXP=1e3;
constexpr int MAXS=1e3;
constexpr int MAXD=1024;

struct Netural{
    double v,u;
    float ap,b,c,d;
    bool operator<(const Netural& a)const{
        return v<a.v;
    }
};

struct Edge{
    int v;
    float w;
    int D;
};

int n,s,p,T;
double deltaT;
```

```
36 Netural net[MAXN];
37 int r[MAXP];
38 int eid,head[MAXN+MAXP],nxt[MAXS+1];
39 Edge e[MAXS+1];
40 int cnt[MAXN];
41 float Q[MAXD+1][MAXN];
42
43 inline int add(reg int a,reg int b){
44     return (a+b)&(MAXD-1);
45     //reg int sum=a+b;
46     //return sum>=MAXD?sum-MAXD:sum;
47 }
48
49 inline void Add_Edge(reg int u,reg int v,reg float w,reg int D){
50     nxt[++eid]=head[u];
51     e[eid].v=v,e[eid].w=w,e[eid].D=D;
52     head[u]=eid;
53     return;
54 }
55
56 int main(){
57     scanf("%d%d%d%d%lf",&n,&s,&p,&T,&deltaT);
58     for(int s{0},Rn{};s!=n;s+=Rn){
59         static double v,u,ap,b,c,d;
60         scanf("%d%lf%lf%lf%lf%lf%lf",&Rn,&v,&u,&ap,&b,&c,&d);
61         ap*=deltaT;
62         for(int i{s};i<s+Rn;++i){
63             net[i].v=v;
64             net[i].u=u;
65             net[i].ap=ap;
66             net[i].b=b;
67             net[i].c=c;
68             net[i].d=d;
69         }
70     }
71     for(int i=0;i<p;++i)
72         scanf("%d",&r[i]);
73     for(int i=0;i<s;++i){
74         static int s,t;
75         static double w;
76         static int D;
```

