ICPC Templates

苏同

April 15, 2021

Contents

1 配置文件

1.1 VIM 配置文件

```
colorscheme evening

set ts=4
set sw=4
set smarttab
set smartindent
set number
set autowrite
set hlsearch
set mouse=a
set showmatch

:inoremap ' ''<ESC>i
:inoremap " ""<ESC>i
:inoremap ( ()<ESC>i
:inoremap [ []<ESC>i
:inoremap { {<CR>}<ESC>O
```

2 总结反思

2.1 总结反思

(算法与小技巧在模板里，以下为常犯的错误)

* 抽象问题，提取模型，再回归问题验证

* 读题要仔细仔细再仔细，一定不能想当然

* 多手造数据模拟题意，多推式子，多找性质，多猜结论

* 通过样例找做法，检查是否读错题

* 考虑特殊情况：寻找切入点，从特殊到一般

* 考虑特殊数据：验证程序正确性，完善性

* 卡题了多冷静，先放松一下，再重新读题，重新换角度思考。可以尝试把思考的每一步写下来，从头到尾检验每一步的是否正确，是否有更巧妙的做法或结论

* 如果觉得复杂度始终无法接受，一般可能有很强的性质和结论

* 不要一有很复杂很多数据结构的思路就写，要想好再写，先找更简便的做法

* 写慢点，注意数组访问是否越界，数组大小是否足够，是否满足题目时间空间限制，每个语句的正确性

* 学会逆向思考：插入-> 删除；建反边

* 多用二分

* 巧用随机化算法

* 对于最难的一类需要分类讨论的 DP 题目，一定要想明白 DP 式子的含义，考虑好每一种特殊情况（!！惨烈的教训：好多次树上 DP 没想明白就开始写，本来 60 分钟的题做了 180 分钟!!）

* 对于最难的一类需要分类讨论的题目，可能通常的做题方法是构造算法-> 找反例-> 补全特判-> 找反例->...，通常来说十分耗费时间和心态，所以要谨慎用这种模式做题

3 数据结构

3.1 线段树

3.1.1 可持久化并查集

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+1;
const int M = 25;

int tot,ls[N*M],rs[N*M],w[N*M],d[N*M],rt[N];
int n,m;

void insert(int &x,int y,int l,int r,int p,int v){
    x=++tot;
    ls[x]=ls[y];rs[x]=rs[y];
    if(l==r){
        w[x]=v;
        return;
    }
    int mid=l+r>>1;
    if(p<=mid)insert(ls[x],ls[y],l,mid,p,v);
    else insert(rs[x],rs[y],mid+1,r,p,v);
}

void init(int &x,int l,int r){
    x=++tot;
    if(l==r){
        w[x]=l;d[x]=1;
        return;
    }
    int mid=l+r>>1;
    init(ls[x],l,mid);
    init(rs[x],mid+1,r);
}

void add(int x,int l,int r,int p){
    if(l==r)return void(d[x]++);
    int mid=l+r>>1;
    if(p<=mid)add(ls[x],l,mid,p);
    else add(rs[x],mid+1,r,p);
}


int gf(int x,int l,int r,int p){
    if(l==r)return w[x];
```

```
43      int mid=l+r>>1;
44      if(p<=mid)return gf(ls[x],l,mid,p);
45      else return gf(rs[x],mid+1,r,p);
46  }
47
48  int find(int r,int x){
49      int p=gf(r,1,n,x);
50      if(x==p)return x;
51      return find(r,p);
52  }
53
54  int main(){
55      scanf("%d%d",&n,&m);
56      init(rt[0],1,n);
57      for(int i=1,o,x,y;i<=m;++i){
58          scanf("%d%d",&o,&x);
59          if(o==1){
60              scanf("%d",&y);
61              rt[i]=rt[i-1];
62              int fx=find(rt[i],x),fy=find(rt[i],y);
63              if(fx==fy)continue;
64              if(d[fx]<d[fy])swap(fx,fy);
65              insert(rt[i],rt[i-1],1,n,fy,fx);
66              if(d[fx]==d[fy])d[fx]++;
67          }
68          if(o==2)rt[i]=rt[x];
69          if(o==3){
70              scanf("%d",&y);
71              rt[i]=rt[i-1];
72              int fx=find(rt[i],x),fy=find(rt[i],y);
73              if(fx==fy)printf("1\n");
74              else printf("0\n");
75          }
76      }
77  }
```

3.1.2 李超线段树

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   typedef double db;
5   const int N = 2e5+1;
6
7   int n,cnt;
8   int ls[N*2],rs[N*2];
9   struct line{
10      db k,b;
11      int id;
12      line(db _k=0,db _b=0,int _id=0){
13          k=_k;b=_b;id=_id;
14      }
```

```
}w[N*2];

line newline(db x0,db y0,db x1,db y1,int id){
    static line ret;
    if(x0==x1)ret.k=0,ret.b=max(y0,y1);
    else ret.k=(y1-y0)/(x1-x0),ret.b=y1-x1*ret.k;
    ret.id=id;
    return ret;
}

db get(line a,db x0){
    return a.k*x0+a.b;
}

db inter(line a,line b){
    return (a.b-b.b)/(b.k-a.k);
}

bool cmp(line a,line b,db x){
    if(!a.id)return 1;//注意a.id为0表示此处没有线段，所以所有插入线段的id都不应为0
    if(get(a,x)==get(b,x))return a.id>b.id;
    return get(a,x)<get(b,x);
}

int build(int l,int r){
    static int tot=0;
    int x=++tot;
    if(l==r)return x;
    int mid=l+r>>1;
    ls[x]=build(l,mid);
    rs[x]=build(mid+1,r);
    return x;
}

void push(int x,int l,int r,line a){
    if(!w[x].id)w[x]=a;
    if(cmp(w[x],a,l))swap(w[x],a);
    if(l==r||a.k==w[x].k)return;
    db p=inter(w[x],a);
    if(p<l||p>r)return;
    int mid=l+r>>1;
    if(p<=mid)push(ls[x],l,mid,w[x]),w[x]=a;
    else push(rs[x],mid+1,r,a);
    return;
}

void change(int x,int l,int r,int L,int R,line a){
    if(L<=l&&r<=R)return void(push(x,l,r,a));
    int mid=l+r>>1;
    if(L<=mid)change(ls[x],l,mid,L,R,a);
    if(R>mid)change(rs[x],mid+1,r,L,R,a);
    return;
}
```

```
68
69  line solve(int x,int l,int r,int x0){
70      if(l==r)return w[x];
71      int mid=l+r>>1;line ret;
72      if(x0<=mid)ret=solve(ls[x],l,mid,x0);
73      else ret=solve(rs[x],mid+1,r,x0);
74      if(cmp(ret,w[x],x0))return w[x];
75      return ret;
76  }
77
78  int main(){
79      scanf("%d",&n);
80      build(1,40000);
81      for(int i=1,o,x,x0,y0,x1,y1,ans=0;i<=n;++i){
82          scanf("%d",&o);
83          if(!o)scanf("%d",&x),printf("%d\n",ans=solve(1,1,40000,x).id);
84          else{
85              scanf("%d%d%d%d",&x0,&y0,&x1,&y1);
86              if(x0>x1)swap(x0,x1),swap(y0,y1);
87              change(1,1,40000,x0,x1,newline(x0,y0,x1,y1,++cnt));
88          }
89      }
90  }
```

3.1.3 线段树合并

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   const int N = 1e5+1;
5   const int lg = 21;
6
7   int ls[N*lg],rs[N*lg],w[N*lg],tot,c[N],q[N],rt[N],ans[N],n;
8   vector<int>g[N];
9
10  void up(int x){
11      w[x]=w[ls[x]]+w[rs[x]];
12  }
13
14  void insert(int &x,int l,int r,int p,int d){
15      if(!x)x=++tot;
16      w[x]+=d;
17      int mid=l+r>>1;
18      if(l==r)return;
19      if(p<=mid)insert(ls[x],l,mid,p,d);
20      else insert(rs[x],mid+1,r,p,d);
21  }
22
23  int merge(int x,int y){
24      if(!x||!y)return x^y;
25      ls[x]=merge(ls[x],ls[y]);
26      rs[x]=merge(rs[x],rs[y]);
```

```
    up(x);
    return x;
}

int sum(int x,int l,int r,int L,int R){
    if(L<=l&&r<=R)return w[x];
    int mid=l+r>>1,ret=0;
    if(L<=mid)ret+=sum(ls[x],l,mid,L,R);
    if(R>mid)ret+=sum(rs[x],mid+1,r,L,R);
    return ret;
}

void dfs(int x){
    int i;
    insert(rt[x],1,n,c[x],1);
    for(i=0;i<g[x].size();++i)dfs(g[x][i]);
    for(i=0;i<g[x].size();++i)rt[x]=merge(rt[x],rt[g[x][i]]);
    ans[x]=sum(rt[x],1,n,c[x]+1,n);
}

