数字

考虑一个经典的数位 dp：先差分成 $f(1,r)-f(1,l-1)$，然后 $dp(i,a,0/1)$ 表示当前在第 i 高位，前面的部分 $t\equiv a \mod x$，前面的位是否完全和 r 相同。这个时间复杂度 $O(dx)$，其中 $d=10$。注意如果有前导零，原数可以不包含 0，需要增设状态 $0/1$ 表示这个数到此为止是否有除了前导零之外的部分。

注意到有一个简单的暴力，在 $[l,r]$ 枚举 x 的倍数，然后 $O(\log_d V)$ 判断。发现这两个东西平衡一下就有 $O(dx+\frac{V}{x}\log_d V)\approx O(10\sqrt{V})$。

```
#include<bits/stdc++.h>
using namespace std;
#define MOD         998244353
#define speMOD      2933256077ll
#define int         long long
#define pii         pair<int,int>
#define all(v)      v.begin(),v.end()
#define pb          push_back
#define REP(i,b,e)  for(int i=(b);i<(int)(e);++i)
#define over(x)     {cout<<(x)<<endl;return;}
#define lowbit(x)   ((x)&(-(x)))
#define cntbit(x)   __builtin_popcount(x)
#define deal(v)     sort(all(v));v.erase(unique(v.begin(),v.end()),v.end())
#define lbound(v,x) lower_bound(all(v),x)-v.begin()
mt19937 sd(random_device{}());
int s,x;
int dp[100005][2][2],f[100005][2][2];
int a[20];
int solve(int r){
    REP(i,0,x){
        REP(j,0,2)REP(k,0,2)dp[i][j][k]=f[i][j][k]=0;
    }
    f[0][1][0]=1;
    int n=0;while(r)a[n++]=r%10,r/=10;
    while(n--){
        REP(i,0,x){
            REP(j,0,2)if(f[i][j][0]||f[i][j][1]){
                REP(p,0,10)if((s>>p)&1){
                    if(p<=a[n]||j==0){
                        dp[(i*10+p)%x][j&&p==a[n]][1]+=f[i][j][1];
                        if(p)dp[(i*10+p)%x][j&&p==a[n]][1]+=f[i][j][0];
                    }
                }
                dp[i*10%x][(a[n]==0)&&j][0]+=f[i][j][0];
            }
        }
        REP(i,0,x){
            REP(j,0,2)REP(k,0,2)f[i][j][k]=dp[i][j][k],dp[i][j][k]=0;
        }
    }
    return f[0][0][1]+f[0][1][1];
}
void Main() {
    int l,r;string t;
```

```
    cin>>x>>l>>r>>t;
    for(auto i:t)s|=(1<<(i-48));
    if(x<=100000){
        cout<<solve(r)-solve(l-1)<<endl;
    }else{
        int ans=0;
        REP(i,(l+x-1)/x,r/x+1)if(i*x>=l&&i*x<=r){
            bool f=1;int t=i*x;
            while(t)if(!((s>>(t%10))&1))f=0,t=0;else t/=10;
            if(f)++ans;
        }
        over(ans)
    }
}
void TC() {
    int tc=1;
    while(tc--){
        Main();
        cout.flush();
    }
}
signed main() {
    freopen("num.in","r",stdin);
    freopen("num.out","w",stdout);
    return cin.tie(0),cout.tie(0),ios::sync_with_stdio(0),TC(),0;
}
/*
1. CLEAR the arrays (ESPECIALLY multitests)
2. DELETE useless output
 */
```

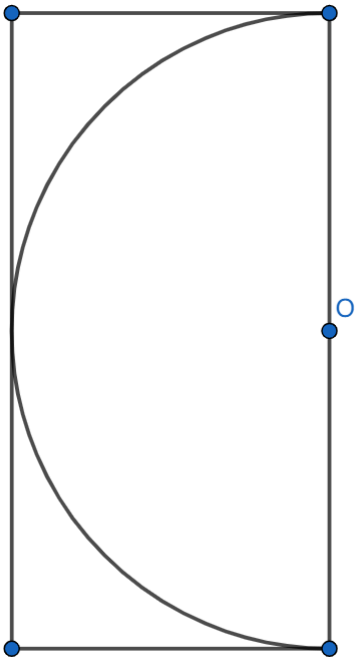
连接

容易猜测当集合的 size 足够大的时候必然存在子集和是 k 的倍数。构造 $k-1$ 个 1 可以使 $size=k-1$，然后打个表猜测最多就是 $k-1$。