```
77          scanf("%d%d%lf%d",&s,&t,&w,&D);
78          Add_Edge(s,t,w,D);
79      }
80      const float A=0.04*deltaT;
81      const float B=5.0*deltaT;
82      const float C=140.0*deltaT;
83      for(reg int t{1},index{1%MAXD};t<=T;++t,index=add(index,1)){
84          for(reg int nid{0};nid<n;++nid){
85              static double v,u;
86              v=net[nid].v,u=net[nid].u;
87              net[nid].v=v+A*v*v+B*v+C-deltaT*u+Q[index][nid],net[nid].u=u+net[nid].ap*(net[nid].b*v-u
                    );
88          }
89          memset(Q[index],0,n*sizeof(float));
90          for(reg int nid{0};nid<n;++nid)
91              if(net[nid].v>=30.0){
92                  net[nid].v=net[nid].c,net[nid].u+=net[nid].d;
93                  ++cnt[nid];
94                  for(reg int i=head[nid];i;i=nxt[i])
95                      Q[add(index,e[i].D)][e[i].v]+=e[i].w;
96              }
97          for(reg int pid{0};pid<p;++pid)
98              if(r[pid]>ProblemCode::myrand())
99                  for(reg int i=head[n+pid];i;i=nxt[i])
100                     Q[add(index,e[i].D)][e[i].v]+=e[i].w;
101     }
102     double ans1{min_element(net,net+n)->v},ans2{max_element(net,net+n)->v};
103     int ans3{*min_element(cnt,cnt+n)},ans4{*max_element(cnt,cnt+n)};
104     printf("%.3f %.3f\n%d %d\n",ans1,ans2,ans3,ans4);
105     return 0;
106 }
```

3.4.3 提交结果

3632318	卢安来	卢安来	箱根山岳险天下	04-20 17:49	4.299KB	CPP14	正确	100	734ms	22.90MB
3632317	卢安来	卢安来	收集卡牌	04-20 17:49	559B	CPP14	正确	100	93ms	43.37MB
3632316	卢安来	卢安来	脉冲神经网络	04-20 17:49	2.339KB	CPP14	正确	100	890ms	7.609MB
3632315	卢安来	卢安来	非零段划分	04-20 17:49	827B	CPP14	正确	100	78ms	13.36MB
3632314	卢安来	卢安来	数组推导	04-20 17:49	314B	CPP14	正确	100	15ms	2.894MB

图 4: 提交结果

4 202109-4 收集卡牌

- 时间限制：1.0 s.
- 空间限制：512.0 MiB.

4.1 题目

4.1.1 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。

4.1.2 子任务

对于 20% 的数据，保证 $1 \leq n, k \leq 5$。

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 $1 \leq n \leq 16$，$1 \leq k \leq 5$，所有的 p_i 满足 $p_i \geq \frac{1}{1000}$ 且 $\sum_{i=1}^{n} p_i = 1$。

4.2 解法 A（20 分）

4.2.1 原理阐释

我们考虑设计状态 $f_{a_1,\cdots,a_n}$ 表示抽到第 i 种卡牌 a_i 次对应的概率，则状态转移方程为

$$f_{a_1,\cdots,a_i,\cdots,a_n} = \sum_{i=1}^{n} [a_i \geq 1] f_{a_1,\cdots,a_i-1,\cdots,a_n} p_i$$

边界条件为

$$f_{0,\cdots,0} = 1$$

。

对于状态 $f_{a_1,\cdots,a_n}$，总的抽卡次数 $\mathrm{tot}=\sum\limits_{i=1}^{n}a_i$，抽到有用卡牌（没有变成硬币）数目为 $\mathrm{cnt}=\sum\limits_{i=1}^{n}[a_i\neq 0]$，从而硬币数目 $\mathrm{coin}=\mathrm{tot}-\mathrm{cnt}$。

此时答案的计算式可导出为

$$\mathbb{E}=\sum_{\mathrm{coin}\geq(n-\mathrm{cnt})\times k} f_{a_1,\cdots,a_n}\mathrm{tot},$$

代入以上表达式，即得计算式

$$\mathbb{E}=\sum_{\left(\sum\limits_{i=1}^{n}a_i-\sum\limits_{i=1}^{n}[a_i\neq 0]\right)\geq\left(n-\sum\limits_{i=1}^{n}[a_i\neq 0]\right)\times k}\left(f_{a_1,\cdots,a_n}\sum_{i=1}^{n}a_i\right).$$

递推过程的时间复杂度为 $\Theta((nk)^n)$，求答案过程的时间复杂度为 $((nk)^n)$，空间复杂度为 $\Theta((nk)^n)$。

4.3 解法 B（40 分）

4.3.1 原理阐释

注意到 p_i 全相等（$=\frac{1}{n}$），从而对于解法 A 中的 $f_{a_1,\cdots,a_n}$，我们由多重排列公式可以有

$$f_{a_1,\cdots,a_n}=\frac{\left(\sum\limits_{i=1}^{n}a_i\right)!}{\prod\limits_{i=1}^{n}(a_i!)}p^{\sum\limits_{i=1}^{n}a_i}.$$

代入求答案表达式可以得出解法，不具有一般性，故此处不再赘述。

4.3.2 C++ 代码实现

4.4 解法 C（100 分）

4.4.1 原理阐释

之所以我们的算法不够优秀，是因为状态设计的问题，我们从状态设计重新入手，设计一个优秀的状态。

注意到抽到过一次某种卡牌后，我们再抽到它不论多少次都会变成硬币，因此我们发现，对于一张卡牌，重要的是有没有抽到过，对于整个过程，重要的是有多少个硬币。

从而设计状态 $f_{i,\mathbf{S}}$，表示抽卡 i 次，抽到卡牌的集合二进制表示为 $\mathbf{S}$ 的概率，则已经抽到卡牌的数量即为 $\mathrm{cnt}(\mathbf{S})$，即 $\mathbf{S}$ 的二进制表示中 1 的个数，抽到硬币的数目即为 $i-\mathrm{cnt}(\mathbf{S})$。

状态转移方程（贡献形式）为

$$f_{i,\mathbf{S}} \to_{j=1}^{n} f_{i+1,\mathbf{S}\cup\{j\}} p_j,$$

边界条件为

$$f_{0,\varnothing} = 1.$$

答案计算式同解法 A，化为

$$\mathbb{E} = \sum_{i-\mathrm{cnt}(\mathbf{S})\geq(n-\mathrm{cnt}(\mathbf{S}))\times k} f_{i,\mathbf{S}} i.$$

算法的时间复杂度为 $\Theta(n^2k2^n)$，空间复杂度为 $\Theta(nk2^n)$。

值得注意的是，本题不提供 “Special Judge”，且要求输出到小数点后 10 位，此时浮点精度难以保证。因此，为了保障浮点精度，我们需要认清浮点数加法不满足加法交换律这一特性，选择最佳的答案求和顺序。

最简单的是将答案的每一项从小到大排序后再相加，此时精度比较高，可以满足题目的精度需要，额外添加了排序的复杂度（使得代码运行时间从 60 ms 增加到 120 ms 左右）。

最准确的是将答案的每项存储在堆中，每次提取最小的两个相加后放回，虽复杂度与排序法一致但常数颇高（实际运行时间 300 ms 左右），故不采用。

此外还有两种令人诟病的方式，一是换用 “long double”，但这平添了不必要的时空负担，二是网上代码常见的交换循环顺序，这实际上是不通用且不负责任的做法。

4.4.2 C++ 代码实现

为提高浮点精度，使用排序升序相加的方法。

202109-4.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
#define cnt __builtin_popcount

int main(){
```

```
    int n,k;
    cin>>n>>k;
    vector<double> p(n);
    for(int i=0;i<n;++i)
        cin>>p[i];
    vector<vector<double>> f(n*k+1);
    for(int i=0;i<=n*k;++i) f[i].resize(1<<n);
    f[0][0]=1.0;
    for(int i=0;i<n*k;++i)
        for(int S=0;S<(1<<n);++S)
            if(i-cnt(S)<(n-cnt(S))*k)
                for(int j=0;j<n;++j)
                    f[i+1][S|(1<<j)]+=f[i][S]*p[j];
    vector<double> ans;
    for(int i=0;i<=n*k;++i)
        for(int S=0;S<(1<<n);++S)
            if(i-cnt(S)>=(n-cnt(S))*k)
                ans.emplace_back(f[i][S]*i);
    sort(ans.begin(),ans.end());
    printf("%.10lf\n",accumulate(ans.begin(),ans.end(),0.0));
    return 0;
}
```

