2019 Multi-University Training Contest 3

比赛情况

题号	A	B	C	D	E	F	G	H	I	J	K
状态	-	-	-	O	-	O	O	Ø	O	-	Ø

O 在比赛中通过 Ø 赛后通过 ! 尝试了但是失败了 - 没有尝试

比赛时间

2020-07-15 13:00-18:00

题解

D - Distribution of books

题意：将一个数字序列的前x数字划分为k组，保证这k组数字的组和最大值最小。

最大值最小很自然的就能想到二分答案，于是考虑如何在知道最大数字组和的情况下检查是否能够分为满足情况的k组。考虑dp□设f_i为对于到第i个数字最多能被分为满足答案的几组，那么dp方程就是$f_i = max(f_j)+1, sum_i - ans \leq sum_j$,于是可以按照sum为下标维护一个线段树，线段树的值是sum对应的dp值最大值，然后每次查一个后缀最大值就行了，好像树状数组也行。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int inf = 1<<30;
const ll INF = 1ll<<60;
const int N = 2e5+5;
int n,k,dp[N],tr[N<<2],a[N],cnt;
ll id[N],sum[N];
void build(int p,int l,int r) {
    tr[p] = -1;
    if (l==r)return;
    int mid = (l+r)>>1;
    build(p<<1,l,mid);
    build(p<<1|1,mid+1,r);
}
void _update(int p,int l,int r,int a,int b) {
    if (l==r) {
        tr[p] = max(tr[p],b);
        return ;
    }
    int mid = (l+r)>>1;
    if (a<=mid)_update(p<<1,l,mid,a,b);
```

```
    else _update(p<<1|1,mid+1,r,a,b);
    tr[p] = max(tr[p<<1],tr[p<<1|1]);
}
int gtaaans(int p,int l,int r,int a,int b) {
    if (l>=a&&r<=b) return tr[p];
    int ans = -1;
    int mid = (l+r)>>1;
    if (a<=mid)ans = max(ans,gtaaans(p<<1,l,mid,a,b));
    if (b >mid)ans = max(ans,gtaaans(p<<1|1,mid+1,r,a,b));
    return ans;
}
int lower(ll x) {
    return lower_bound(id+1,id+n+1,x)-id;
}
int upper(ll x) {
    return upper_bound(id+1,id+n+1,x)-id;
}
bool check(ll ans) {
    build(1,1,cnt);
    bool flag = false;
    _update(1,1,cnt,lower(0),0);
    for (int i = 1;i<= n;i++)
    {
        dp[i] = 0;
        ll temp = sum[i]-ans;
        int pos = lower(temp);
        if (pos>cnt)continue;
        int dpj = gtaaans(1,1,cnt,pos,cnt);
        if (dpj == -1)continue;
        else dp[i] = max(dp[i],dpj+1);
        _update(1,1,cnt,lower(sum[i]),dp[i]);
        if (dp[i]>=k) {flag = true;break;}
    }
    return flag;
}
int main()
{
    int cas;
    scanf("%d",&cas);
    while (cas--) {
        scanf("%d%d",&n,&k);
        for (int i = 1;i<= n;i++)
        {
            scanf("%d",&a[i]);
            sum[i] = sum[i-1]+a[i];
            id[i] = sum[i];
        }
        id[n+1] = 0;
        sort(id+1,id+n+2);
        cnt = unique(id+1,id+n+2)-id-1;
```

```
        ll L = -2e14-5,R = 2e14+5,mid;
        while (L<R) {
            mid = (L+R)>>1;
            if (check(mid))
                R = mid;
            else L = mid+1;
        }
        printf("%lld\n",R);
    }
}
```

F - Fansblog

给一个大质数 $1e9 \le P \le 1e14$，求小于它的最大质数 Q 的阶乘 $Q!$ 模 P 的值。

猜 Q 和 P 之间差距不会太大，从 $P-1$ 往下枚举，用米勒拉宾素性测试。然后求阶乘的时候考虑威尔逊定理（学到了，$(P-1)! \equiv -1 \mod P$，注意要用快速乘^^_