> 一个简单的证明：
>
> 考虑一个集合所有非空子集，按子集的大小排序。若两个不同大小的子集有相同的子集和，则必然存在两个相互包含的不同大小子集有相同的子集和（加上对方多余的部分），则作差得到 0。设所有子集和相等的子集的大小都相等，则最多有 $k-1$ 个不同的子集和，即 $size\le k-1$。

确定性

对于这题显然先二分答案，考虑建图。如果一个连通块的大小至少是 k，则已经存在合法了。于是有用的连边数不超过 nk。考虑类似平面最近点对的做法。我们按横坐标排序，用 `set` 维护【横坐标和当前点差距不超过 mid 的点集】，按纵坐标排序。然后对于一个点的连边，找出纵坐标范围在 $[y-mid,y+mid]$ 之间的所有点，然后依次检验连边。考虑复杂度：要卡满复杂度，则任意连通块的大小不超过 $k-1$。如下图，设当前点为 O 点，我们取了左侧 $mid\times 2mid$ 大小的矩形内的点，距离当前点正确的范围是一个半圆，剩下的点最多组成两个连通块（上下两个弓形），于是点数是 $O(k)$ 的。



建图之后用 bitset 或压位优化背包即可。时间复杂度 $O(\log V(n\log n+nk))$。

```
#include<bits/stdc++.h>
using namespace std;
#define MOD         998244353
#define speMOD      293325607711
#define int         long long
#define pii         pair<int,int>
#define all(v)      v.begin(),v.end()
#define pb          push_back
#define REP(i,b,e)  for(int i=(b);i<(int)(e);++i)
#define over(x)     {cout<<(x)<<endl;return;}
#define lowbit(x)   ((x)&(-(x)))
#define cntbit(x)   __builtin_popcount(x)
#define deal(v)     sort(all(v));v.erase(unique(v.begin(),v.end()),v.end())
#define lbound(v,x) lower_bound(all(v),x)-v.begin()
struct point{
    int x,y,val;
    bool operator <(point a){return x==a.x? y<a.y:x<a.x;}
}a[50005];
int n,k;
int fa[50005],sz[50005];
vector<int>v[50005];
int findfa(int x){return fa[x]==x? x:fa[x]=findfa(fa[x]);}
int merge(int x,int y){
    x=findfa(x);y=findfa(y);
    if(x==y)return sz[x];
    if(sz[x]<sz[y])swap(x,y);
    sz[x]+=sz[y];fa[y]=x;return sz[x];
}
#define sq(x) ((x)*(x))
int dist(int x,int y){return sq(a[x].x-a[y].x)+sq(a[x].y-a[y].y);}
bool check(int d){
    set<pii>s;
    int x=-1;
    iota(fa,fa+n,0);
```

```
    REP(i,0,n)sz[i]=1;
    REP(i,0,n){
        while(x+1<i&&sq(a[i].x-a[x+1].x)>d)++x,s.erase(s.find({a[x].y,x}));
        auto it=s.lower_bound({a[i].y-ceil(sqrt(d))-1,-1});
        while(it!=s.end()&&sq((it->first)-a[i].y)<=d){
            if(dist(it->second,i)<=d){
                if(merge(i,it->second)>=k)return 1;
            }
            ++it;
        }
        s.insert({a[i].y,i});
    }
    REP(i,0,n)v[i].clear();
    REP(i,0,n)v[findfa(i)].pb(a[i].val);
    REP(i,0,n)if(fa[i]==i){
        bitset<30>b;b.set(0);
        for(auto j:v[i]){
            if(b[k-j])return 1;
            b|=(b<<j)|(b>>(k-j));
        }
    }
    return 0;
}
void Main() {
    cin>>n>>k;
    REP(i,0,n)cin>>a[i].x>>a[i].y>>a[i].val,a[i].val%=k;
    sort(a,a+n);
    int l=0,r=2e16,res=0;
    while(l<=r){
        int mid=(l+r)>>1;
        if(check(mid))res=mid,r=mid-1;
        else l=mid+1;
    }
    cout<<fixed<<setprecision(3)<<sqrt(res)<<endl;
}
void TC() {
    int tc=1;
    while(tc--){
        Main();
        cout.flush();
    }
}
signed main() {
    freopen("connect.in","r",stdin);
    freopen("connect.out","w",stdout);
    return cin.tie(0),cout.tie(0),ios::sync_with_stdio(0),TC(),0;
}
/*
1. CLEAR the arrays (ESPECIALLY multitests)
2. DELETE useless output
 */
```