4.4.3 提交结果

3632318	卢安来	卢安来	箱根山岳险天下	04-20 17:49	4.299KB	CPP14	正确	100	734ms	22.90MB
3632317	卢安来	卢安来	收集卡牌	04-20 17:49	559B	CPP14	正确	100	93ms	43.37MB
3632316	卢安来	卢安来	脉冲神经网络	04-20 17:49	2.339KB	CPP14	正确	100	890ms	7.609MB
3632315	卢安来	卢安来	非零段划分	04-20 17:49	827B	CPP14	正确	100	78ms	13.36MB
3632314	卢安来	卢安来	数组推导	04-20 17:49	314B	CPP14	正确	100	15ms	2.894MB

图 5: 提交结果

5 202109-5 箱根山岳险天下

- 时间限制：5.0 s.
- 空间限制：512.0 MiB.

5.1 题目

5.1.1 题目背景

“你知道对长跑选手来说，最棒的赞美是什么吗？”

“是‘快’吗？”

“不，是‘强’，”清濑说，“光跑得快，是没办法在长跑中脱颖而出的。天候、场地、比赛的发展、体能，还有自己的精神状态–长跑选手必须冷静分析这许多要素，即使面对再大的困难，也要坚忍不拔地突破难关。长跑选手需要的，是真正的‘强’。所以我们必须把‘强’当作最高的荣誉，每天不断跑下去。”

