第23次CSP解题报告

本次报告的撰写人：张晨阳 题目：202109，第23次CSP

题目 1：数组推导

题目描述：

给定一个由n个自然数组成的数组B，其中Bi等于数组A中前i个数的最大值。
求在所有可能的A中，A的总和sum的最大值和最小值。
n ≤ 100, 0 ≤ B_1 ≤ B_2 ≤... ≤Bn ≤ 10^5。

题解：

分为sum最大值和sum最小值两部分。

sum最大值：

对于每一个Ai，一定有Ai≤Bi，则sum的最大值即Bi的和。

sum最小值：

解法一：

因为数组B单调递增，若$Bi = Bi\text{-}1$，则说明0≤Ai≤Bi；若Bi>$Bi\text{-}1$，则说明$Ai = Bi$。
上述两种情况都取最小值即为所求。

时间复杂度$O(n)$，空间复杂度$O(n)$，期望得分100。

3630416	张晨阳	张晨阳	数组推导	04-18 22:08	470B	C++	正确	100	15ms	2.867MB

```language-none
#include<iostream>

using std::cin;
using std::cout;
using std::endl;

int Maxsum(int n, int *b)
{
        int sum = 0;
        for (int i = 0; i < n; i++)
                sum += b[i];
        return sum;
}
```

```
int Minsum(int n, int *b)
{
        int sum = b[0];
        for (int i = 1; i < n; i++)
                if (b[i] != b[i - 1])
                        sum += b[i];
        return sum;
}

int main()
{
        int n;
        int B[100];
        cin >> n;
        for (int i = 0; i < n; i++)
                cin >> B[i];
        cout << Maxsum(n, B) << endl;
        cout << Minsum(n, B) << endl;
        return 0;
}
```

解法二：

对空间复杂度的改进。
无需用数组存储读入的数据，读入数据流中的数据进行计算处理即可。

时间复杂度$O(n)$，空间复杂度$O(1)$，期望得分100。

3636101	张晨阳	张晨阳	数组推导	04-23 16:23	301B	C++	正确	100	15ms	2.898MB

```language-none
#include<iostream>

using std::cin;
using std::cout;
using std::endl;

int main()
{
        int n, B, summax = 0, summin = 0, next = -1;
        cin >> n;
        for (int i = 0; i < n; i++) {
                cin >> B;
                summax += B;
                if (B != next)
                        summin += B;
                next = B;
        }
        cout << summax << "\n" << summin << endl;
        return 0;
}
```

题目 2：非零段划分

题目描述：

给定一个长度为 n 的非负整数数组 $a_1, a_2, \cdots, an$，要求将小于正整数x的数全部变为0，求出非零段段数的最大值。
$n \leq 5 \times 10^5$, $0 \leq ai \leq 10^4$。

题解：

设 V 为数组中不同元素个数。

解法一：

$n \leq 1000$。

利用set的自动去重功能，统计不同元素的个数。枚举set中的每个元素作为选取的值x，遍历数组A，如果 $ai \geqslant x$ 且 $ai\text{-}1 < x$，则计数器+1。根据所有选取的值x，计算出非零段个数的最大值。

时间复杂度$O(Vnlogn)$，空间复杂度$O(n)$。可通过70%的数据。

3635931	张晨阳	张晨阳	非零段划分	04-23 15:17	592B	C++	运行超时	70	运行超时	5.097MB

```language-none
#include<iostream>
#include<set>

using namespace std;

int Count(int n, int* a,set<int> & s)
{
        int max = 0;
        for (set<int>::iterator it = s.begin(); it != s.end(); it++) {
                int count = 0, flag = 0;
                for (int i = 0; i < n; i++) {
                        if (a[i] >= *it && flag == 0 && *it != 0) {
                                count++;
                                flag = 1;
                        }
                        else if (a[i] < *it)
                                flag = 0;
                }
                if (count > max)
                        max = count;
        }
        return max;
}

int main()
{
```

```
        int n;
        int A[500000] = { 0 };
        set<int> s;
        cin >> n;
        for (int i = 0; i < n; i++) {
                cin >> A[i];
                s.insert(A[i]);
        }
        cout << Count(n, A, s) << endl;
        return 0;
}
```