随机化

考虑人类智慧，随机偏移+随机旋转，然后按横坐标排序后，每个点往前暴力找横坐标的差不超过 mid 的所有点尝试连边。一个重要的剪枝是，如果当前连通块以及存在 k 的倍数的子集和就要 `return 1`，否则对于大部分数据直接退化成 $O(n^2)$。这个剪枝直接写 $sz \geq k$ 也能通过原题数据。

```
#include<bits/stdc++.h>
using namespace std;
#define MOD         998244353
#define speMOD      2933256077ll
#define int         long long
#define pii         pair<int,int>
#define all(v)      v.begin(),v.end()
#define pb          push_back
#define REP(i,b,e)  for(int i=(b);i<(int)(e);++i)
#define over(x)     {cout<<(x)<<endl;return;}
#define lowbit(x)   ((x)&(-(x)))
#define cntbit(x)   __builtin_popcount(x)
#define deal(v)     sort(all(v));v.erase(unique(v.begin(),v.end()),v.end())
#define lbound(v,x) lower_bound(all(v),x)-v.begin()
mt19937 sd(random_device{}());
struct point{
    double x,y;
    int val;
    bool operator <(point a){return x==a.x? y<a.y:x<a.x;}
    void move(double X,double Y){
        double xx=x,yy=y;
        x=(xx*X-yy*Y);y=xx*Y+X*yy;
    }
}a[50005];
int n,k;
int fa[50005],sz[50005];
vector<int>v[50005];
int findfa(int x){return fa[x]==x? x:fa[x]=findfa(fa[x]);}
int merge(int x,int y){
    x=findfa(x);y=findfa(y);
    if(x==y)return sz[x];
    sz[x]+=sz[y];fa[y]=x;return sz[x];
}
#define sq(x) ((x)*(x))
double dist(int x,int y){return sq(a[x].x-a[y].x)+sq(a[x].y-a[y].y);}
bool check(double d){
    set<pii>s;
    iota(fa,fa+n,0);
    REP(i,0,n)sz[i]=1;
    REP(i,0,n){
        for(int j=i-1;j>=0;--j)if(dist(i,j)<=d+0.5)merge(i,j);
        else if(sq(a[i].x-a[j].x)>d+0.5)break;
        if(sz[findfa(i)]>=k)return 1;
    }
    REP(i,0,n)v[i].clear();
    REP(i,0,n)v[findfa(i)].pb(a[i].val);
    REP(i,0,n)if(fa[i]==i){
        bitset<30>b;b.set(0);
        for(auto j:v[i]){
```

```
            if(b[k-j])return 1;
            b|=(b<<j)|(b>>(k-j));
        }
    }
    return 0;
}
void Main() {
    cin>>n>>k;
    REP(i,0,n)cin>>a[i].x>>a[i].y>>a[i].val,a[i].val%=k;
    uniform_real_distribution<double>rd(0,acos(-1)*2),delta(-1e8,0);
    double X=delta(sd),Y=delta(sd);
    REP(i,0,n)a[i].x+=X,a[i].y+=Y;
    double th=rd(sd);X=cos(th),Y=sin(th);
    REP(i,0,n)a[i].move(X,Y);
    sort(a,a+n);
    int l=0,r=2e16,res=0;
    while(l<=r){
        int mid=(l+r)>>1;
        if(check(mid))res=mid,r=mid-1;
        else l=mid+1;
    }
    cout<<fixed<<setprecision(3)<<sqrt(res)<<endl;
}
void TC() {
    int tc=1;
    while(tc--){
        Main();
        cout.flush();
    }
}
signed main() {
    freopen("connect.in","r",stdin);
    freopen("connect.out","w",stdout);
    return cin.tie(0),cout.tie(0),ios::sync_with_stdio(0),TC(),0;
}
/*
1. CLEAR the arrays (ESPECIALLY multitests)
2. DELETE useless output
 */
```