```
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define TIMES 30
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

ll gcd(ll a,ll b) {return b==0?a:gcd(b,a%b);}
ll qmul(ll a,ll b,ll M) {a%=M;b%=M;ll
s=0;while(b){if(b&1)s=(s+a)%M;a=(a+a)%M;b>>=1;}return s; }
ll qpow(ll a,ll b,ll M) {a%=M;;ll
s=1;while(b){if(b&1)s=qmul(s,a,M);a=qmul(a,a,M);b>>=1;}return s; }
ll Rand(){
    static ll x=(srand((int)time(0)),rand());
    x+=1000003;
    if(x>1000000007)x-=1000000007;
```

```
    return x;
}
bool Witness(ll a,ll n) {
    ll u=n-1,t=0;
    while(u%2==0)t++,u>>=1;
    ll x=qpow(a,u,n);
    if(x==1)return false;
    for(int i=1; i<=t; i++,x=qmul(x,x,n))
        if(x!=n-1&&x!=1&&qmul(x,x,n)==1)return true;
    return x!=1;
}

bool Miller_Rabin(ll n) {
    if(n==2)return true;
    if(n<2||!(n&1))return false;
    for(int i=1; i<=TIMES; i++)
        if(Witness(Rand()%(n-1)+1,n))return false;
    return true;
}


int main()
{
    fastio();
    int _;
    for(cin>>_;_;_--){
        ll P; cin>>P;
        ll Q = P-1;
        while(1){
            if(Miller_Rabin(Q)) break;
            Q--;
        }
        ll ans = P-1;
        for(ll i=Q+1;i<=P-1;i++){
            ans = qmul(ans,qpow(i,P-2,P),P);
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

G - Find the answer

对于 1 到 n 每个位置 i 删除 $[1,i)$ 区间的几个元素能使得该位置的前缀和不大于 m

离散上权值线段树，查询最少多少个元素和大于等于 $sum-m$ 肯定是最大的若干个元素），相当于线段树上二分。

```cpp
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")
#include<bits/stdc++.h>
#define ll long long
#define pb push_back
using namespace std;
const int N=2e5+5;
ll w[N],num[N];
ll read()
{
    ll x=0,f=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
struct Node{int l,r;ll sz,sum;}t[N*4];
void build(int idx,int l,int r)
{
    t[idx].l=l,t[idx].r=r,t[idx].sz=t[idx].sum=0;
    if(l==r)return;
    int mid=(l+r)>>1;
    build(idx<<1,l,mid),build(idx<<1|1,mid+1,r);
}
void ins(int idx,int x)
{
    if(t[idx].l==t[idx].r){t[idx].sz++,t[idx].sum+=num[x];return;}
    int mid=(t[idx].l+t[idx].r)>>1;
    if(x<=mid)ins(idx<<1,x);else ins(idx<<1|1,x);
    t[idx].sz=t[idx<<1].sz+t[idx<<1|1].sz;
    t[idx].sum=t[idx<<1].sum+t[idx<<1|1].sum;
}
int que(int idx,ll x)
{
    if(x<=0)return 0;
    if(t[idx].l==t[idx].r)return x/num[t[idx].l]+(x%num[t[idx].l]>0);
    if(t[idx<<1|1].sum>=x)return que(idx<<1|1,x);
    return t[idx<<1|1].sz+que(idx<<1,x-t[idx<<1|1].sum);
}
int main()
{
    int q=read();
    while(q--)
    {
        int n,tot=0;
        ll m,sum=0;
        n=read(),m=read();
        for(int i=1;i<=n;i++)w[i]=read(),num[++tot]=w[i];
        sort(num+1,num+1+tot);
        tot=unique(num+1,num+1+tot)-num-1;
        build(1,1,tot);
```

```
        for(int i=1;i<=n;i++)
        {
            sum+=w[i];
            printf("%d ",que(1,sum-m));
            int x=lower_bound(num+1,num+1+tot,w[i])-num;
            ins(1,x);
        }
        puts("");
    }
    return 0;
}
```