解法二：

- 若一个数比旁边两个数都大，那p就+1；
- 若一个大一个小就不变；
- 若比两个都小就-1。

时间复杂度：$O(nlogn)$，空间复杂度：$O(n)$，期望得分：100。

3637216	张晨阳	张晨阳	非零段划分	04-24 00:10	499B	C++	正确	100	312ms	7.839MB

```language-none
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;

const int mx = 500005, mm = 10005;
vector<int> v[mm];
int n, m = 10000, x, p, ans;
bool vs[mx];

int main()
{
        cin >> n;
        for (int i = 0; i < n; i++) {
                cin >> x;
                v[x].push_back(i);
        }
        for (int i = m; i >= 1; i--) {
                for (vector<int>::iterator it = v[i].begin(); it !=
v[i].end(); it++) {
                        int t = *it;
                        p += 1 - vs[t - 1] - vs[t + 1];
                        vs[t] = 1;
                }
                ans = max(ans, p);
        }
        cout << ans << endl;
```

```
        return 0;
}
```

题目 3：脉冲神经网络

题目描述：

有一个图，每个节点分为脉冲源和神经元。每个节点有权值 (u, v)，计算方式为：
$vk = vk-1 + \Delta t(0.04vk\ 2\ -1 + 5vk-1 + 140 - uk - 1) + Ik$
$uk = uk-1 + \Delta t(bvk-1 - uk-1\ )$

脉冲源每秒有一个脉冲，按照题目给定的方式生成。节点间连单向边，激励能按照边在 D 时刻后传到下一个节点，然后每个节点根据当前的输入激励之和判断其是否大于等于 30，从而发生脉冲。

问 T 秒后所有节点的情况，及所有神经元脉冲次数的最值。

题解：

解法一：

直接遍历。

每个时刻遍历一遍脉冲源，判断是否脉冲，遍历每个突触，记录传播的Ik。
Ik[i][j],表示第i个神经元第j时刻收到的脉冲强度。
每个时刻遍历一遍神经元，同上。
遍历T时刻的所有神经元求的结果。

时间复杂度$O(Tn^2\ )$，空间复杂度$O(Tn)$，期望得分66。

3636973	张晨阳	张晨阳	脉冲神经网络	04-23 22:01	2.144KB	C++	运行超时	66	运行超时	253.8MB

```language-none
#include<iostream>
#include<algorithm>
#include<iomanip>
using namespace std;

static unsigned long nxt = 1;
/* RAND_MAX assumed to be 32767 */
int myrand(void) {
        nxt = nxt * 1103515245 + 12345;
        return((unsigned)(nxt / 65536) % 32768);
}

int N, S, P, T;
double dt;

class Shen {
public:
```

```
        double vlast, ulast;
        double v, u;
        double a, b, c, d;
}Sh[1000];

class Edg {
public:
        double w;
        int D;
        int start, end;
}E[1000];

int r[1000];
double Ik[1000][100000];

int main()
{
        cin >> N >> S >> P >> T;         //N神经元，S突触，P脉冲
        cin >> dt;
        int Rn,num = 0;
        double vv, uu, aa, bb, cc, dd;
        while (num < N) {
                cin >> Rn >> vv >> uu >> aa >> bb >> cc >> dd;
                for (int i = num; i < num + Rn; i++) {
                        Sh[i].vlast = Sh[i].v = vv;
                        Sh[i].ulast = Sh[i].u = uu;
                        Sh[i].a = aa;
                        Sh[i].b = bb;
                        Sh[i].c = cc;
                        Sh[i].d = dd;
                }
                num += Rn;
        }
        for (int i = 0; i < P; i++)
                cin >> r[i];
        for (int i = 0; i < S; i++)
                cin >> E[i].start >> E[i].end >> E[i].w >> E[i].D;
        //输入结束
        //脉冲
        for (int t = 1; t < T; t++)
                for (int i = 0; i < P; i++)
                        if (r[i] > myrand())
                                for (int j = 0; j < S; j++)
                                        if (E[j].start == i + N && t +
E[j].D <= T)
                                                Ik[E[j].end][E[j].D +
t] += E[j].w;
        //神经元
        int count[1000] = { 0 };
        for (int t = 1; t <= T; t++)
```

```
                for (int i = 0; i < N; i++) {
                        Sh[i].v = Sh[i].vlast + dt * (0.04 *
Sh[i].vlast * Sh[i].vlast + 5 * Sh[i].vlast + 140 - Sh[i].ulast) +
Ik[i][t];
                        Sh[i].u = Sh[i].ulast + dt * Sh[i].a * (Sh[i].b
* Sh[i].vlast - Sh[i].ulast);
                        if (Sh[i].v >= 30) {
                                count[i]++;
                                Sh[i].v = Sh[i].c;
                                Sh[i].u = Sh[i].u + Sh[i].d;
                                for (int j = 0; j < S; j++)
                                        if (E[j].start == i && t +
E[j].D <= T)
                                                Ik[E[j].end][E[j].D +
t] += E[j].w;
                        }
                        Sh[i].vlast = Sh[i].v;
                        Sh[i].ulast = Sh[i].u;
                }
        int maxC = count[0], minC = count[0];
        double maxV = Sh[0].v, minV = Sh[0].v;
        for (int i = 0; i < N; i++) {
                maxV = max(maxV, Sh[i].v);
                minV = min(minV, Sh[i].v);
                maxC = max(maxC, count[i]);
                minC = min(minC, count[i]);
        }
        cout << setiosflags(ios::fixed);
        cout << setprecision(3) << minV << " " << maxV << endl;
        cout << setprecision(0) << minC << " " << maxC << endl;
        return 0;
}
```