int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i)scanf("%d",&c[i]),q[i]=c[i];
    sort(q+1,q+n+1);
    for(int i=1;i<=n;++i)c[i]=lower_bound(q+1,q+n+1,c[i])-q;
    for(int i=2,f;i<=n;++i)scanf("%d",&f),g[f].push_back(i);
    dfs(1);
    for(int i=1;i<=n;++i)printf("%d\n",ans[i]);
}
```

3.1.4 查询删除区间内所有子区间

区间由左右端点表示，对于每个左端点在线段树上维护右端点最远已经删到第 pos 个位置，那么查询并删除区间内子区间个数的操作变成了将 $pos[l...r]$ 分别与 r 取 max。注意到 pos 序列单调不降，所以取 max 后会发生改变的 pos 一定是连续的区间 $[l..R]$，在线段树上二分第一个 pos 值小于 r 的位置即可。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5+1;

int Case;
int n,a[N];
ll ans;

struct ST{
    int tot;
    int ls[N<<1],rs[N<<1],mn[N<<1],tag[N<<1];
    ll sum[N<<1];
    void up(int x){
        mn[x]=mn[ls[x]];
```

```
16      sum[x]=sum[ls[x]]+sum[rs[x]];
17  }
18  void down(int x,int l,int r){
19      if(!tag[x])return;
20      int mid=l+r>>1;
21      if(ls[x]){
22          tag[ls[x]]=tag[x];
23          mn[ls[x]]=tag[x];
24          sum[ls[x]]=1ll*tag[x]*(mid-l+1);
25      }
26      if(rs[x]){
27          tag[rs[x]]=tag[x];
28          mn[rs[x]]=tag[x];
29          sum[rs[x]]=1ll*tag[x]*(r-mid);
30      }
31      tag[x]=0;
32  }
33  int build(int l,int r){
34      int x=++tot;
35      if(l==r){
36          mn[x]=sum[x]=l-1;
37          return x;
38      }
39      int mid=l+r>>1;
40      ls[x]=build(l,mid);
41      rs[x]=build(mid+1,r);
42      up(x);
43      return x;
44  }
45  void change(int x,int l,int r,int L,int R,int d){
46      if(L<=l&&r<=R){
47          mn[x]=d;
48          sum[x]=1ll*d*(r-l+1);
49          tag[x]=d;
50          return;
51      }
52      down(x,l,r);
53      int mid=l+r>>1;
54      if(L<=mid)change(ls[x],l,mid,L,R,d);
55      if(R>mid)change(rs[x],mid+1,r,L,R,d);
56      up(x);
57  }
58  int getlast(int x,int l,int r,int d){
59      if(l==r)return l;
60      down(x,l,r);
61      int mid=l+r>>1;
62      if(mn[rs[x]]<d)return getlast(rs[x],mid+1,r,d);
63      return getlast(ls[x],l,mid,d);
64  }
65  void init(){
66      mn[0]=1e9;
67      for(int i=1;i<=tot;++i){
68          ls[i]=rs[i]=0;
```

```
            mn[i]=sum[i]=tag[i]=0;
        }
        tot=0;
        int root=build(1,n);
    }
}T;

ll Solve(int l,int r){
    if(l>r||T.mn[1]>=r)return 0;
    int p=T.getlast(1,1,n,r);
    if(p<l)return 0;
    ll ret=T.sum[1];
    T.change(1,1,n,l,p,r);
    return T.sum[1]-ret;
}

int main(){
    scanf("%d",&Case);
    while(Case--){
        scanf("%d",&n);
        memset(a,0,sizeof(a));
        for(int i=1,x;i<=n;++i){
            scanf("%d",&x);
            a[x]=i;
        }
        T.init();
        ans=0;
        for(int i=N-1;i;--i){
            vector<int>g;
            for(int j=1;i*j<N;++j)
            if(a[i*j]){
                g.push_back(a[i*j]);
            }
            sort(g.begin(),g.end());
            int m=g.size();
            if(m<2)continue;
            ll tot=0;
            tot+=Solve(g[1]+1,n);
            tot+=Solve(1,g[m-2]-1);
            tot+=Solve(g[0]+1,g[m-1]-1);
            ans+=tot*i;
        }
        printf("%lld\n",ans);
    }
}
```

3.1.5 线段树优化建图

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e6+1;
```

```
5  const ll inf = 2e18;
6
7  int n,m,S,head[N*2],cnt,tot;
8  ll d[N*2];
9
10 struct nd{
11    int ne,to;
12    ll w;
13 }e[N*2];
14
15 void in(int x,int y,ll w){
16    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;e[cnt].w=w;
17 }
18
19 struct Segment_Tree{
20    int ls[N],rs[N],id[N],rt;
21
22    int build(bool tp,int l,int r){
23       int x=++tot;
24       if(l==r){
25          if(!tp)in(x,l,0);
26          else in(l,x,0);
27          return x;
28       }
29       int mid=l+r>>1;
30       ls[x]=build(tp,l,mid);
31       rs[x]=build(tp,mid+1,r);
32       if(!tp)in(x,ls[x],0),in(x,rs[x],0);
33       else in(ls[x],x,0),in(rs[x],x,0);
34       return x;
35    }
36    void change(int fr,int L,int R,ll d,bool tag,int x,int l,int r){
37       if(L<=l&&r<=R){
38          if(!tag)in(fr,x,d);
39          else in(x,fr,d);
40          return;
41       }
42       int mid=l+r>>1;
43       if(L<=mid)change(fr,L,R,d,tag,ls[x],l,mid);
44       if(R>mid)change(fr,L,R,d,tag,rs[x],mid+1,r);
45       return;
46    }
47    void init(bool tp){
48       rt=build(tp,1,n);
49    }
50 }t1,t2;
51
52 void spfa(int s){
53    static bool inq[N*2];
54    queue<int>q;q.push(s);
55    for(int i=1;i<=tot;++i)d[i]=inf;
56    d[s]=0;inq[s]=1;
57    while(!q.empty()){
```

```
58          int x=q.front();q.pop();inq[x]=0;
59          for(int i=head[x];i;i=e[i].ne){
60              int y=e[i].to;
61              if(d[y]>d[x]+e[i].w){
62                  d[y]=d[x]+e[i].w;
63                  if(!inq[y])q.push(y),inq[y]=1;
64              }
65          }
66      }
67  }
68
69  int main(){
70      scanf("%d%d%d",&n,&m,&S);
71      tot=n;
72      t1.init(1);
73      t2.init(0);
74      for(int i=1,tp,x,y,l,r,w;i<=m;++i){
75          scanf("%d",&tp);
76          if(tp==1){
77              scanf("%d%d%d",&x,&y,&w);
78              in(x,y,w);
79          }
80          if(tp==2){
81              scanf("%d%d%d%d",&x,&l,&r,&w);
82              t2.change(x,l,r,w,0,t2.rt,1,n);
83          }
84          if(tp==3){
85              scanf("%d%d%d%d",&y,&l,&r,&w);
86              t1.change(y,l,r,w,1,t1.rt,1,n);
87          }
88      }
89      spfa(S);
90      for(int i=1;i<=n;++i)printf("%lld ",(d[i]==inf?-1:d[i]));
91  }
```

3.1.6 吉司机线段树区间取 min

给定一个序列，要求实现区间取 min（即区间内 a_i 更新为 $min(a_i, val)$），区间求 max，区间求 sum。

做法：

维护区间最大值 mx，区间严格次大值 sx，区间最大值出现的次数 cx，然后进行分类讨论：

1、$val >= mx$，明显对区间无影响，退出；

2、$sx < val < mx$，此时 mx 会被更改成 val，而 sx, cx 则不变，对区间和 sum 的贡献为 $(mx - val) * cx$；（此处不能使 $sx = val$，否则在更新后 $mx = cx$，此时 cx 也需要改变）

3、$val <= sx$，跳到当前节点的两个子节点去处理这种情况再 $pushup$ 回来。

在以上算法中，区间取 min 操作实际上被转化为了对区间的最大值集体减小一个数的操作，且这次操作不会让最大值比严格次大值更小。

另外，我们发现 mx 本身就可以作为取 min 的标记，所以我们无需另外打标记。

时间复杂度为 $O(log^2n)$。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e6+5;

int Case;
int n,m;
int a[N];

struct STB{
   int tot;
   int ls[N<<1],rs[N<<1];
   int mx[N<<1];//区间最大值
   int sx[N<<1];//区间严格次大值
   int cx[N<<1];//区间最大值出现的次数
   ll sum[N<<1];

   void up(int x){
      sum[x]=sum[ls[x]]+sum[rs[x]];
      mx[x]=max(mx[ls[x]],mx[rs[x]]);
      sx[x]=max(sx[ls[x]],sx[rs[x]]);
      if(mx[ls[x]]!=mx[rs[x]])sx[x]=max(sx[x],min(mx[ls[x]],mx[rs[x]]));
      cx[x]=0;
      if(mx[x]==mx[ls[x]])cx[x]+=cx[ls[x]];
      if(mx[x]==mx[rs[x]])cx[x]+=cx[rs[x]];
   }

   int build(int l,int r){
      int x=++tot;
      if(l==r){
         mx[x]=sum[x]=a[l];
         sx[x]=-1e9;
         cx[x]=1;
         return x;
      }
      int mid=l+r>>1;
      ls[x]=build(l,mid);
      rs[x]=build(mid+1,r);
      up(x);
      return x;
   }

   void downmin(int x,int val){
      sum[x]-=1ll*(mx[x]-val)*cx[x];
      mx[x]=val;
   }

   void down(int x){
      if(mx[x]<mx[ls[x]])downmin(ls[x],mx[x]);
      if(mx[x]<mx[rs[x]])downmin(rs[x],mx[x]);
   }
```

```
52
53      void changemin(int x,int l,int r,int L,int R,int val){
54          if(val>=mx[x])return;
55          if(L<=l&&r<=R&&val>sx[x]){
56              downmin(x,val);
57              return;
58          }
59          down(x);
60          int mid=l+r>>1;
61          if(L<=mid)changemin(ls[x],l,mid,L,R,val);
62          if(R>mid)changemin(rs[x],mid+1,r,L,R,val);
63          up(x);
64      }
65
66      ll getsum(int x,int l,int r,int L,int R){
67          if(L<=l&&r<=R)return sum[x];
68          down(x);
69          int mid=l+r>>1;
70          ll sum=0;
71          if(L<=mid)sum+=getsum(ls[x],l,mid,L,R);
72          if(R>mid)sum+=getsum(rs[x],mid+1,r,L,R);
73          up(x);
74          return sum;
75      }
76
77      int getmax(int x,int l,int r,int L,int R){
78          if(L<=l&&r<=R)return mx[x];
79          down(x);
80          int mid=l+r>>1;
81          int mx=0;
82          if(L<=mid)mx=max(mx,getmax(ls[x],l,mid,L,R));
83          if(R>mid)mx=max(mx,getmax(rs[x],mid+1,r,L,R));
84          up(x);
85          return mx;
86      }
87
88      void clear(){
89          for(int i=1;i<=tot;++i){
90              ls[i]=rs[i]=0;
91              mx[i]=sx[i]=cx[i]=0;
92              sum[i]=0;
93          }
94          tot=0;
95      }
96  }T;
97
98  int main(){
99      scanf("%d",&Case);
100     while(Case--){
101         scanf("%d%d",&n,&m);
102         for(int i=1;i<=n;++i)scanf("%d",&a[i]);
103         T.clear();
104         T.build(1,n);
```

```
105         for(int i=1,o,x,y,t;i<=m;++i){
106             scanf("%d%d%d",&o,&x,&y);
107             if(o==0){
108                 scanf("%d",&t);
109                 T.changemin(1,1,n,x,y,t);
110             }
111             else if(o==1){
112                 printf("%d\n",T.getmax(1,1,n,x,y));
113             }
114             else{
115                 printf("%lld\n",T.getsum(1,1,n,x,y));
116             }
117         }
118     }
119 }
```