不论阿走或其他房客，全都全神贯注地聆听清濑的话。

“看了你这三个月来的表现，我越来越相信自己没看错人，”清濑接着说，“你很有天分，也很有潜力。所以呢，阿走，你一定要更相信自己，不要急着想一飞冲天。变强需要时间，也可以说它永远没有终点。长跑是值得一生投入的竞赛，有些人即使老了，仍然没有放弃慢跑或马拉松运动。”

—三浦紫苑《强风吹拂》

箱根驿传（正式名称为东京箱根间往复大学驿传竞走）是日本一项在每年 1 月 2-3 日举行的驿站接力赛，由关东学生田径联盟主办，关东的每所高校都有机会参加。在日本，箱根驿传是新年假期必看的比赛，许多家庭会一边吃年糕汤一边欣赏激烈的比赛。

今年，京都大学也想派出长跑队参加箱根驿传，田径部的长跑教练组织起一批预备役运动员，并开展了严苛的训练。

5.1.2 题目描述

京都大学的训练一共会持续 m 天，在训练过程中正式队员的名单可能发生变化。简单起见，我们约定在且仅在第 $t(1 \le t \le m)$ 天结束时，会有以下三种事件之一发生：

1. 有一个学生跑 10 km 的速度达到了正式队员要求，教练将其作为最后一名纳入正式队员的名单中，这个学生的强度为 x；或者速度排名在最后一位的正式队员，由

于速度过慢，而被从正式队员的名单中淘汰。

- 在训练过程中，我们假定队员的速度的相对排名不会发生变化，与强度无关。
- 严苛的教练制订了残酷的规则：被淘汰的学生虽然依然会跟大家一起训练，但将不能再次加入本年度参加箱根驿传的正式队员的名单中。

2. 由于近日的训练，第 s 天结束时速度排名为 l 至 r 的选手的强度有了变化，变为此前的 y 倍。
3. 教练在深夜想知道近日训练的效果，于是他统计了第 s 天结束时速度排名为 l 至 r 的选手目前（即第 t 天结束时）强度的和。由于这个结果可能很大，方便起见我们只考虑其模 p 的值。

出于学生们的隐私考虑，事件日志有可能会被加密。

5.1.3 子任务

$1 \le m \le 3 \times 10^5$，$2 \le p < 2^{30}$，mode $\in \{0, 1\}$。

测试点	特殊性质	mode
1	$m \le 5000$	1
2	事件 1 中 $x > 0$	1
3	没有事件 2	1
4	事件 1 中 x 在 $0, 1$ 中随机选取	0
5	$r - l \le 10$	0
6	$r - l \le 10$	1
7, 8	无	0
9, 10	无	1

5.2 解法 A（100 分）

5.2.1 原理阐释

这个题目描述理解难度较大，我们首先解读一下题目：

- 每个人可抽象为一个节点。
- 排名由前到后排成一条链，那么加入操作相当于在最后加入一个节点，删除操作相当于回退一个节点。

- 这些节点形成了若干个有根树的集合。
- 若我们加多一个空的根节点，那么就变成一个有根树。
- 四种操作分别对应
  1. 当前节点（初始为我们的空根节点）后加上一个叶子节点；
  2. 当前节点设为当前节点的父亲节点；
  3. 修改 s 时对应节点的 l, r 级祖先之间的链；
  4. 查询 s 时对应节点的 l, r 级祖先之间的链。

本题出题人给了许多特殊性质，分别为：

- 离线：对应解法为树上数据结构，例如树链剖分。
- $x > 0$：对应解法为链上数据结构，例如线段树。
- 没有事件 2：对应解法为无修动态数据结构，例如倍增 + 树上缀和。
- 事件 1 中 x 在 $0, 1$ 随机选取：对应解法为树高较矮时的统计，例如朴素求和。
- $r - l \leq 10$：对应解法为树上短链修改查询，例如倍增 + 朴素求和。