解法二：

解法一的问题：依次遍历不同的突触寻找入结点的相应脉冲源和神经元，耗时过长。
优化：使用图结构存储突触的信息。

时间复杂度$O(T(N+S))$,空间复杂度$O(DN)$,期望得分100.

3637672	张晨阳	张晨阳	脉冲神经网络	04-24 13:16	3.418KB	C++	正确	100	968ms	11.33MB

```
#include<bits/stdc++.h>                                    language-none
using namespace std;

const int MAX_NUM=2010;
const double INF=1e8;
int N,S,P,T;
double  t1;
double I_k[MAX_NUM/2][MAX_NUM/2];
```

```
struct node
{
   double v,u,a,b,c,d;
   int count_put;
}Node[MAX_NUM/2];

struct tuchu
{
        int t;
        int D;
        double w;
        tuchu(int _t, double _w, int _D)
        {
                t = _t;
                w = _w;
                D = _D;
        }
};
vector<tuchu> G[MAX_NUM];

static unsigned long nnext = 1;
// RAND_MAX assumed to be 32767
int myrand(void) {
    nnext = nnext * 1103515245 + 12345;
    return((unsigned)(nnext/65536) % 32768);
}

int main()
{
    std::ios::sync_with_stdio(false);
    cin>>N>>S>>P>>T;
    cin>>t1;

    int temp_N=N;
    int N_tmp=0;
    while(temp_N>0) {
        int RN;
        cin>>RN;
        temp_N-=RN;
        int RN_tmp=RN;
        double v,u,a,b,c,d;
        cin>>v>>u>>a>>b>>c>>d;
        while(RN_tmp>0) {
            Node[N_tmp].v=v;
            Node[N_tmp].u=u;
            Node[N_tmp].a=a;
            Node[N_tmp].b=b;
            Node[N_tmp].c=c;
            Node[N_tmp].d=d;
```

```
            N_tmp++;
            RN_tmp--;
        }
    }

    int maic[MAX_NUM];
    int P_tmp=0;
    int temp_P=P;
    while(temp_P>0) {
        int r;
        cin>>r;
        maic[P_tmp+N]=r;
        P_tmp++;
        temp_P--;
    }

    int S_tmp=0;
    int temp_S=S;
    int mod=0;
    while(temp_S>0) {
        int s,t,D;
        double w;
        cin>>s>>t>>w>>D;
        G[s].push_back(tuchu(t,w,D));//存储图
        mod=max(mod,D+1);
        temp_S--;
        S_tmp++;
    }
    //输入完毕
    for(int i=0;i<T;i++) {
        int t_tmp=i % mod;
        for (int j=N;j<P+N;j++)
            if(maic[j]>myrand())
                for(int k=0;k<G[j].size();k++) {//遍历脉冲源j连接神经元
                    int node_get=G[j][k].t;//脉冲源j的连接的神经元
                    double w_tmp=G[j][k].w;
                    int D_tmp=G[j][k].D;
                    I_k[(t_tmp+D_tmp)%mod][node_get]+=w_tmp;
                }

        for (int j=0;j<N;j++) {
            double v_tmp=Node[j].v;
            double u_tmp=Node[j].u;
            double a_tmp=Node[j].a;
            double b_tmp=Node[j].b;
            double c_tmp=Node[j].c;
            double d_tmp=Node[j].d;
            Node[j].v=v_tmp+t1*(0.04*v_tmp*v_tmp+5*v_tmp+140-
u_tmp)+I_k[t_tmp][j];