H - Game

其实是在可以动态修改的情况下（交换 a_i,a_{i+1} □□回答若干 $[L,R]$ 有多少子区间异或和为零的问题。记录前缀异或和（相同则意味着区间异或和为零□□swap只改变 $pre[i]$ □带修改莫队一下，可能要卡常。

```
#include<bits/stdc++.h>
#define ll long long
#define pb push_back
using namespace std;
const int N=1e5+5;
const int M=2e7+10;
int n,m,swp[N],bloc,a[N],pre[N],qcnt,mcnt,inq[N],l,r;
ll res[N],num[M],ans;
ll read()
{
    ll x=0,f=1;char c=getchar();
    while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0'&&c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
struct Node
{
    int l,r,now,id;
    Node(int l=0,int r=0,int now=0,int id=0):l(l),r(r),now(now),id(id){}
}q[N];
bool cmp(Node x,Node y)
{
    if((x.l-1)/bloc==(y.l-1)/bloc)
    {
        if((x.r-1)/bloc==(y.r-1)/bloc)return
((x.r-1)/bloc&1)?x.now<y.now:x.now>y.now;
        return x.r<y.r;
    }
```

```
    return x.l<y.l;
}
void exchange(int x)
{
    swap(a[x],a[x+1]);
    if(x>=l-1&&x<=r)
    {
        ans-=num[pre[x]]*(num[pre[x]]-1)/2;
        num[pre[x]]--;
        ans+=num[pre[x]]*(num[pre[x]]-1)/2;
    }
    pre[x]^=a[x]^a[x+1];
    if(x>=l-1&&x<=r)
    {
        ans-=num[pre[x]]*(num[pre[x]]-1)/2;
        num[pre[x]]++;
        ans+=num[pre[x]]*(num[pre[x]]-1)/2;
    }
}
void work(int x)
{
    int f=1;
    if(inq[x])f=-1;
    if(x==l)
    {
        ans-=num[pre[x-1]]*(num[pre[x-1]]-1)/2;
        num[pre[x-1]]+=f;
        ans+=num[pre[x-1]]*(num[pre[x-1]]-1)/2;
    }
    if(x==r)
    {
        ans-=num[pre[x]]*(num[pre[x]]-1)/2;
        num[pre[x]]+=f;
        ans+=num[pre[x]]*(num[pre[x]]-1)/2;
    }
    inq[x]^=1;
}
int main()
{
    clock_t start1=clock(),end1;
    while(~scanf("%d%d",&n,&m))
    {
        qcnt=mcnt=ans=0,bloc=pow(n,0.6666666666)+1;
        for(int i=1;i<=n;i++)a[i]=read(),pre[i]=pre[i-1]^a[i];
        for(int i=1;i<=m;i++)
        {
            int op=read();
            if(op==1)
            {
                int l=read(),r=read();
                ++qcnt;
```

```
                q[qcnt]=Node(l,r,mcnt,qcnt);
            }
            else swp[++mcnt]=read();
        }
        sort(q+1,q+1+qcnt,cmp);
        l=1,r=0;
        for(int i=1;i<=qcnt;i++)
        {
            for(int j=q[i-1].now+1;j<=q[i].now;j++)exchange(swp[j]);
            for(int j=q[i-1].now;j>q[i].now;j--)exchange(swp[j]);
            while(l>q[i].l)--l,work(l);
            while(r<q[i].r)++r,work(r);
            while(l<q[i].l)work(l),l++;
            while(r>q[i].r)work(r),r--;
            ll len=q[i].r-q[i].l+1;
            res[q[i].id]=len*(len-1)/2+len-ans;
        }
        for(int i=1;i<=qcnt;i++)printf("%lld\n",res[i]);
        while(l<=r)work(l),l++;
    }
    return 0;
}
```