3.2 平衡树

3.2.1 SPLAY

```
1  #include<bits/stdc++.h>
2  const int N = 1e5+5;
3  using namespace std;
4
5  int n;
6  int tot,root;
7  int w[N],num[N],sz[N],fa[N],son[N][2];
8
9  void update(int x){
10     sz[x]=sz[son[x][0]]+sz[son[x][1]]+num[x];
11 }
12
13 //void pushdown(int x){
14 // do something...
15 //}
16
17 void rotate(int x){
18     //pushdown(fa[x]);pushdown(x);
19     int y=fa[x],z=fa[y],t=(son[y][0]==x);
20     fa[y]=x;fa[x]=z;
21     if(z) son[z][son[z][1]==y]=x;
22     son[y][!t]=son[x][t];fa[son[x][t]]=y;
23     son[x][t]=y;
24     update(y);update(x);
25 }
26
27 void splay(int x,int f){
28 // pushdown(x);
29     while(fa[x]!=f){
30         int y=fa[x],z=fa[y];
31         if(z!=f){
32             if(son[y][0]==x^son[z][0]==y)rotate(x);
```

```
33          else rotate(y);
34      }
35      rotate(x);
36   }
37   if(!f)root=x;
38}
39
40void insert(int &x,int val,int f){
41   if(!x){
42      x=++tot;fa[x]=f;
43      son[x][0]=son[x][1]=0;
44      w[x]=val;sz[x]=num[x]=1;
45      splay(x,0);
46      return;
47   }
48   if(val==w[x]){
49      sz[x]++;num[x]++;
50      splay(x,0);
51      return;
52   }
53   insert(son[x][val>w[x]],val,x);
54   update(x);
55}
56
57int get(int val){
58   int x=root;
59// pushdown(x);
60   while(x&&w[x]!=val){
61      x=son[x][val>w[x]];
62//  pushdown(x);
63   }
64   return x;
65}
66
67void delet(int w){
68   int x=get(w);
69   if(!x)return;
70   splay(x,0);
71   if(num[x]>1){
72      num[x]--;sz[x]--;
73      return;
74   }
75   if(!son[x][0]||!son[x][1])root=son[x][0]+son[x][1];
76   else{
77      int y=son[x][1];
78      while(son[y][0])y=son[y][0];
79      splay(y,x);
80      fa[son[x][0]]=y;
81      son[y][0]=son[x][0];
82      root=y;
83      son[x][0]=son[x][1]=0;
84   }
85   fa[root]=0;
```

```
86      update(root);
87  }
88
89  int getrank(int val){
90      int x=root,ret=0,last=0;
91      while(x){
92          last=x;
93          if(val>w[x]){
94              ret+=sz[son[x][0]]+num[x];
95              x=son[x][1];
96          }
97          else if(val==w[x]){
98              ret+=sz[son[x][0]];
99              break;
100         }
101         else{
102             x=son[x][0];
103         }
104     }
105     if(last)splay(last,0);
106     return ret+1;
107 }
108
109 int kth(int k){
110     int x=root;
111 // pushdown(x);
112     while(k<=sz[son[x][0]]||k>sz[son[x][0]]+num[x]){
113         if(k<=sz[son[x][0]])x=son[x][0];
114         else k-=sz[son[x][0]]+num[x],x=son[x][1];
115 //  pushdown(x);
116     }
117     return w[x];
118 }
119
120 int getpre(int val){
121     int x=root,ret=0,last=0;
122     while(x){
123         last=x;
124         if(val>w[x]){
125             ret=w[x];
126             x=son[x][1];
127         }
128         else if(val==w[x]){
129             if(son[x][0]){
130                 x=son[x][0];
131                 while(son[x][1])x=son[x][1];
132                 ret=w[x];
133             }
134             break;
135         }
136         else{
137             x=son[x][0];
138         }
```

```
    }
    if(last)splay(last,0);
    return ret;
}

int getne(int val){
    int x=root,ret=0,last=0;
    while(x){
        last=x;
        if(val<w[x]){
            ret=w[x];
            x=son[x][0];
        }
        else if(val==w[x]){
            if(son[x][1]){
                x=son[x][1];
                while(son[x][0])x=son[x][0];
                ret=w[x];
            }
            break;
        }
        else{
            x=son[x][1];
        }
    }
    if(last)splay(last,0);
    return ret;
}

int main(){
    scanf("%d",&n);
    for(int i=1,x,y;i<=n;++i){
        scanf("%d%d",&x,&y);
        if(x==1)insert(root,y,0);
        if(x==2)delet(y);
        if(x==3)printf("%d\n",getrank(y));
        if(x==4)printf("%d\n",kth(y));
        if(x==5)printf("%d\n",getpre(y));
        if(x==6)printf("%d\n",getne(y));
    }
}
```

3.2.2 线段树套 SPLAY 维护二维区间

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define pd(i) (i>='0'&&i<='9')
using namespace std;
const int N = 5e4+1;
const int inf = 0x7fffffff;
const int M = N*40;
```

```
9
10 int n,m,tot,a[N];
11 int son[M][2],fa[M],w[M],s[M],L[M],R[M],num[M],root[M];
12
13 int read(){
14    int x=0,f=1;char ch=getchar();
15    while(!pd(ch))ch=='-'?f=-1:0,ch=getchar();
16    while(pd(ch))x=(x<<3)+(x<<1)+ch-'0',ch=getchar();
17    return x*f;
18 }
19
20 void up(int x){
21    s[x]=s[son[x][0]]+s[son[x][1]]+num[x];
22 }
23
24 void rotate(int x){
25    int y=fa[x],z=fa[y],t=son[y][0]==x;
26    fa[y]=x;fa[x]=z;
27    if(z)son[z][son[z][1]==y]=x;
28    son[y][!t]=son[x][t];fa[son[x][t]]=y;
29    son[x][t]=y;
30    up(y);up(x);
31 }
32
33 void splay(int i,int x,int f){
34    while(fa[x]!=f){
35       int y=fa[x],z=fa[y];
36       if(z!=f){
37          if(son[y][0]==x^son[z][0]==y)rotate(x);
38          else rotate(y);
39       }
40       rotate(x);
41    }
42    if(!f)root[i]=x;
43 }
44
45 void insert(int i,int &x,int f,int val){
46    if(!x){
47       x=++tot;fa[x]=f;
48       son[x][0]=son[x][1]=0;
49       w[x]=val;s[x]=num[x]=1;
50       splay(i,x,0);
51       return;
52    }
53    if(w[x]==val){
54       s[x]++;num[x]++;
55       splay(i,x,0);
56       return;
57    }
58    insert(i,son[x][val>w[x]],x,val);
59    up(x);
60 }
61
```

```
62  int get(int i,int val){
63      int x=root[i];
64      while(x&&w[x]!=val)x=son[x][val>w[x]];
65      return x;
66  }
67
68  void delet(int i,int x){
69      x=get(i,x);splay(i,x,0);
70      if(num[x]>1){
71          num[x]--;s[x]--;
72          return;
73      }
74      if(son[x][0]*son[x][1]==0)root[i]=son[x][0]+son[x][1];
75      else{
76          int y=son[x][1];while(son[y][0])y=son[y][0];
77          splay(i,y,x);
78          fa[son[x][0]]=y;
79          son[y][0]=son[x][0];
80          root[i]=y;
81          son[x][0]=son[x][1]=0;
82      }
83      fa[root[i]]=0;
84      up(root[i]);
85  }
86
87  void build(int now,int l,int r,int x,int val){
88      L[now]=l;R[now]=r;
89      insert(now,root[now],0,val);
90      if(l==r)return;
91      int mid=l+r>>1;
92      if(x<=mid)build(now*2,l,mid,x,val);
93      else build(now*2+1,mid+1,r,x,val);
94      return;
95  }
96
97  int rk(int i,int val){
98      int x=root[i],ret=0;
99      while(x){
100         if(w[x]<val)ret+=s[son[x][0]]+num[x],x=son[x][1];
101         else x=son[x][0];
102     }
103     return ret;
104 }
105
106 int getrk(int now,int x,int y,int k){
107     int l=L[now],r=R[now];
108     if(x<=l&&r<=y)return rk(now,k);
109     int mid=l+r>>1;int ret=0;
110     if(x<=mid)ret+=getrk(now*2,x,y,k);
111     if(y>mid)ret+=getrk(now*2+1,x,y,k);
112     return ret;
113 }
114
```

```
115 int kth(int x,int y,int k){
116     int l=0,r=inf;
117     while(l!=r){
118         int mid=l+r+1>>1;
119         if(getrk(1,x,y,mid)+1<=k)l=mid;
120         else r=mid-1;
121     }
122     return l;
123 }
124 
125 void change(int now,int x,int val,int __val){
126     int l=L[now],r=R[now];
127     delet(now,__val);
128     insert(now,root[now],0,val);
129     if(l==r)return;
130     int mid=l+r>>1;
131     if(x<=mid)change(now*2,x,val,__val);
132     else change(now*2+1,x,val,__val);
133     return;
134 }
135 
136 int pre(int i,int val){
137     int x=root[i],ret=-inf;
138     while(x){
139         if(w[x]>=val)x=son[x][0];
140         else ret=max(ret,w[x]),x=son[x][1];
141     }
142     return ret;
143 }
144 
145 int ne(int i,int val){
146     int x=root[i],ret=inf;
147     while(x){
148         if(w[x]>val)ret=min(ret,w[x]),x=son[x][0];
149         else x=son[x][1];
150     }
151     return ret;
152 }
153 
154 int getpre(int now,int x,int y,int val){
155     int l=L[now],r=R[now];
156     if(x<=l&&r<=y)return pre(now,val);
157     int mid=l+r>>1,ret=-inf;
158     if(x<=mid)ret=max(ret,getpre(now*2,x,y,val));
159     if(y>mid)ret=max(ret,getpre(now*2+1,x,y,val));
160     return ret;
161 }
162 
163 int getne(int now,int x,int y,int val){
164     int l=L[now],r=R[now];
165     if(x<=l&&r<=y)return ne(now,val);
166     int mid=l+r>>1,ret=inf;
167     if(x<=mid)ret=min(ret,getne(now*2,x,y,val));
```