            Node[j].u=u_tmp+t1*a_tmp*(b_tmp*v_tmp-u_tmp);
```

```
            if(Node[j].v>=30) {
                Node[j].v=c_tmp;
                Node[j].u+=d_tmp;
                Node[j].count_put++;
                for(int k=0;k<G[j].size();k++) {//遍历脉冲源j连接神经元
                    int node_get=G[j][k].t;//脉冲源j的连接的神经元
                    double w_tmp=G[j][k].w;
                    int D_tmp=G[j][k].D;
                    I_k[(t_tmp+D_tmp)%mod][node_get]+=w_tmp;
                }
            }
        }
        memset(I_k[t_tmp],0, sizeof I_k[t_tmp]);
    }

    double max_v=-INF;
    double min_v=INF;
    int max_count=-INF;
    int min_count=INF;
    for (int i=0;i<N;i++) {
        max_v=max(max_v,Node[i].v);
        min_v=min(min_v,Node[i].v);
        max_count=max(max_count,Node[i].count_put);
        min_count=min(min_count,Node[i].count_put);
    }
    cout<<setiosflags(ios::fixed)<<setprecision(3)<<min_v<<" "
<<max_v<<endl;
    cout<<min_count<<" "<<max_count<<endl;
    return 0;
}
```

题目 4：收集卡片

题目描述：

n 种卡牌，抽到 i 的概率为 pi。如果已经抽过 i 了，转换为一枚硬币。每 k 个硬币可购买一张还未抽过的卡牌。
小林去抽卡，问在才去最优策略的情况下，期望需要抽的卡牌数量。绝对误差 ≤10^{-4}。
n ≤ 16，1 ≤ k ≤ 5。

题解：

动态规划、状压。

解法一：

fi,j 表示状态为 i (二进制，1表示抽到，0没有抽到)，已经抽了 j 次的概率。
边界条件$f0,0$=1。循环枚举。

时间复杂度$O(2^n n^2 k)$，空间复杂度为 $O(2^n nk)$，期望得分100。

3637115	张晨阳	张晨阳	收集卡牌	04-23 23:12	735B	C++	正确	100	203ms	53.18MB

```language-none
#include<iostream>
#include<cstdio>
using namespace std;

const int maxn = (1 << 17) + 17;//防越界
double dp[maxn][100];
double p[100];
int cnt[maxn];

int main() {
        double ans = 0;
        int n, k, x;
        cin >> n >> k;
        for (int i = 1; i <= n; i++)
                cin >> p[i];
        for (int i = 1; i <= (1 << n); i++) {
                x = i;
                while (x) x &= x - 1, cnt[i]++;
        }
        dp[0][0] = 1;
        for (int i = 0; i <= (1 << n); i++) {          //i表示当前的集合状态
                for (int j = 0; j <= 100; j++) {           //j表示卡牌数量
                        if (cnt[i] + (j - cnt[i]) / k == n) { //已获得完整卡牌
                                ans += j * dp[i][j];
                                continue;
                        }
                        for (int k = 1; k <= n; k++) {
                                if (i & (1 << (k - 1)))
                                        dp[i][j + 1] += dp[i][j] * p[k];
                                else
                                        dp[i + (1 << (k - 1))][j + 1] += dp[i][j] * p[k];//将该卡牌加入集合
                        }
                }
        }
        printf("%.10lf", ans);
        return 0;
}
```