I - K Subsequence

题意：给定一个数字序列，求k个不下降子序列，使得这些子序列的和最大。

不下降子序列最大和好像是个比较经典的费用流模型，每个数字拆点，所有原点连这个数字前面的点，后面的点连汇点（流量1，费用0），两个点之间连流量为1，费用为$-a_i$的边，这个要找k个，那就在原点前面再连个超级原点，汇点后面来个超级汇点，流量为k即可。

```
#include <bits/stdc++.h>
using namespace std;
#define fi first
#define se second
typedef long long ll;
const int INF = 1<<30;
const int N = 4e3+20;
typedef pair<int, int> pii;
struct Edge {
    int to, cap, cost, rev;
    Edge() {}
    Edge(int to, int _cap, int _cost, int _rev):to(to), cap(_cap),
cost(_cost), rev(_rev) {}
};
int V;
```

```
int h[N];
int dis[N];
int prevv[N];
int pree[N];
vector<Edge> G[N];

void init(int n) {
    V = n;
    for(int i = 0; i <= V; ++i) G[i].clear();
}
void add(int x, int y, int cap, int cost){
    G[x].push_back(Edge(y, cap, cost, G[y].size()));
    G[y].push_back(Edge(x, 0, -cost, G[x].size() - 1));
}
int MCMF(int s, int t, int f, int &flow) {
    int res = 0;
    for(int i = 0; i < 1 + V; i++) h[i] = 0;
    while(f){
        priority_queue<pii, vector<pii>, greater<pii> > q;
        for(int i = 0; i < 1 + V; i++) dis[i] = INF;
        dis[s] = 0; q.push(pii(0, s));
        while(!q.empty()) {
            pii now = q.top(); q.pop();
            int v = now.second;
            if(dis[v] < now.first) continue;
            for(int i = 0; i < G[v].size(); ++i) {
                Edge &e = G[v][i];
                if(e.cap > 0 && dis[e.to] > dis[v] + e.cost + h[v] -
h[e.to]){
                    dis[e.to] = dis[v] + e.cost + h[v] - h[e.to];
                    prevv[e.to] = v;
                    pree[e.to] = i;
                    q.push(pii(dis[e.to], e.to));
                }
            }
        }
        if(dis[t] == INF) break;
        for(int i = 0; i <= V; ++i) h[i] += dis[i];
        int d = f;
        for(int v = t; v != s; v = prevv[v]) d = min(d,
G[prevv[v]][pree[v]].cap);
        f -= d; flow += d; res += d * h[t];
        for(int v = t; v != s; v = prevv[v]) {
            Edge &e = G[prevv[v]][pree[v]];
            e.cap -= d;
            G[v][e.rev].cap += d;
        }
    }
    return res;
}
int num[N];
```

```
int main() {
    int cas;
    scanf("%d",&cas);
    while (cas--)
    {
        int n,k;
        scanf("%d%d",&n,&k);
        int s = 2*n+1,t = 2*n+2;
        int S = s+2,T = t+2;
        init(n*2+4);
        for (int i = 1;i<= n;i++)
        {
            scanf("%d",&num[i]);
            add(s,i,1,0);
            add(i+n,t,1,0);
            add(i,i+n,1,-num[i]);
        }
        for (int i = 1;i<= n;i++)
            for (int j = i+1;j<= n;j++)
                if (num[j] >= num[i])
                    add(i+n,j,1,0);
        add(S,s,k,0);add(t,T,k,0);
        int flow = k;
        printf("%d\n", -MCMF(S,T,INF,flow));
    }
    return 0;
}
```