但上述这些解法繁琐而失去一般性，下面我们根据出题人的提示，探索正确解法。

毋庸置疑，本题考察树上数据结构，且其要求为：强制在线、修改树结构、修改链、询问链。显然，我们想到的最好解决办法即为 Link-Cut Tree。

直接实现即可，时间复杂度 $\Theta(m \log m)$，空间复杂度 $\Theta(m)$。

5.2.2 C++ 代码实现

202109-5.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;
#define reg register
#define getchar() (p1==p2&&(p2=(p1=buf)+fread(buf,1,1<<22,stdin),p1==p2)?EOF:*p1++)
static char buf[1<<22],*p1=buf,*p2=buf;
#define flush() (fwrite(wbuf,1,wp1,stdout),wp1=0)
#define putchar(c)(wp1==wp2&&(flush(),0),wbuf[wp1++]=c)
static char wbuf[1<<22];int wp1;const int wp2=1<<22;
inline int read(void){
    reg char ch=getchar();
    reg int res=0;
    while(!isdigit(ch)) ch=getchar();
    while(isdigit(ch)) res=10*res+(ch-'0'),ch=getchar();
    return res;
```

```
15 }
16
17 inline void writeln(reg int x){
18     static char buf[32];
19     reg int p=-1;
20     if(x==0) putchar('0');
21     else while(x) buf[++p]=(x%10)^'0',x/=10;
22     while(~p) putchar(buf[p--]);
23     putchar('\n');
24     return;
25 }
26
27 const int MAXM=3e5+5;
28
29 int m,p,T;
30 int ptr[MAXM],fa[MAXM];
31
32 inline int add(reg int a,reg int b){
33     reg int sum=a+b;
34     return sum>=p?sum-p:sum;
35 }
36
37 inline int sub(reg int a,reg int b){
38     reg int sum=a-b;
39     return sum<0?sum+p:sum;
40 }
41
42 inline int mul(reg int a,reg int b){
43     return 1ll*a*b%p;
44 }
45
46 namespace LCT{
47     #define fa(u) unit[u].fa
48     #define ch(u) unit[u].ch
49     #define siz(u) unit[u].siz
50     #define tRev(u) unit[u].tRev
51     #define val(u) unit[u].val
52     #define sum(u) unit[u].sum
53     #define tMul(u) unit[u].tMul
54     #define lson(u) ch(u)[0]
55     #define rson(u) ch(u)[1]
```

```
56      struct Node{
57          int fa;
58          int ch[2];
59          int siz;
60          bool tRev;
61          int val;
62          int sum;
63          int tMul;
64      };
65      int tot;
66      Node unit[MAXM];
67      inline int getNode(reg int v){
68          reg int p=++tot;
69          siz(p)=1,val(p)=sum(p)=v,tMul(p)=1;
70          return p;
71      }
72      inline bool get(reg int p){
73          return rson(fa(p))==p;
74      }
75      inline bool isRt(reg int p){
76          return lson(fa(p))!=p&&rson(fa(p))!=p;
77      }
78      inline void pushup(reg int p){
79          siz(p)=siz(lson(p))+siz(rson(p))+1;
80          sum(p)=add(add(sum(lson(p)),sum(rson(p))),val(p));
81          return;
82      }
83      inline void Rev(reg int p){
84          swap(lson(p),rson(p));
85          tRev(p)=!tRev(p);
86          return;
87      }
88      inline void Mul(reg int p,reg int val){
89          val(p)=mul(val(p),val);
90          sum(p)=mul(sum(p),val);
91          tMul(p)=mul(tMul(p),val);
92          return;
93      }
94      inline void pushdown(reg int p){
95          if(tRev(p)){
96              if(lson(p)) Rev(lson(p));
```