题目 5：箱根山岳险天下

题目描述：

有一个集训队，每个人有排名，和权值。
有 m 个事件，每个事件有 4 种类型：

- 将排名最后的一个人踢出集训队，但可一直随队训练。
- 在集训队中加入一个人，排名在末尾。
- 将在第 s 天排名为 $[l, r]$ 的人的权值乘 y。
- 询问当前在第 s 天排名为 $[l, r]$ 的人的权值和，对 p 取模。

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

题解：

解法一：

时间复杂度$O(mlog^2m)$，空间复杂度$O(m)$，期望得分：40。

3637195	张晨阳	张晨阳	箱根山岳险天下	04-23 23:51	1.981KB	C++	内存超限	40	1.078s	859.6MB

```language-none
#include<iostream>
#include<vector>
using namespace std;
#define ll long long

const int MAX = 3e5 + 10;
int m, p, t;
int a = 0;
int ty[MAX];
int x[MAX], y[MAX];
ll qiang[MAX];  //每个队员的强度
vector<int> id; //正式队员名单
int id1;        //当前加入队员的id;
vector<int> st[MAX];//第i天被哪一天查询
vector<int> ls[MAX];
vector<int> rs[MAX];//被查询的区间范围
vector<int> res[MAX];

int main()
{
    std::ios::sync_with_stdio(false);
    cin >> m >> p >> t;
    for (int i = 1; i <= m; i++) {
        cin >> ty[i];
        ll s, l, r;
        if (ty[i] == 1)
            cin >> x[i];
        else {
            cin >> s >> l >> r;
```

```
            //记录某一天S被之后的某一天T查询
            st[s].push_back(i);
            ls[s].push_back(l);
            rs[s].push_back(r);
            if (ty[i] == 2)
                cin >> y[i];
        }
    }
    for (int i = 1; i <= m; i++) {
        if (ty[i] == 1) {
            if (t == 1)
                x[i] = x[i] ^ a;
            if (x[i] == 0)
                id.pop_back();
            else {
                qiang[id1] = x[i];
                id.push_back(id1);
                id1++;
            }
        }
        else if (ty[i] == 2) {
            if (t == 1)
                y[i] = y[i] ^ a;
            for (int j = 0; j < res[i].size(); j++) {
                int v = res[i][j];
                qiang[v] = (qiang[v] * y[i]) % p;
            }
        }
        else if (ty[i] == 3) {
            int sum = 0;
            for (int j = 0; j < res[i].size(); j++) {
                int v = res[i][j];
                sum = (sum + qiang[v]) % p;
            }
            cout << sum << endl;
            a = sum;
        }
        for (int j = 0; j < st[i].size(); j++) {
            int v = st[i][j];
            for (int k = ls[i][j] - 1; k <= rs[i][j] - 1; k++)
                res[v].push_back(id[k]);
        }
    }
    return 0;
}
```

解法二：

参考解法：动态+树链剖分。

以排名为树的高度，以时间为序依次加点的树。对于删点就等于在时间上退到父亲，然后之后就在父亲的基础下再加点。
如果可以离线那么树剖维护，but在线那么改成LCT即可。期望得分100。