K - Squrirrel

在一颗带权树上割掉一条边，求到其他点最远距离最小的点和这个最远距离。

容易想到换根 dp 得到每个节点 u 沿着子树的前三远的距离和沿着父亲的最远距离。然后用这些距离求割掉一条边后的最远距离。

```
#include<bits/stdc++.h>
#define MEM(x) memset((x),0,sizeof(x))
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define ull unsigned long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
```

```
#define per(i,a,b) for(int i=(a);i>=(b);i--)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

int n,head[maxn],tot;
int
idx1[maxn],idx2[maxn],idx3[maxn],fi[maxn][2],se[maxn][2],th[maxn][2],fa[max
n][2],d[maxn];
struct edge
{
    int u,v,w,nxt;
}es[maxn<<1];
void addedge(int u,int v,int w)
{
    es[++tot] = (edge){u,v,w,head[u]};
    head[u] = tot;
}
void dfs1(int u,int f)
{
    for(int i=head[u];~i;i=es[i].nxt){
        int v = es[i].v,w = es[i].w;
        if(v==f) continue;
        dfs1(v,u);
        d[v] = w;
        if(fi[v][0] + w > fi[u][0]){
            th[u][0] = se[u][0];
            idx3[u] = idx2[u];
            se[u][0] = fi[u][0];
            idx2[u] = idx1[u];
            fi[u][0] = fi[v][0] + w;
            idx1[u] = v;
        }else if(fi[v][0] + w > se[u][0]){
            th[u][0] = se[u][0];
            idx3[u] = idx2[u];
            se[u][0] = fi[v][0] + w;
            idx2[u] = v;
        }else if(fi[v][0] + w > th[u][0]){
            th[u][0] = fi[v][0] + w;
            idx3[u] = v;
        }
    }
    fi[u][1] = min(fi[idx1[u]][0], max(fi[idx1[u]][1],se[idx1[u]][0]) +
d[idx1[u]]);
    se[u][1] = min(fi[idx2[u]][0], max(fi[idx2[u]][1],se[idx2[u]][0]) +
d[idx2[u]]);
}
```

```
void dfs2(int u,int f)
{
    for(int i=head[u];~i;i=es[i].nxt){
        int v = es[i].v,w = es[i].w;
        if(v==f) continue;
        if(idx1[u]==v){
            fa[v][0] = max(fa[u][0],se[u][0]) + w;
            fa[v][1] = min(max(fa[u][0],se[u][0]),
min(max(fa[u][1],se[u][0]), max(fa[u][0],max(se[u][1],th[u][0]))) + w);
        }else{
            fa[v][0] = max(fa[u][0],fi[u][0]) + w;
            if(idx2[u]==v){
                fa[v][1] = min(max(fa[u][0],fi[u][0]),
min(max(fa[u][1],fi[u][0]), max(fa[u][0],max(fi[u][1],th[u][0]))) + w);
            }else{
                fa[v][1] = min(max(fa[u][0],fi[u][0]),
min(max(fa[u][1],fi[u][0]), max(fa[u][0],max(fi[u][1],se[u][0]))) + w);
            }
        }
        dfs2(v,u);
    }
}
int main()
{
    fastio();
    int _,u,v,w;
    for(cin>>_;_;_--){
        tot=0;
        cin>>n;
        MEM(fi);MEM(se);MEM(th);MEM(fa);
        memset(head,-1,sizeof(head));
        rep(i,1,n-1){
            cin>>u>>v>>w;
            addedge(u,v,w);
            addedge(v,u,w);
        }
        dfs1(1,0);
        dfs2(1,0);
        int ans = inf,id;
        rep(i,1,n){
            int res =
min(max(fa[i][1],fi[i][0]),max(fa[i][0],max(fi[i][1],se[i][0])));
            if(res<ans) ans=res,id=i;
        }
        cout<<id<<" "<<ans<<endl;
    }
    return 0;
}
```

比赛总结与反思