```
168     if(y>mid)ret=min(ret,getne(now*2+1,x,y,val));
169     return ret;
170 }
171 
172 int main(){
173     n=read();m=read();
174     for(int i=1;i<=n;++i){
175         a[i]=read();
176         build(1,1,n,i,a[i]);
177     }
178     for(int i=1,op,x,y,k,pos;i<=m;++i){
179         op=read();
180         switch(op){
181             case 1:x=read();y=read();k=read();printf("%d\n",getrk(1,x,y,k)+1);break;
182             case 2:x=read();y=read();k=read();printf("%d\n",kth(x,y,k));break;
183             case 3:pos=read();k=read();change(1,pos,k,a[pos]);a[pos]=k;break;
184             case 4:x=read();y=read();k=read();printf("%d\n",getpre(1,x,y,k));break;
185             case 5:x=read();y=read();k=read();printf("%d\n",getne(1,x,y,k));break;
186         }
187     }
188 }
```

3.2.3 SPLAY 维护序列

支持区间翻转，区间求和，区间求最大值（还可以支持区间整体移动等等，但是以下代码中没有包含）

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 1e5+5;
5  
6  int n,m;
7  int tot;
8  int c[N],cmx[N];
9  ll ctot[N];
10 int sz[N],fa[N],son[N][2],tag[N];
11 bool rev[N];
12 int rt;
13 
14 void Rever(int x){
15     swap(son[x][0],son[x][1]);
16     rev[x]^=1;
17 }
18 
19 void update(int x){
20     sz[x]=sz[son[x][0]]+sz[son[x][1]]+1;
21     ctot[x]=ctot[son[x][0]]+ctot[son[x][1]]+c[x];
22     cmx[x]=max(c[x],max(cmx[son[x][0]],cmx[son[x][1]]));
23 }
24 
25 void pushdown(int x){
```

```
26      if(tag[x]){
27          if(son[x][0]){
28              c[son[x][0]]+=tag[x];
29              cmx[son[x][0]]+=tag[x];
30              ctot[son[x][0]]+=tag[x]*sz[son[x][0]];
31              tag[son[x][0]]+=tag[x];
32          }
33          if(son[x][1]){
34              c[son[x][1]]+=tag[x];
35              cmx[son[x][1]]+=tag[x];
36              ctot[son[x][1]]+=tag[x]*sz[son[x][1]];
37              tag[son[x][1]]+=tag[x];
38          }
39          tag[x]=0;
40      }
41      if(rev[x]){
42          if(son[x][0])Rever(son[x][0]);
43          if(son[x][1])Rever(son[x][1]);
44          rev[x]=0;
45      }
46  }
47
48  void rotate(int x){
49      pushdown(fa[x]);pushdown(x);
50      int y=fa[x],z=fa[y],t=(son[y][0]==x);
51      fa[y]=x;fa[x]=z;
52      if(z)son[z][son[z][1]==y]=x;
53      son[y][!t]=son[x][t];fa[son[x][t]]=y;
54      son[x][t]=y;
55      update(y);update(x);
56  }
57
58  void splay(int x,int f){
59      pushdown(x);
60      while(fa[x]!=f){
61          int y=fa[x],z=fa[y];
62          if(z!=f){
63              if(son[y][0]==x^son[z][0]==y)rotate(x);
64              else rotate(y);
65          }
66          rotate(x);
67      }
68      if(!f)rt=x;
69  }
70
71  int build(int f,int l,int r){
72      int x=++tot;
73      int mid=l+r>>1;
74      if(l<mid)son[x][0]=build(x,l,mid-1);
75      if(mid<r)son[x][1]=build(x,mid+1,r);
76      fa[x]=f;
77      c[x]=ctot[x]=cmx[x]=0;
78      sz[x]=1;
```

```
79      update(x);
80      return x;
81  }
82
83  int get(int val){//the val+1 th position
84      ++val;
85      int x=rt;
86      pushdown(x);
87      while(val!=sz[son[x][0]]+1){
88          if(val>sz[son[x][0]]){
89              val-=sz[son[x][0]]+1;
90              x=son[x][1];
91          }
92          else{
93              x=son[x][0];
94          }
95          pushdown(x);
96      }
97      return x;
98  }
99
100 void add(int x,int y,int k){
101     x=get(x);y=get(y);
102     splay(x,0);
103     splay(y,x);
104     int t=son[y][0];
105     c[t]+=k;
106     ctot[t]+=k*sz[t];
107     cmx[t]+=k;
108     tag[t]+=k;
109     update(y);update(x);
110 }
111
112 void rever(int x,int y){
113     x=get(x);y=get(y);
114     splay(x,0);
115     splay(y,x);
116     int t=son[y][0];
117     Rever(t);
118 }
119
120 int getmx(int x,int y){
121     x=get(x);y=get(y);
122     splay(x,0);
123     splay(y,x);
124     int t=son[y][0];
125     return cmx[t];
126 }
127
128 ll gettot(int x,int y){
129     x=get(x);y=get(y);
130     splay(x,0);
131     splay(y,x);
```

```
    int t=son[y][0];
    return ctot[t];
}

int main(){
    cmx[0]=-2e9;
    scanf("%d%d",&n,&m);
    rt=build(0,0,n+1);
    for(int i=1,o,l,r,k;i<=m;++i){
        scanf("%d%d%d",&o,&l,&r);
        if(o==1){
            scanf("%d",&k);
            add(l-1,r+1,k);
        }
        else if(o==2){
            rever(l-1,r+1);
        }
        else{
            printf("%d\n",getmx(l-1,r+1));
        }
    }
}
```

3.2.4 SPLAY 维护序列终极版

1：插入 $INSERT\ \ pos_i\ tot\ c_1c_2\ c_{tot}$

在当前数列的第 pos_i 个数字后插入 tot 个数字：$c_1, c_2\ c_{tot}$；若在数列首插入，则 pos_i 为 0

2：删除 $DELETE\ \ pos_i\ tot$

从当前数列的第 pos_i 个数字开始连续删除 tot 个数字

3：修改 $MAKE-SAME\ \ pos_i\ tot\ c$

从当前数列的第 pos_i 个数字开始的连续 tot 个数字统一修改为 c

4：翻转 $REVERSE\ \ pos_i\ tot$

取出从当前数列的第 pos_i 个数字开始的 tot 个数字，翻转后放入原来的位置

5：求和 $GET-SUM\ \ pos_i\ tot$

计算从当前数列的第 pos_i 个数字开始的 tot 个数字的和并输出

6：求和最大的子列 $MAX-SUM$

求出当前数列中和最大的一段子列，并输出最大和

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>
using namespace std;
const int inf = 1000000000;
const int N = 1e6+1;
int read()
{
```

```
    int x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
int n,m,rt,cnt;
int a[N],id[N],fa[N],son[N][2];
int sum[N],sz[N],v[N],mx[N],lx[N],rx[N];
bool tag[N],rev[N];
queue<int>q;//recycle
void update(int x)
{
    int l=son[x][0],r=son[x][1];
    sum[x]=sum[l]+sum[r]+v[x];
    sz[x]=sz[l]+sz[r]+1;
    mx[x]=max(mx[l],mx[r]);
    mx[x]=max(mx[x],rx[l]+v[x]+lx[r]);
    lx[x]=max(lx[l],sum[l]+v[x]+lx[r]);
    rx[x]=max(rx[r],sum[r]+v[x]+rx[l]);
}
void pushdown(int x)
{
    int l=son[x][0],r=son[x][1];
    if(tag[x])
    {
        rev[x]=tag[x]=0;
        if(l)tag[l]=1,v[l]=v[x],sum[l]=v[x]*sz[l];
        if(r)tag[r]=1,v[r]=v[x],sum[r]=v[x]*sz[r];
        if(v[x]>=0)
        {
            if(l)lx[l]=rx[l]=mx[l]=sum[l];
            if(r)lx[r]=rx[r]=mx[r]=sum[r];
        }
        else
        {
            if(l)lx[l]=rx[l]=0,mx[l]=v[x];
            if(r)lx[r]=rx[r]=0,mx[r]=v[x];
        }
    }
    if(rev[x])
    {
        rev[x]^=1;rev[l]^=1;rev[r]^=1;
        swap(lx[l],rx[l]);swap(lx[r],rx[r]);
        swap(son[l][0],son[l][1]);swap(son[r][0],son[r][1]);
    }
}
void rotate(int x,int &k)
{
    int y=fa[x],z=fa[y],l,r;
    l=(son[y][1]==x);r=l^1;
    if(y==k)k=x;
    else son[z][son[z][1]==y]=x;
    fa[son[x][r]]=y;fa[y]=x;fa[x]=z;
```

```
64      son[y][l]=son[x][r];son[x][r]=y;
65      update(y);update(x);
66  }
67  void splay(int x,int &k)
68  {
69      while(x!=k)
70      {
71          int y=fa[x],z=fa[y];
72          if(y!=k)
73          {
74              if(son[y][0]==x^son[z][0]==y)rotate(x,k);
75              else rotate(y,k);
76          }
77          rotate(x,k);
78      }
79  }
80  int get(int val)
81  {
82      int x=rt;
83      pushdown(x);
84      while(val!=sz[son[x][0]]+1){
85          if(val>sz[son[x][0]]){
86              val-=sz[son[x][0]]+1;
87              x=son[x][1];
88          }
89          else{
90              x=son[x][0];
91          }
92          pushdown(x);
93      }
94      return x;
95  }
96  void rec(int x)
97  {
98      if(!x)return;
99      int l=son[x][0],r=son[x][1];
100     rec(l);rec(r);q.push(x);
101     fa[x]=son[x][0]=son[x][1]=0;
102     tag[x]=rev[x]=0;
103 }
104 int split(int k,int tt)
105 {
106     int x=get(k),y=get(k+tt+1);
107     splay(x,rt);splay(y,son[x][1]);
108     return son[y][0];
109 }
110 void modify(int k,int tt,int val)
111 {
112     int x=split(k,tt),y=fa[x];
113     v[x]=val;tag[x]=1;sum[x]=sz[x]*val;
114     if(val>=0)lx[x]=rx[x]=mx[x]=sum[x];
115     else lx[x]=rx[x]=0,mx[x]=val;
116     update(y);update(fa[y]);
```

```
}
void rever(int k,int tt)
{
    int x=split(k,tt),y=fa[x];
    if(!tag[x])
    {
        rev[x]^=1;
        swap(son[x][0],son[x][1]);
        swap(lx[x],rx[x]);
        update(y);update(fa[y]);
    }
}
void delet(int k,int tt)
{
    int x=split(k,tt),y=fa[x];
    rec(x);son[y][0]=0;
    update(y);update(fa[y]);
}
void build(int l,int r,int f)
{
    if(l>r)return;
    int mid=(l+r)>>1,now=id[mid],last=id[f];
    if(l==r)
    {
        sum[now]=a[l];sz[now]=1;
        tag[now]=rev[now]=0;
        if(a[l]>=0)lx[now]=rx[now]=mx[now]=a[l];
        else lx[now]=rx[now]=0,mx[now]=a[l];
    }
    else build(l,mid-1,mid),build(mid+1,r,mid);
    v[now]=a[mid];fa[now]=last;update(now);
    son[last][mid>=f]=now;
   }
void insert(int k,int tt)
{
    for(int i=1;i<=tt;i++)a[i]=read();
    for(int i=1;i<=tt;i++)
    if(!q.empty())id[i]=q.front(),q.pop();
    else id[i]=++cnt;
    build(1,tt,0);int z=id[(1+tt)>>1];
    int x=get(k+1),y=get(k+2);
    splay(x,rt);splay(y,son[x][1]);
    fa[z]=y;son[y][0]=z;
    update(y);update(x);
}
int main()
{
// freopen("seq2005.in","r",stdin);
// freopen("seq2005.out","w",stdout);
    scanf("%d%d",&n,&m);
    mx[0]=a[1]=a[n+2]=-inf;
    for(int i=2;i<=n+1;i++)scanf("%d",a+i);
    for(int i=1;i<=n+2;i++)id[i]=i;
```