3637720	张晨阳	张晨阳	箱根山岳险天下	04-24 14:05	3.228KB	CPP14	正确	100	1.093s	23.46MB

```language-none
#include <bits/stdc++.h>
using namespace std;

int MOD;
int ad(int x, int y)
{
    x += y;
    return x >= MOD ? x - MOD : x;
}
int mu(int x, int y) { return 1ll * x * y % MOD; }
const int maxn = 3e5 + 5;

int ch[maxn][2], fa[maxn], fa_tru[maxn];
int sm[maxn], tot, A[maxn], rev[maxn], dep[maxn], lz[maxn];
vector<pair<int, int>> BC[maxn];
static bool isroot(int p) { return ch[fa[p]][0] != p && ch[fa[p]][1] !=
p; }
void putup(int p) { sm[p] = ad(ad(sm[ch[p][0]], sm[ch[p][1]]), A[p]); }
void rotate(int x)
{
    int y = fa[x];
    int z = fa[y];
    fa[x] = z;
    if (!isroot(y)) {
        ch[z][ch[z][1] == y] = x;
    }
    int k = (ch[y][1] == x);
    ch[y][k] = ch[x][k ^ 1];
    fa[ch[y][k]] = y;
    fa[y] = x;
    ch[x][k ^ 1] = y;
    putup(y);
    putup(x);
}
void pushdown(int p)
{
    if (rev[p]) {
        swap(ch[p][0], ch[p][1]);
        rev[ch[p][0]] ^= 1, rev[ch[p][1]] ^= 1;
        rev[p] = 0;
    }
    if (lz[p] != 1) {
        A[ch[p][0]] = mu(A[ch[p][0]], lz[p]);
        A[ch[p][1]] = mu(A[ch[p][1]], lz[p]);
```

```
        sm[ch[p][0]] = mu(sm[ch[p][0]], lz[p]);
        sm[ch[p][1]] = mu(sm[ch[p][1]], lz[p]);
        lz[ch[p][0]] = mu(lz[ch[p][0]], lz[p]);
        lz[ch[p][1]] = mu(lz[ch[p][1]], lz[p]);
        lz[p] = 1;
    }
}
void splay(int x)
{
    int y, z;
    static int sta[maxn], top;
    top = 0;
    sta[++top] = x;
    for (int i = x; !isroot(i); i = fa[i]) {
        sta[++top] = fa[i];
    }
    while (top)
        pushdown(sta[top--]);
    while (!isroot(x)) {
        y = fa[x];
        z = fa[y];
        if (!isroot(y))
            (ch[z][0] == y) ^ (ch[y][0] == x) ? rotate(x) : rotate(y);
        rotate(x);
    }
}
void acc(int x)
{
    for (int y = 0; x; y = x, x = fa[x]) {
        splay(x);
        ch[x][1] = y;
        putup(x);
    }
}
void setroot(int x)
{
    acc(x);
    splay(x);
    rev[x] ^= 1;
}
void link(int x, int y)
{
    setroot(x);
    fa[x] = y;
}
int getp(int tim, int rank)
{
    auto it = --upper_bound(BC[rank].begin(), BC[rank].end(),
                            make_pair(tim, 0x3f3f3f3f));
    return it->second;
```

```
}
int M, T, NOW, LA;

int main()
{
    scanf("%d%d%d", &M, &MOD, &T);
    int OPT, S, L, R, Y;
    for (int tim = 1; tim <= M; tim++) {
        scanf("%d%d", &OPT, &S);
        if (OPT == 1) {
            if (T)
                S = (S ^ LA);
            if (S == 0) {
                NOW = fa_tru[NOW];
            } else {
                ++tot;
                lz[tot] = 1;
                fa_tru[tot] = fa[tot] = NOW;
                dep[tot] = dep[NOW] + 1;
                BC[dep[tot]].push_back(make_pair(tim, tot));
                A[tot] = sm[tot] = S;
                NOW = tot;
            }
        } else if (OPT == 2) {
            scanf("%d%d%d", &L, &R, &Y);
            if (T)
                Y = Y ^ LA;
            L = getp(S, L); // S 天 第L名
            R = getp(S, R);
            setroot(L);
            acc(R);
            splay(R);
            A[R] = mu(A[R], Y);
            sm[R] = mu(sm[R], Y);
            lz[R] = mu(lz[R], Y);
        } else {
            scanf("%d%d", &L, &R);
            L = getp(S, L);
            R = getp(S, R);
            setroot(L), acc(R), splay(R);
            LA = sm[R];
            printf("%d\n", LA);
        }
    }
    return 0;
}
```