关键点

先做子树。设状态 $f(i,j=0/1,k=0/1)$ 表示子树 i 内，i 点是否选了，i 的儿子选了 k 个的答案。转移手动枚举当前这个点是否选即可。

考虑环的部分，先断环成链，然后钦定链首尾的两个点是否选择，然后对着上面的东西做一样的 dp。具体地，转移枚举上一个点是否选，上上个点是否选，当前点是否选，当前点的儿子选了几个。最后把四种钦定的情况的答案取最优的即可。

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

```
#define MOD         998244353
#define speMOD      2933256077ll
#define int         long long
#define pii         pair<int,int>
#define all(v)      v.begin(),v.end()
#define pb          push_back
#define REP(i,b,e)  for(int i=(b);i<(int)(e);++i)
#define over(x)     {cout<<(x)<<endl;return;}
#define lowbit(x)   ((x)&(-(x)))
#define cntbit(x)   __builtin_popcount(x)
#define deal(v)     sort(all(v));v.erase(unique(v.begin(),v.end()),v.end())
#define lbound(v,x) lower_bound(all(v),x)-v.begin()
int n;
int dp[100005][2][2];
vector<int>v[100005];
int deg[100005];
int g[100005][2][2];
void Main() {
    cin>>n;
    REP(i,0,n){
        int x,y;
        cin>>x>>y;
        --x,--y;
        v[x].pb(y);v[y].pb(x);
    }
    REP(i,0,n)deg[i]=v[i].size();
    queue<int>q;
    REP(i,0,n)if(deg[i]==1)q.push(i);
    while(!q.empty()){
        int x=q.front();q.pop();
        vector<int>t;deg[x]=0;
        for(auto i:v[x])if(!deg[i])t.pb(i);
        else{
            --deg[i];
            if(deg[i]==1)q.push(i);
        }
        int mn=n,m2=n;
        for(auto i:t)dp[x][0][0]+=dp[i][0][1],dp[x][1][0]+=dp[i][0][0];
        for(auto i:t)mn=min(mn,dp[i][1][0]-dp[i][0][0]),m2=min(m2,dp[i][1][1]-
dp[i][0][1]);
        dp[x][1][1]=dp[x][1][0]+mn;
        dp[x][0][1]=dp[x][0][0]+mn;
        ++dp[x][1][1];++dp[x][1][0];
    }
    vector<int>t;
    int f=-1;
    REP(i,0,n)if(deg[i]>1)f=i;
    t.pb(f);int l=f;
    for(auto i:v[l])if(deg[i]>1){f=i;break;}
    while(f!=t[0]){
        t.pb(f);
        for(auto i:v[f])if(deg[i]>1&&l!=i){
            l=f;f=i;break;
        }
    }
    for(auto x:t){
```

```
        vector<int>p;
        for(auto i:v[x])if(deg[i]==0)p.pb(i);
        int mn=n+1,m2=n+1;
        for(auto i:p)dp[x][0][0]+=dp[i][0][1],dp[x][1][0]+=dp[i][0][0];
        for(auto i:p)mn=min(mn,dp[i][1][0]-dp[i][0][0]),m2=min(m2,dp[i][1][1]-
dp[i][0][1]);
        dp[x][1][1]=dp[x][1][0]+mn;
        dp[x][0][1]=dp[x][0][0]+m2;
        ++dp[x][1][1];++dp[x][1][0];
    }
    int ans=n+1,m=t.size();
    REP(_,0,2){
        REP(_2,0,2){
            REP(j,0,4)g[0][j/2][j%2]=n+1;
            REP(p,0,2)if(p+_2<=1)g[0][_][p+_2]=min(g[0][_][p+_2],dp[t[0]][_][p]);
            REP(i,1,m){
                REP(j,0,4)g[i][j/2][j%2]=n+1;
                REP(j,0,2){
                    REP(k,0,2)if(g[i-1][k][j]<=n){
                        REP(l,0,2){
                            REP(p,0,2)if(j+l==1){
                                g[i][l][k+p]=min(g[i][l][k+p],g[i-1][k]
[j]+dp[t[i]][l][p]);
                            }
                        }
                    }
                }
            }
            ans=min(ans,g[m-1][_2][1-_]);
//          cout<<ans<<endl;
        }
    }
    over(ans>=n+1? -1:ans)
}
void TC() {
    int tc=1;
    while(tc--){
        Main();
        cout.flush();
    }
}
signed main() {
    freopen("key.in","r",stdin);
    freopen("key.out","w",stdout);
    return cin.tie(0),cout.tie(0),ios::sync_with_stdio(0),TC(),0;
}
/*
1. CLEAR the arrays (ESPECIALLY multitests)
2. DELETE useless output
 */
```