```
    build(1,n+2,0);
    rt=(n+3)>>1;cnt=n+2;
    int k,tt,val;
    char op[19];
    for(int i=1;i<=m;++i)
    {
        scanf("%s",op);
        if(op[0]!='M'||op[2]!='X')scanf("%d%d",&k,&tt);
        if(op[0]=='I')insert(k,tt);
        if(op[0]=='D')delet(k,tt);
        if(op[0]=='M')
        {
            if(op[2]=='X')  printf("%d\n",mx[rt]);
            else     scanf("%d",&val),modify(k,tt,val);
        }
        if(op[0]=='R')rever(k,tt);
        if(op[0]=='G'){printf("%d\n",sum[split(k,tt)]);}
    }
    return 0;
}
```

3.3 KDtree

3.3.1 动态插入

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5+5;

int n,K;
struct nd{
    int a[2],val;
    friend bool operator < (nd a,nd b){
        return a.a[K]<b.a[K];
    }
}b[2];

struct KDtree{
    int rt,tot,cnt;
    int ls[N],rs[N],sz[N],val[N],st[N];
    nd mn[N],mx[N],w[N],a[N];
    inline void up(int x){
        for(int i=0;i<2;++i){
            mn[x].a[i]=min(w[x].a[i],min(mn[ls[x]].a[i],mn[rs[x]].a[i]));
            mx[x].a[i]=max(w[x].a[i],max(mx[ls[x]].a[i],mx[rs[x]].a[i]));
        }
        val[x]=val[ls[x]]+val[rs[x]]+w[x].val;
        sz[x]=sz[ls[x]]+sz[rs[x]]+1;
    }
    int build(int l,int r,int k){
        K=k;
```

```
28          int x=st[cnt--],mid=l+r>>1;
29          nth_element(a+l,a+mid,a+r+1);
30          w[x]=a[mid];
31          val[x]=w[x].val;
32          sz[x]=1;
33          if(l<mid)ls[x]=build(l,mid-1,k^1);
34          if(mid<r)rs[x]=build(mid+1,r,k^1);
35          up(x);
36          return x;
37      }
38      void recycle(int x){
39          a[++cnt]=w[x];
40          st[cnt]=x;
41          if(ls[x])recycle(ls[x]);
42          if(rs[x])recycle(rs[x]);
43          ls[x]=rs[x]=0;
44      }
45      void insert(int &x,nd P,int k=0){
46          if(!x){
47              x=++tot;
48              sz[x]=1;
49              val[x]=P.val;
50              mn[x]=mx[x]=w[x]=P;
51              return;
52          }
53          if(w[x].a[0]==P.a[0]&&w[x].a[1]==P.a[1]){
54              val[x]+=P.val;
55              w[x].val+=P.val;
56              return;
57          }
58          int &y=P.a[k]<w[x].a[k]?ls[x]:rs[x];
59          insert(y,P,k^1);
60          up(x);
61          if(sz[y]>sz[x]*0.75){
62              recycle(x);
63              x=build(1,cnt,k);
64          }
65      }
66      int query(int x,nd l,nd r){
67          int c1=0,c2=0;
68          for(int i=0;i<2;++i){
69              if(l.a[i]<=mn[x].a[i]&&mx[x].a[i]<=r.a[i])c1++;
70              if(mx[x].a[i]<l.a[i]||mn[x].a[i]>r.a[i])return 0;
71              if(l.a[i]<=w[x].a[i]&&w[x].a[i]<=r.a[i])c2++;
72          }
73          if(c1==2)return val[x];
74          int ret=(c2==2)?w[x].val:0;
75          if(ls[x])ret+=query(ls[x],l,r);
76          if(rs[x])ret+=query(rs[x],l,r);
77          return ret;
78      }
79  }T;
80
```

```
int main(){
   for(int i=0;i<2;++i)T.mn[0].a[i]=2e9,T.mx[0].a[i]=-2e9;
   scanf("%d",&n);
   for(int i=1,op,val,ans=0;;++i){
      scanf("%d",&op);
      if(op==1){
         b[0].a[0]=read();b[0].a[1]=read();b[0].val=read();
         T.insert(T.rt,b[0]);
         /*
         if(T.tot%10000==0){
            T.cnt=0;
            T.recycle(T.rt);
            T.rt=T.build(1,tmp,0);
         }
         */
      }
      else if(op==2){
         for(int j=0;j<2;++j){
            b[j].a[0]=read(),b[j].a[1]=read();
         }
         printf("%d\n",ans=T.query(T.rt,b[0],b[1]));
      }
      else break;
   }
}
```

3.3.2 求 k 远点对

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;

int n,k,K;
multiset<ll>s;

struct nd{
   int a[2];
   friend bool operator < (nd a,nd b){
      return a.a[K]<b.a[K];
   }
};

inline ll sqr(ll x){
   return x*x;
}

struct KDtree{
   int ls[N],rs[N];
   nd mx[N],mn[N],w[N];

   void up(int x){
```

```
25      for(int i=0;i<2;++i){
26          mn[x].a[i]=min(w[x].a[i],min(mn[ls[x]].a[i],mn[rs[x]].a[i]));
27          mx[x].a[i]=max(w[x].a[i],max(mx[ls[x]].a[i],mx[rs[x]].a[i]));
28      }
29  }
30
31  int build(int l,int r,int k=0){
32      K=k;
33      int x=l+r>>1;
34      nth_element(w+l,w+x,w+r+1);
35      if(l<x)ls[x]=build(l,x-1,k^1);
36      if(r>x)rs[x]=build(x+1,r,k^1);
37      up(x);
38      return x;
39  }
40
41  //以下为欧式距离的版本，若要求曼哈顿距离，将所有sqr改成abs即可。
42  inline ll mx_dis(int x,nd P){
43      ll r=0;
44      for(int i=0;i<2;++i)r+=max(sqr(mn[x].a[i]-P.a[i]),sqr(mx[x].a[i]-P.a[i]));
45      return r;
46  }
47
48  /*
49  inline ll mn_dis(int x,nd P){//当所求为k近点对时，估价函数应由mx_dis改为mn_dis。
50      ll r=0;
51      for(int i=0;i<2;++i)
52      if(P.a[i]<mn[x].a[i])r+=sqr(mn[x].a[i]-P.a[i]);
53      else if(P.a[i]>mx[x].a[i])r+=sqr(mx[x].a[i]-P.a[i]);
54      return r;
55  }
56  */
57
58  inline ll dis(int x,nd P){
59      ll r=0;
60      for(int i=0;i<2;++i)r+=sqr(w[x].a[i]-P.a[i]);
61      return r;
62  }
63
64  inline void insert(ll val){
65      if(s.size()<k)s.insert(val);
66      else{
67          multiset<ll>::iterator it=s.begin();
68          if(*it<val){
69              s.erase(it);
70              s.insert(val);
71          }
72      }
73  }
74
75  void query(int x,nd P){
76      if(!x)return;
77      insert(dis(x,P));
```

```
78          ll val[2];
79          val[0]=ls[x]?mx_dis(ls[x],P):-1e9;
80          val[1]=rs[x]?mx_dis(rs[x],P):-1e9;
81          int y=val[0]>val[1]?ls[x]:rs[x];
82          if(s.size()<k||max(val[0],val[1])>*s.begin()){
83              query(y,P);
84              if(s.size()<k||min(val[0],val[1])>*s.begin())query(ls[x]+rs[x]-y,P);
85          }
86      }
87  }T;
88
89  int main(){
90      scanf("%d%d",&n,&k);//k<=100
91      k<<=1;
92      for(int i=0;i<2;++i){
93          T.mn[0].a[i]=2e9;
94          T.mx[0].a[i]=-2e9;
95      }
96      for(int i=1;i<=n;++i)scanf("%d%d",&T.w[i].a[0],&T.w[i].a[1]);
97      int rt=T.build(1,n);
98      for(int i=1;i<=n;++i)T.query(rt,T.w[i]);
99      printf("%lld\n",*s.begin());
100 }
```

3.4 LCT

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int N = 1e5+5;
4
5   int n,m,fa[N],son[N][2],size[N],key[N],sum[N],mx[N],tag[N],rev[N];
6
7   inline bool isroot(int x){
8       return son[fa[x]][0]!=x&&son[fa[x]][1]!=x;
9   }
10
11  inline void Add(int x,int v){
12      key[x]+=v;sum[x]+=v*size[x];mx[x]+=v;tag[x]+=v;
13  }
14
15  inline void push_up(int x){
16      size[x]=size[son[x][0]]+size[son[x][1]]+1;
17      sum[x]=sum[son[x][0]]+sum[son[x][1]]+key[x];
18      mx[x]=max(max(mx[son[x][0]],mx[son[x][1]]),key[x]);
19  }
20
21  inline void push_down(int x){
22      if(rev[x])rev[x]^=1,rev[son[x][0]]^=1,rev[son[x][1]]^=1,std::swap(son[x][0],son[
            x][1]);
23      if(tag[x])Add(son[x][0],tag[x]),Add(son[x][1],tag[x]),tag[x]=0;
24  }
25
```

```
26 inline void Rotate(int x){
27     push_down(fa[x]);push_down(x);
28     int y=fa[x],z=fa[y],t=(son[y][0]==x);
29     if(!isroot(y)) son[z][son[z][1]==y]=x;
30     son[y][!t]=son[x][t];fa[son[y][!t]]=y;
31     son[x][t]=y;
32     fa[y]=x;fa[x]=z;
33     push_up(y);push_up(x);
34 }
35 
36 inline void Splay(int x){
37     push_down(x);
38     while(!isroot(x)){
39         int y=fa[x],z=fa[y];
40         if(!isroot(y)){
41             if(son[z][0]==y^son[y][0]==x) Rotate(x);
42             else Rotate(y);
43         }
44         Rotate(x);
45     }
46 }
47 
48 inline void Access(int x){
49     for(int y=0;x;y=x,x=fa[x]){
50         Splay(x);
51         son[x][1]=y;
52         push_up(x);
53     }
54 }
55 
56 inline void Makeroot(int x){
57     Access(x);
58     Splay(x);
59     rev[x]^=1;
60 }
61 
62 inline void Split(int x,int y){
63     Makeroot(x);
64     Access(y);
65     Splay(x);
66 }
67 
68 inline void Link(int x,int y){
69     Makeroot(x);
70     fa[x]=y;
71 }
72 
73 inline void Cut(int x,int y){
74     Split(x,y);
75     son[x][1]=fa[y]=0;
76 }
77 
78 inline int Find(int x){
```

```
    Access(x);
    Splay(x);
    while(son[x][0])x=son[x][0];
    return x;
}

int main(){
    mx[0]=-0x7fffffff;
    scanf("%d%d",&n,&m);
    for(int i=1,x;i<=n;i++){
        scanf("%d",&x);
        size[i]=1;Add(i,x);tag[i]=0;
    }
    for(int i=1,opt,x,y,z;i<=m;++i){
        scanf("%d%d%d",&opt,&x,&y);
        if(opt==1) Link(x,y);
        else if(opt==2) Cut(x,y);
        else if(opt==3) Splay(x),key[x]=y,push_up(x);
        else if(opt==4) Split(x,y),printf("%d\n",mx[x]);
        else if(opt==5) Split(x,y),printf("%d\n",sum[x]);
        else if(opt==6) printf("%d\n",Find(x)==Find(y));
        else scanf("%d",&z),Split(x,y),Add(x,z);
    }
}
```