```
97              if(rson(p)) Rev(rson(p));
98              tRev(p)=false;
99          }
100         if(tMul(p)!=1){
101             if(lson(p)) Mul(lson(p),tMul(p));
102             if(rson(p)) Mul(rson(p),tMul(p));
103             tMul(p)=1;
104         }
105         return;
106     }
107     inline void rotate(reg int p){
108         if(isRt(p)) return;
109         reg int f=fa(p),ff=fa(f),dir=get(p);
110         if(!isRt(f)) ch(ff)[get(f)]=p;
111         if(ch(p)[dir^1]) fa(ch(p)[dir^1])=f;
112         ch(f)[dir]=ch(p)[dir^1];
113         ch(p)[dir^1]=f,fa(f)=p;
114         fa(p)=ff;
115         pushup(f),pushup(p);
116         return;
117     }
118     inline void update(reg int p){
119         if(!isRt(p)) update(fa(p));
120         pushdown(p);
121         return;
122     }
123     inline void splay(reg int x){
124         update(x);
125         for(reg int y=fa(x);!isRt(x);rotate(x),y=fa(x))
126             if(!isRt(y))
127                 rotate(get(x)==get(y)?y:x);
128         pushup(x);
129         return;
130     }
131     inline void access(reg int p){
132         for(reg int las=0;p;las=p,p=fa(p))
133             splay(p),rson(p)=las,pushup(p);
134         return;
135     }
136     inline void makeRt(reg int p){
137         access(p),splay(p),Rev(p);
```

```
138         return;
139     }
140     inline void split(reg int x,reg int y){
141         makeRt(x),access(y),splay(y);
142         return;
143     }
144     inline void link(reg int x,reg int y){
145         makeRt(x),makeRt(y);
146         fa(x)=y;
147         pushup(y);
148         return;
149     }
150     inline int kth(reg int u,reg int k){
151         while(true){
152             pushdown(u);
153             if(k<=siz(lson(u)))
154                 u=lson(u);
155             else if(siz(lson(u))+1==k)
156                 return u;
157             else
158                 k-=siz(lson(u))+1,u=rson(u);
159         }
160         return -1;
161     }
162     inline void update(int x,int l,int r,int v){
163         split(1,x);
164         reg int L=kth(x,l+1),R=kth(x,r+1);
165         split(L,R);
166         Mul(R,v);
167         return;
168     }
169     inline int query(reg int x,reg int l,reg int r){
170         split(1,x);
171         reg int L=kth(x,l+1),R=kth(x,r+1);
172         split(L,R);
173         return sum(R);
174     }
175     #undef fa
176     #undef ch
177     #undef siz
178     #undef tRev
```

```
179     #undef val
180     #undef sum
181     #undef tMul
182     #undef lson
183     #undef rson
184 }
185
186 int main(void){
187     LCT::getNode(0);
188     m=read(),p=read(),T=read();
189     reg int ans=0;
190     reg int p=1;
191     for(reg int i=1;i<=m;++i){
192         static int opt,x,s,l,r;
193         opt=read();
194         switch(opt){
195             case 1:{
196                 x=read();
197                 if(T) x^=ans;
198                 if(!x) p=fa[p];
199                 else{
200                     reg int q=LCT::getNode(x);
201                     LCT::link(q,p);
202                     fa[q]=p,p=q;
203                 }
204                 ptr[i]=p;
205                 break;
206             }
207             case 2:{
208                 s=read(),l=read(),r=read(),x=read();
209                 if(T) x^=ans;
210                 LCT::update(ptr[s],l,r,x);
211                 ptr[i]=ptr[i-1];
212                 break;
213             }
214             case 3:{
215                 s=read(),l=read(),r=read();
216                 writeln(ans=LCT::query(ptr[s],l,r));
217                 ptr[i]=ptr[i-1];
218                 break;
219             }
```

```
220         }
221     }
222     flush();
223     return 0;
224 }
```

5.2.3 提交结果

3632318	卢安来	卢安来	箱根山岳险天下	04-20 17:49	4.299KB	CPP14	正确	100	734ms	22.90MB
3632317	卢安来	卢安来	收集卡牌	04-20 17:49	559B	CPP14	正确	100	93ms	43.37MB
3632316	卢安来	卢安来	脉冲神经网络	04-20 17:49	2.339KB	CPP14	正确	100	890ms	7.609MB
3632315	卢安来	卢安来	非零段划分	04-20 17:49	827B	CPP14	正确	100	78ms	13.36MB
3632314	卢安来	卢安来	数组推导	04-20 17:49	314B	CPP14	正确	100	15ms	2.894MB

图 6: 提交结果