披萨

注意到代价是 $\sum L_i$ 减去所有披萨完成的时间和。贪心地按 T 排序，答案就是 $\sum L_i - \sum_{i=1}^{n}(n-i+1)T_i$。

朴素地考虑用平衡树维护这个东西。$\sum(n-i)T_i$ 的东西是可合并的，维护两边的这个式子，每一块的点数以及权值和即可。操作是删除一个点，插入一个新点。fhq treap 即可，时间复杂度 $O(n\log n)$，跑起来比较卡（这题为啥要卡常）。

```
#include<bits/stdc++.h>
using namespace std;
#define MOD         998244353
#define speMOD      293325607711
#define int         long long
#define pii         pair<int,int>
#define all(v)      v.begin(),v.end()
#define pb          push_back
#define REP(i,b,e)  for(int i=(b);i<(int)(e);++i)
#define over(x)     {cout<<(x)<<endl;return;}
#define lowbit(x)   ((x)&(-(x)))
#define cntbit(x)   __builtin_popcount(x)
#define deal(v)     sort(all(v));v.erase(unique(v.begin(),v.end()),v.end())
#define lbound(v,x) lower_bound(all(v),x)-v.begin()
mt19937 sd(random_device{}());
int read(){
    int res=0;char c=getchar();
    while(c<48||c>57)c=getchar();
    do res=(res<<1)+(res<<3)+(c^48),c=getchar();while(c>=48&&c<=57);
    return res;
}
#define N 200005
uniform_int_distribution<int>rd(1,1ll<<40);
int n,q;
int a[N],b[N];
int ls[N],rs[N],sz[N],sum[N],tr[N],val[N],pr[N];
int rt=-1;
void pushup(int p){
    sz[p]=sz[ls[p]]+sz[rs[p]]+1;
    sum[p]=sum[ls[p]]+val[p]+sum[rs[p]];
    tr[p]=tr[ls[p]]+(sz[rs[p]]+1)*sum[ls[p]]+val[p]*(sz[rs[p]]+1)+tr[rs[p]];
}
pii split(int x,int p){
    if(!p)return {p,p};
    if(val[p]<=x){
        auto [y,z]=split(x,rs[p]);
        rs[p]=y;pushup(p);
        return {p,z};
    }else{
        auto [y,z]=split(x,ls[p]);
        ls[p]=z;pushup(p);
        return {y,p};
    }
}
int merge(int x,int y){
    if(!x)return y;
```

```
    if(!y)return x;
    if(pr[x]<pr[y]){
        rs[x]=merge(rs[x],y);
        pushup(x);
    }else{
        ls[y]=merge(x,ls[y]);
        pushup(y);
        x=y;
    }
    return x;
}
pii dellst(int p){
    if(rs[p]){
        auto [x,y]=dellst(rs[p]);
        rs[p]=x;pushup(p);
        return {p,y};
    }
    return {ls[p],p};
}
void addnode(int p){
    if(rt==-1)return rt=p,void();
    auto [x,y]=split(val[p],rt);
    rt=merge(x,merge(p,y));
}
void Main() {
    n=read();q=read();
    REP(i,0,n)b[i]=read(),a[i]=read();
    int ans=0;
    REP(i,0,n)ans+=b[i];
    REP(i,0,n){
        int p=i+1;pr[p]=rd(sd);
        sz[p]=1;val[p]=a[i];sum[p]=a[i];tr[p]=a[i];
        addnode(p);
    }
    cout<<ans-tr[rt]<<"\n";
    while(q--){
        int x=read()-1,y=read(),z=read();
        ans-=b[x];ans+=b[x]=y;
        auto [X,Y]=split(a[x],rt);
        auto [Z,p]=dellst(X);
        rt=merge(Z,Y);
        a[x]=z;
        pr[p]=rd(sd);sz[p]=1;val[p]=tr[p]=sum[p]=a[x];ls[p]=rs[p]=0;
        addnode(p);
        cout<<ans-tr[rt]<<"\n";
    }
}
void TC() {
    int tc=1;
    while(tc--){
        Main();
        cout.flush();
    }
    cerr<<clock()<<endl;
}
signed main() {
```

```
    freopen("pizza.in","r",stdin);
    freopen("pizza.out","w",stdout);
    return cin.tie(0),cout.tie(0),ios::sync_with_stdio(0),TC(),0;
}
/*
1. CLEAR the arrays (ESPECIALLY multitests)
2. DELETE useless output
 */
```