3.5 树链剖分

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;

int n,m,r,mod;
int head[N];
int son[N],fa[N],dep[N],size[N],top[N],st[N],ed[N],p[N];
int a[N];

struct nd{
    int ne,to;
}e[N<<1];

void in(int x,int y){
    static int cnt;
    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
}

void dfs1(int x){
    size[x]=1;
    for(int i=head[x];i;i=e[i].ne)
    if(e[i].to!=fa[x]){
        int y=e[i].to;
        fa[y]=x;
```

```
26        dep[y]=dep[x]+1;
27        dfs1(y);
28        size[x]+=size[y];
29        if(size[son[x]]<size[y])son[x]=y;
30    }
31 }
32
33 void dfs2(int x){
34    static int totw;
35    st[x]=ed[x]=++totw;
36    p[totw]=x;
37    int y=son[x];
38    if(!y)return;
39    top[y]=top[x];dfs2(y);
40    for(int i=head[x];i;i=e[i].ne){
41        int y=e[i].to;
42        if(fa[x]==y||son[x]==y)continue;
43        top[y]=y;dfs2(y);
44    }
45    ed[x]=totw;
46 }
47
48 struct ST{
49    int tot;
50    int ls[N<<1],rs[N<<1];
51    ll w[N<<1],tag[N<<1];
52
53    void up(int x){
54        w[x]=(w[ls[x]]+w[rs[x]])%mod;
55    }
56
57    void down(int x,int l,int r){
58        if(!tag[x])return;
59        int mid=l+r>>1;
60        if(ls[x]){
61            (w[ls[x]]+=tag[x]*(mid-l+1)%mod)%=mod;
62            (tag[ls[x]]+=tag[x])%=mod;
63        }
64        if(rs[x]){
65            (w[rs[x]]+=tag[x]*(r-mid)%mod)%=mod;
66            (tag[rs[x]]+=tag[x])%=mod;
67        }
68        tag[x]=0;
69    }
70
71    int build(int l,int r){
72        int x=++tot;
73        if(l==r){
74            w[x]=a[p[l]];
75            return x;
76        }
77        int mid=l+r>>1;
78        ls[x]=build(l,mid);
```

```
79          rs[x]=build(mid+1,r);
80          up(x);
81          return x;
82      }
83
84      void change(int x,int l,int r,int L,int R,ll val){
85          if(L<=l&&r<=R){
86              (tag[x]+=val)%=mod;
87              (w[x]+=val*(r-l+1)%mod)%=mod;
88              return;
89          }
90          down(x,l,r);
91          int mid=l+r>>1;
92          if(L<=mid)change(ls[x],l,mid,L,R,val);
93          if(mid<R)change(rs[x],mid+1,r,L,R,val);
94          up(x);
95      }
96
97      ll sum(int x,int l,int r,int L,int R){
98          if(L<=l&&r<=R)return w[x];
99          down(x,l,r);
100         int mid=l+r>>1;
101         ll ret=0;
102         if(L<=mid)(ret+=sum(ls[x],l,mid,L,R))%=mod;
103         if(mid<R)(ret+=sum(rs[x],mid+1,r,L,R))%=mod;
104         up(x);
105         return ret;
106     }
107 }T;
108
109 void change(int x,int y,ll val){
110     while(top[x]!=top[y]){
111         if(dep[top[x]]<dep[top[y]])swap(x,y);
112         T.change(1,1,n,st[top[x]],st[x],val);
113         x=fa[top[x]];
114     }
115     if(st[x]>st[y])swap(x,y);
116     T.change(1,1,n,st[x],st[y],val);
117 }
118
119 ll sum(int x,int y){
120     ll ret=0;
121     while(top[x]!=top[y]){
122         if(dep[top[x]]<dep[top[y]])swap(x,y);
123         (ret+=T.sum(1,1,n,st[top[x]],st[x]))%=mod;
124         x=fa[top[x]];
125     }
126     if(st[x]>st[y])swap(x,y);
127     (ret+=T.sum(1,1,n,st[x],st[y]))%=mod;
128     return ret;
129 }
130
131 int main(){
```

```
    scanf("%d%d%d%d",&n,&m,&r,&mod);
    for(int i=1;i<=n;++i)scanf("%d",&a[i]);
    for(int i=1,x,y;i<n;++i){
        scanf("%d%d",&x,&y);
        in(x,y);
        in(y,x);
    }
    dep[1]=1;dfs1(r);
    top[r]=r;dfs2(r);
    T.build(1,n);
    for(int i=1,op,x,y;i<=m;++i){
        ll z;
        scanf("%d",&op);
        if(op==1){
            scanf("%d%d%lld",&x,&y,&z);
            change(x,y,z%mod);
        }
        if(op==2){
            scanf("%d%d",&x,&y);
            printf("%lld\n",sum(x,y));
        }
        if(op==3){
            scanf("%d%lld",&x,&z);
            T.change(1,1,n,st[x],ed[x],z%mod);
        }
        if(op==4){
            scanf("%d",&x);
            printf("%lld\n",T.sum(1,1,n,st[x],ed[x]));
        }
    }
    return 0;
}
```

3.6 二维 ST 表

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 5e2+5;
const int M = 10;

int n,m;
int f[N][N][M][M];

void prepare(){
    for(int k=1;k<M;++k)
    for(int i=1;i<=n;++i)
    for(int j=1;j+(1<<k-1)<=m;++j){
        f[i][j][0][k]=max(f[i][j][0][k-1],f[i][j+(1<<k-1)][0][k-1]);
    }
    for(int k=1;k<M;++k)
    for(int i=1;i+(1<<k-1)<=n;++i)
```

```
    for(int j=1;j<=m;++j){
        f[i][j][k][0]=max(f[i][j][k-1][0],f[i+(1<<k-1)][j][k-1][0]);
    }
    for(int k1=1;k1<M;++k1)
    for(int k2=1;k2<M;++k2)
    for(int i=1;i+(1<<k1-1)<=n;++i)
    for(int j=1;j+(1<<k2-1)<=m;++j){
        f[i][j][k1][k2]=max(f[i][j][k1][k2],f[i][j][k1-1][k2-1]);
        f[i][j][k1][k2]=max(f[i][j][k1][k2],f[i+(1<<k1-1)][j][k1-1][k2-1]);
        f[i][j][k1][k2]=max(f[i][j][k1][k2],f[i][j+(1<<k2-1)][k1-1][k2-1]);
        f[i][j][k1][k2]=max(f[i][j][k1][k2],f[i+(1<<k1-1)][j+(1<<k2-1)][k1-1][k2-1]);
    }
}

int getmx(int lx,int ly,int rx,int ry){
    int k1=log2(rx-lx+1);
    int k2=log2(ry-ly+1);
    int ret=0;
    ret=max(ret,f[lx][ly][k1][k2]);
    ret=max(ret,f[rx-(1<<k1)+1][ly][k1][k2]);
    ret=max(ret,f[lx][ry-(1<<k2)+1][k1][k2]);
    ret=max(ret,f[rx-(1<<k1)+1][ry-(1<<k2)+1][k1][k2]);
    return ret;
}

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;++i)
    for(int j=1;j<=m;++j){
        scanf("%d",&f[i][j][0][0]);
    }
    prepare();
    cout<<getmx(1,1,n,m)<<endl;
}
```

3.7 树上莫队

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;
const int B = 19;

int n,Q;
int head[N],cnt;
int id[N];
int f1[N],f2[N],d[N],f[N][B],st[N],ed[N],dfx[N*2],totw,block;
int tong[2][N];
ll ans[N];
bool inq[N];

struct nd{
```

```
16      int ne,to;
17  }e[N*2];
18
19  void in(int x,int y){
20      e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
21  }
22
23  struct qs{
24      int id,m,l,r,b;
25  }q[N*2];
26
27  bool cmp(qs a,qs b){
28      if(a.b!=b.b)return a.b<b.b;
29      return a.r>b.r;
30  }
31
32  void dfs(int x,int fa){
33      st[x]=++totw;
34      dfx[totw]=x;
35      for(int i=head[x];i;i=e[i].ne)
36      if(e[i].to!=fa){
37          int y=e[i].to;
38          d[y]=d[x]+1;
39          f[y][0]=x;
40          dfs(y,x);
41      }
42      ed[x]=++totw;
43      dfx[totw]=x;
44  }
45
46  int lca(int x,int y){
47      if(d[x]<d[y])swap(x,y);
48      for(int i=B-1;i>=0;--i)
49      if(d[f[x][i]]>=d[y]){
50          x=f[x][i];
51      }
52      if(x==y)return x;
53      for(int i=B-1;i>=0;--i)
54      if(f[x][i]!=f[y][i]){
55          x=f[x][i];
56          y=f[y][i];
57      }
58      return f[x][0];
59  }
60
61  void prepare(){
62      d[1]=1;dfs(1,-1);
63      for(int i=1;i<B;++i)
64      for(int t=1;t<=n;++t)
65      f[t][i]=f[f[t][i-1]][i-1];
66  }
67
68  ll ret=0;
```

```
69
70  void insert(int x){
71      inq[x]^=1;
72      if(inq[x]){
73          tong[id[x]][f1[x]]++;
74          ret+=tong[id[x]^1][f1[x]];
75      }
76      else{
77          tong[id[x]][f1[x]]--;
78          ret-=tong[id[x]^1][f1[x]];
79      }
80  }
81
82
83  int main(){
84      scanf("%d",&n);
85      block=sqrt(n)+1;
86      for(int i=1;i<=n;++i)scanf("%d",id+i);
87      for(int i=1;i<=n;++i)scanf("%d",f1+i),f2[i]=f1[i];
88      sort(f2+1,f2+n+1);
89      int nn=unique(f2+1,f2+n+1)-f2-1;
90      for(int i=1;i<=n;++i)f1[i]=lower_bound(f2+1,f2+nn+1,f1[i])-f2;
91      for(int i=1,x,y;i<n;++i){
92          scanf("%d%d",&x,&y);
93          in(x,y);
94          in(y,x);
95      }
96      prepare();
97      scanf("%d",&Q);
98      for(int i=1,a,b;i<=Q;++i){
99          scanf("%d%d",&a,&b);
100         q[i].id=i;
101         int p=lca(a,b);
102         if(st[a]>st[b])swap(a,b);
103         if(p==a)q[i].l=st[a],q[i].r=st[b],q[i].m=0;
104         else q[i].l=ed[a],q[i].r=st[b],q[i].m=p;
105         q[i].b=q[i].l/block+1;
106     }
107     int L=1,R=0;
108     sort(q+1,q+Q+1,cmp);
109     for(int i=1;i<=Q;++i)
110     {
111         int l=q[i].l,r=q[i].r;
112         while(L<l)insert(dfx[L]),L++;
113         while(L>l)L--,insert(dfx[L]);
114         while(R>r)insert(dfx[R]),R--;
115         while(R<r)R++,insert(dfx[R]);
116         if(q[i].m)insert(q[i].m);
117         ans[q[i].id]=ret;
118         if(q[i].m)insert(q[i].m);
119     }
120     for(int i=1;i<=Q;++i)printf("%I64d\n",ans[i]);
121 }
```

3.8 可撤销并查集

```
struct UFS {
   stack<pair<int*, int>> stk;
   int fa[N], rnk[N];
   inline void init(int n) {
      for (int i = 0; i <= n; ++i) fa[i] = i, rnk[i] = 0;
   }
   inline int Find(int x) {
      while(x^fa[x]) x = fa[x];
      return x;
   }
   int Merge(int x, int y) {
      int c=0;
      x = Find(x), y = Find(y);
      if(x == y){
         ++block;
         return c;
      }
      if(rnk[x] <= rnk[y]) {
         stk.push({fa+x, fa[x]});
         fa[x] = y;
         c++;
         if(rnk[x] == rnk[y]) {
            stk.push({rnk+y, rnk[y]});
            rnk[y]++;
            c++;
         }
      }
      else {
         stk.push({fa+y, fa[y]});
         fa[y] = x;
         c++;
      }
      return c;
   }
   #define fi first
   #define se second
   inline void Undo() {
      *stk.top().fi = stk.top().se;
      stk.pop();
   }
}ufs;
```

3.9 树状数组上二分

维护一个序列 a，支持修改某个元素的值和寻找最小的满足 $\sum_{i=1}^{t} a_i >= k$ 的 t。(如果把该序列下标看成值域，值看作出现的次数，则第二种操作表示找第 k 大元素)

显然线段树上二分可以做到 $O(logn)$ 的查询修改复杂度。

但是树状数组同样存在一种类似数位 DP 的方式，将修改操作做到一个 log 的复杂度。(时间空间上更优秀)

考虑树状数组的定义方式，第 i 位 $Bit[i]=a[i-2^k+1]+a[i-2^k+2]+...+a[i]$（$k$ 表示 i 的二进制最低非零位，2^k 即 $lowbit(i)$）（求前缀和时一直减 $lowbit$ 类似于数位 DP 中枚举最低紧贴上限的二进制位来遍历所有数）

利用数位 DP 的思想，从高位到低位逐位确定所求值。（具体看代码实现）

```
struct BIT{
    int a[N];

    void add(int x,int d){
        for(;x<=n;x+=x&-x)a[x]+=d;
    }

    int sum(int x){
        int ret=0;
        for(;x;x-=x&-x)ret+=a[x];
        return ret;
    }

    int get(int k){
        int x=0;
        for(int i=MAXLOGN;~i;--i){
            int x1=x+(1<<i);
            if(x1<=n&&k>a[x1]){
                k-=a[x1];
                x=x1;
                //类似于数位DP，此处所有下标<=x的原序列的值都已从k中减去
            }
        }
        return x+1;//此处x在满足前x个位置的和小于k的条件下尽可能地大，所以第k大在第x+1个位置上
    }
}T;
```

3.10 树状数组区间加区间求和

假设原数组是 $\{a_i\}$，差分数组 $\{d_i=a_i-a_{i-1}\}$

易知 $a_x=\sum_{i=1}^{x}d_i$

将区间和拆成两个前缀和之差后，可得 $Ans=\sum_{i=1}^{n}a_i=\sum_{i=1}^{n}d_i(x-i+1)$ $Ans=\sum_{i=1}^{n}(n+1)d_i-\sum_{i=1}^{n}id_i$

用树状数组维护 d_i 与 id_i 的前缀和即可，每次区间加仅会改变 l 与 $r+1$ 处的 d_i 与 id_i 的值，这一部分对应树状数组的单点加，求 d_i 与 id_i 的前缀和则对应树状数组的前缀求和。

```
struct BIT{
    ll a[N],b[N];

    void add(int x,ll d){
        int t=x;
        for(;x<=n;x+=x&-x){
            a[x]+=d;
            b[x]+=d*t;
```

```
        }
    }

    void add(int l,int r,ll d){
        add(l,d);
        add(r+1,-d);
    }

    ll sum(int x){
        ll ret=0;
        int t=x+1;
        for(;x;x-=x&-x){
            ret+=a[x]*t;
            ret-=b[x];
        }
        return ret;
    }

    ll sum(int l,int r){
        return sum(r)-sum(l-1);
    }
}T;
```

4 字符串

4.1 后缀数组

4.1.1 后缀数组

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5+5;

int n;
int sa[N],tong[N],wa[N],wb[N],wv[N],rk[N],h[N];
char s[N];

int cmp(int *r,int a,int b,int l){
    return r[a]==r[b]&&r[a+l]==r[b+l];
}

void da(int n,int m){
    int i,j,p,*x=wa,*y=wb;
    for(i=0;i<m;++i)tong[i]=0;
    for(i=0;i<n;++i)tong[x[i]=s[i]]++;
    for(i=1;i<m;++i)tong[i]+=tong[i-1];
    for(i=n-1;i>=0;--i)sa[--tong[x[i]]]=i;
    for(j=1,p=1;p<n;j<<=1,m=p){
        for(p=0,i=n-j;i<n;++i)y[p++]=i;
        for(i=0;i<n;++i)if(sa[i]>=j)y[p++]=sa[i]-j;
        for(i=0;i<n;++i)wv[i]=x[y[i]];
        for(i=0;i<m;++i)tong[i]=0;
```

```
        for(i=0;i<n;++i)tong[wv[i]]++;
        for(i=1;i<m;++i)tong[i]+=tong[i-1];
        for(i=n-1;i>=0;--i)sa[--tong[wv[i]]]=y[i];
        for(swap(x,y),p=1,x[sa[0]]=0,i=1;i<n;++i)
        x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
    }
    return;
}

void calh(int n){
    int i,j,k=0;
    for(i=1;i<=n;i++)rk[sa[i]]=i;
    for(i=0;i<n;h[rk[i++]]=k)
    for(k?k--:0,j=sa[rk[i]-1];s[i+k]==s[j+k];k++);
    return;
}

int main(){
    scanf("%s",s);
    n=strlen(s);
    da(n+1,255);
    calh(n);
    for(int i=1;i<=n;++i)printf("%d ",sa[i]+1);
    printf("\n");
    for(int i=0;i<n;++i)printf("%d ",rk[i]);
    printf("\n");
    for(int i=2;i<=n;++i)printf("%d ",h[i]);
}
```

4.1.2 后缀数组 +ST 表 + 主席树

后缀数组 +ST 表 + 主席树求区间第 k 大

子串 $S_l, S_{l+1}, ..., S_r$ 在 S 中第 k 次出现的位置

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
const int M = 20;

int Case;
int n,Q,f[N][M],rt[N];
int sa[N],tong[N],wa[N],wb[N],wv[N],rk[N],h[N];
char s[N];

int cmp(int *r,int a,int b,int l){
    return r[a]==r[b]&&r[a+l]==r[b+l];
}

void da(int n,int m){
    int i,j,p,*x=wa,*y=wb;
    for(i=0;i<m;++i)tong[i]=0;
    for(i=0;i<n;++i)tong[x[i]=s[i]]++;
```

```
    for(i=1;i<m;++i)tong[i]+=tong[i-1];
    for(i=n-1;i>=0;--i)sa[--tong[x[i]]]=i;
    for(j=1,p=1;p<n;j<<=1,m=p){
        for(p=0,i=n-j;i<n;++i)y[p++]=i;
        for(i=0;i<n;++i)if(sa[i]>=j)y[p++]=sa[i]-j;
        for(i=0;i<n;++i)wv[i]=x[y[i]];
        for(i=0;i<m;++i)tong[i]=0;
        for(i=0;i<n;++i)tong[wv[i]]++;
        for(i=1;i<m;++i)tong[i]+=tong[i-1];
        for(i=n-1;i>=0;--i)sa[--tong[wv[i]]]=y[i];
        for(swap(x,y),p=1,x[sa[0]]=0,i=1;i<n;++i)
        x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
    }
    return;
}

void calh(int n){
    int i,j,k=0;
    for(i=1;i<=n;i++)rk[sa[i]]=i;
    for(i=0;i<n;h[rk[i++]]=k)
    for(k?k--:0,j=sa[rk[i]-1];s[i+k]==s[j+k];k++);
    return;
}

int lcp(int l,int r){
    if(l==r)return n-r+1;
    l++;
    int d=log2(r-l+1);
    return min(f[l][d],f[r-(1<<d)+1][d]);
}

struct CMT{
    int tot;
    int ls[N*M],rs[N*M],w[N*M];
    void init(){
        for(int i=1;i<=tot;++i)ls[i]=rs[i]=w[i]=0;
        tot=0;
    }
    int insert(int y,int l,int r,int p){
        int x=++tot;
        ls[x]=ls[y];rs[x]=rs[y];
        w[x]=w[y]+1;
        if(l==r)return x;
        int mid=l+r>>1;
        if(p<=mid)ls[x]=insert(ls[y],l,mid,p);
        else rs[x]=insert(rs[y],mid+1,r,p);
        return x;
    }
    int kth(int y,int x,int l,int r,int k){
        if(l==r)return l;
        int mid=l+r>>1;
        if(w[ls[x]]-w[ls[y]]>=k)return kth(ls[y],ls[x],l,mid,k);
        else return kth(rs[y],rs[x],mid+1,r,k-(w[ls[x]]-w[ls[y]]));
```

```
    }
}T;

int get(int L,int R,int k){
    static int l,r,h;
    h=R-L+1;
    l=1,r=rk[L];
    while(l!=r){
        int mid=l+r>>1;
        if(lcp(mid,rk[L])>=h)r=mid;
        else l=mid+1;
    }
    int a=l;
    l=rk[L],r=n;
    while(l!=r){
        int mid=l+r+1>>1;
        if(lcp(rk[L],mid)>=h)l=mid;
        else r=mid-1;
    }
    int b=r;
    if(b-a+1<k)return -1;
    return T.kth(rt[a-1],rt[b],1,n,k);
}

int main(){
    scanf("%d",&Case);
    while(Case--){
        T.init();
        scanf("%d%d",&n,&Q);
        scanf("%s",s);
        da(n+1,255);
        calh(n);
        for(int i=n;i>=1;--i)rk[i]=rk[i-1];
        for(int i=1;i<=n;++i)sa[i]++;
        for(int i=1;i<=n;++i)f[i][0]=h[i];
        for(int j=1;j<M;++j)
        for(int i=1;i+(1<<j)-1<=n;++i)
        f[i][j]=min(f[i][j-1],f[i+(1<<j-1)][j-1]);
        for(int i=1;i<=n;++i)rt[i]=T.insert(rt[i-1],1,n,sa[i]);
        for(int i=1,l,r,k;i<=Q;++i){
            scanf("%d%d%d",&l,&r,&k);
            printf("%d\n",get(l,r,k));
        }
    }
}
```

4.2 AC 自动机

4.2.1 AC 自动机

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