考虑如何卡常。观察到值域不大，把平衡树上的东西压到值域线段树上就好了，信息同样可合并，需要支持单点修改，全局查询。时间复杂度 $O(n \log n)$，跑得很快。

```
#include<bits/stdc++.h>
using namespace std;
#define MOD         998244353
#define speMOD      293325607711
#define pii         pair<int,int>
#define all(v)      v.begin(),v.end()
#define pb          push_back
#define REP(i,b,e)  for(int i=(b);i<(int)(e);++i)
#define over(x)     {cout<<(x)<<endl;return;}
#define lowbit(x)   ((x)&(-(x)))
#define cntbit(x)   __builtin_popcount(x)
#define deal(v)     sort(all(v));v.erase(unique(v.begin(),v.end()),v.end())
#define lbound(v,x) lower_bound(all(v),x)-v.begin()
int read(){
    int res=0;char c=getchar();
    while(c<48||c>57)c=getchar();
    do res=(res<<1)+(res<<3)+(c^48),c=getchar();while(c>=48&&c<=57);
    return res;
}
struct node{
    int cnt;long long sum,ans;
    node(int x=0,long long y=0,long long z=0){cnt=x;sum=y;ans=z;}
    node operator +(node a){
        return {cnt+a.cnt,sum+a.sum,sum*a.cnt+ans+a.ans};
    }
}seg[400005];
void update(int pos,int l,int r,int p,int val){
    if(l==r){
        seg[p].cnt+=val;
        seg[p].sum+=pos*val;
        seg[p].ans=1ll*seg[p].cnt*(seg[p].cnt+1)/2*pos;
        return;
    }
    int m=(l+r)>>1;
    if(m>=pos)update(pos,l,m,p*2+1,val);
    else update(pos,m+1,r,p*2+2,val);
    seg[p]=seg[p*2+1]+seg[p*2+2];
}
int n,q,m=100000;
int a[200005],b[200005];
void Main() {
    n=read(),q=read();
    long long ans=0;
```

```
    REP(i,0,n)ans+=b[i]=read(),a[i]=read();
    REP(i,0,n)update(a[i],1,m,0,1);
    cout<<ans-seg[0].ans<<"\n";
    while(q--){
        int x=read()-1,y=read(),z=read();
        ans-=b[x];ans+=b[x]=y;
        if(a[x]!=z)update(a[x],1,m,0,-1),update(a[x]=z,1,m,0,1);
        cout<<ans-seg[0].ans<<"\n";
    }
}
void TC() {
    int tc=1;
    while(tc--){
        Main();
        cout.flush();
    }
}
signed main() {
    freopen("pizza.in","r",stdin);
    freopen("pizza.out","w",stdout);
    return cin.tie(0),cout.tie(0),ios::sync_with_stdio(0),TC(),0;
}
/*
1. CLEAR the arrays (ESPECIALLY multitests)
2. DELETE useless output
 */
```