```
3  const int N = 501*201+5;
4  const int maxf = 128;
5
6  int ans[1001],ans2,tmp,tr[N][maxf],tot,tag[N],fail[N];
7  char s[10001];
8  bool use[N];
9
10 void insert(int noww){
11    int n=strlen(s),now=0;
12    for(int i=0;i<n;++i){
13       if(tr[now][s[i]]==-1){
14          tr[now][s[i]]=++tot;
15          for(int j=0;j<maxf;++j)tr[tot][j]=-1;
16          tag[tot]=0;
17       }
18       now=tr[now][s[i]];
19    }
20    tag[now]=noww;
21 }
22
23 void getfail(){
24    queue<int>q;
25    fail[0]=0;
26    for(int i=0;i<maxf;++i)
27    if(tr[0][i]!=-1)
28    fail[tr[0][i]]=0,q.push(tr[0][i]);
29    else tr[0][i]=0;
30    while(!q.empty()){
31       int x=q.front();q.pop();
32       for(int j=0;j<maxf;++j)
33       if(tr[x][j]!=-1)fail[tr[x][j]]=tr[fail[x]][j],q.push(tr[x][j]);
34       else tr[x][j]=tr[fail[x]][j];
35    }
36 }
37
38 void solve(int noww){
39    int n=strlen(s),now=0;
40    memset(use,0,sizeof(use));
41    tmp=0;
42    for(int i=0;i<n;++i){
43       now=tr[now][s[i]];
44       for(int j=now;j;j=fail[j])
45       if(tag[j]&&!use[tag[j]])
46       ans[++tmp]=tag[j],use[tag[j]]=1;
47    }
48    if(tmp){
49       ans2++;
50       printf("web %d:",noww);
51       sort(ans+1,ans+tmp+1);
52       for(int j=1;j<=tmp;++j)
53       printf(" %d",ans[j]);
54       printf("\n");
55    }
```

```
}

int main(){
   int n,m;
   scanf("%d",&n);
   for(int i=0;i<maxf;++i)tr[0][i]=-1;
   for(int i=1;i<=n;++i)
   scanf("%s",s),insert(i);
   getfail();
   scanf("%d",&m);
   for(int i=1;i<=m;++i)
   scanf("%s",s),solve(i);
   printf("total: %d\n",ans2);
}
```

4.2.2 AC 自动机 (last 优化)

```
#include<bits/stdc++.h>
#define id(x) x-'a'
using namespace std;
const int N = 51;
const int az = 27;

int tr[N][az],tot,T,head,tail,q[N],fail[N*az],tag[N*az],ans[N*az],last[N*az],to[11];
char s2[100000001],s1[11][N];

int insert(char *s,int now){
   int n=strlen(s);
   int x=0;
   for(int i=0;i<n;++i){
      if(tr[x][id(s[i])]==-1){
         tr[x][id(s[i])]=++tot;
         for(int t=0;t<az;++t)tr[tot][t]=-1;
      }
      x=tr[x][id(s[i])];
   }
   tag[x]=now;
   to[now]=x;
}

int getfail(){
   head=tail=0;
   for(int i=0;i<az;++i)
   if(tr[0][i]!=-1) last[tr[0][i]]=fail[tr[0][i]]=0,q[++tail]=tr[0][i];
   else tr[0][i]=0;
   while(head!=tail){
      int x=q[++head];
      for(int i=0;i<az;++i){
         if(tag[tr[fail[x]][i]])last[tr[x][i]]=tr[fail[x]][i];
         else last[tr[x][i]]=last[tr[fail[x]][i]];
         if(tr[x][i]!=-1) fail[tr[x][i]]=tr[fail[x]][i],q[++tail]=tr[x][i];
         else tr[x][i]=tr[fail[x]][i];
```

```
36          }
37
38      }
39      int n=strlen(s2);
40      int x=0;
41      for(int i=0;i<n;++i){
42          x=tr[x][id(s2[i])];
43          ans[x]++;
44      }
45      for(int i=tail;i;i--)
46      ans[last[q[i]]]+=ans[q[i]];
47  // for(int i=0;i<n;++i)
48  // {
49  //  for(int j=tr[x][id(s2[i])];j;j=fail[j])
50  //  if(tag[j])ans[tag[j]]++;
51  //  x=tr[x][id(s2[i])];
52  // }
53      return 0;
54  }
55
56  int main(){
57      for(int i=0;i<az;++i)tr[0][i]=-1;
58      scanf("%d",&T);
59      for(int i=1;i<=T;++i){scanf("%s",s1[i]);insert(s1[i],i);}
60      scanf("%s",&s2);
61      getfail();
62      for(int i=1;i<=T;++i)
63      cout<<s1[i]<<" "<<ans[to[i]]<<endl;
64      return 0;
65  }
```

4.3 回文树

4.3.1 回文树

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   const int N = 5e5+1;
5
6   char s[N];
7   int n;
8
9   struct PalindromesTree{
10      int tot,last,l[N],son[N][26],c[N],f[N];
11
12      int newnode(int x){
13          l[++tot]=x;
14          return tot;
15      }
16
17      void init(){
```

```
        tot=-1;last=0;
        f[newnode(0)]=1;
        f[newnode(-1)]=0;
    }

    int get(int x,int n){
        while(s[n]!=s[n-l[x]-1])x=f[x];
        return x;
    }

    void add(int x,int i){
        int y=get(last,i);
        if(!son[y][x]){
            int now=newnode(l[y]+2),ne=get(f[y],i);
            f[now]=son[ne][x];
            son[y][x]=now;
        }
        c[last=son[y][x]]++;
    }

    ll solve(){
        for(int i=1;i<=n;++i)add(s[i]-'a',i);
        for(int i=tot;i>=1;--i)c[f[i]]+=c[i];
        ll ret=0;
        for(int i=1;i<=tot;++i)ret=max(ret,(ll)l[i]*c[i]);
        return ret;
    }
}T;

int main(){
    scanf("%s",s+1);
    T.init();
    n=strlen(s+1);
    printf("%lld\n",T.solve());
}
```

4.3.2 可撤销回文树

问题：给定一个字符串，可以在末尾删除字符或者添加字符，请动态维护整个回文树。

与普通的回文树仅有两点区别：

1. 由于本题存在撤销操作，所以需要 $last$ 的回溯，进而需要记录所有历史版本的 $last$。

2. 由于 $last$ 可能回溯，普通的跳 $fail$ 操作的复杂度将从均摊 $O(n)$ 退化为 $O(n^2)$。所以需要利用类似并查集路径压缩的方式对跳 $fail$ 进行记忆化（记 $trans[x][c]$ 表示将 x 点首尾添加 c 字符后跳 $fail$ 后到达的节点，详见以下代码）。

均摊复杂度为 $O(n)$。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e6+5;
```

```
5  const ll mod = 1e9+7;
6
7  int Case;
8  int n,m,len,len1;
9  char s[N];
10 int base26[N];
11 int ans;
12
13 void prepare(){
14    base26[0]=1;
15    for(int i=1;i<N;++i)base26[i]=1ll*base26[i-1]*26%mod;
16 }
17
18 struct PalindromesTree{
19    int tot,last[N],l[N],son[N][26],f[N];
20    int gline[N];
21    int trans[N][26];
22
23    int newnode(int x){
24       l[++tot]=x;
25       memset(son[tot],0,sizeof(son[tot]));
26       memset(trans[tot],-1,sizeof(trans[tot]));
27       return tot;
28    }
29
30    void init(){
31       for(int i=0;i<=tot;++i){
32          l[i]=0;
33          gline[i]=0;
34          f[i]=0;
35          last[i]=0;
36       }
37       tot=-1;
38       f[newnode(0)]=1;
39       f[newnode(-1)]=0;
40    }
41
42    int get(int x,int c,int n){
43       int now=x;
44       while(s[n]!=s[n-l[x]-1]){
45          if(~trans[x][c])x=trans[x][c];
46          else x=f[x];
47       }
48       for(;now!=x;now=f[now]){
49          if(~trans[now][c])break;
50          trans[now][c]=x;
51       }
52       return x;
53    }
54
55    void add(int x){
56       int i=++len1;
57       s[i]=x+'a';
```

```
58          int y=get(last[i-1],x,i);
59          if(!son[y][x]){
60              int len=l[y]+2;
61              int now=newnode(len);
62              int ne=get(f[y],x,i);
63              f[now]=son[ne][x];
64              son[y][x]=now;
65              if(len<=n)gline[now]=(gline[f[now]]+1ll*base26[n-len]*(n-len+1)%mod)%mod;
66              else gline[now]=gline[f[now]];
67          }
68          last[i]=son[y][x];
69          (ans+=gline[last[i]])%=mod;
70      }
71
72      void dec(){
73          int i=len1--;
74          ans=(ans-gline[last[i]]+mod)%mod;
75      }
76  }T;
77
78  int main(){
79      prepare();
80      scanf("%d",&Case);
81      while(Case--){
82          scanf("%d%d",&n,&m);
83          memset(s,0,sizeof(s));
84          scanf("%s",s+1);
85          len=strlen(s+1);
86          ans=0;
87          len1=0;
88          T.init();
89          for(int i=1;i<=len;++i){
90              T.add(s[i]-'a');
91          }
92          printf("%d\n",ans);
93          for(int i=1,o;i<=m;++i){
94              scanf("%d",&o);
95              if(o==1){
96                  char s=getchar();
97                  while(s>'z'||s<'a')s=getchar();
98                  T.add(s-'a');
99              }
100             else{
101                 T.dec();
102             }
103             printf("%d\n",ans);
104         }
105     }
106 }
```

4.3.3 回文套娃计数

给定一个长度为 $n(n \leq 10^6)$ 的字符串 S，求其中的回文套娃个数。回文套娃定义为两个序列 $l_1 < l_2 < ... < l_k$ 和 $r_1 > r_2 > ... > r_k$，其中每个 $S[l_i : r_i]$ 均为回文串。

在回文树中，每个节点 x 代表的字符串有两个前驱，第一个是删除第一个和最后一个字符得到的父亲节点 $fa(x)$，第二个是最长前/后缀所代表的节点 $f(x)$。

对于每个回文串 $s[l : r]$，其包含的回文子串 $s[x : y]$ 都可以通过如下的方式不重不漏地访问：先跳若干步 fa，直到 $l == x$ 或 $r == y$。此时 $s[x : y]$ 一定是 $s[l : r]$ 的前缀或者后缀，所以再连续跳若干步最长前缀或者跳若干步最长后缀即可（此步骤中不能既跳前缀又跳后缀，否则将不满足 $l == x$ 或 $r == y$，此时得到的串将会被重复访问）。

我们设 $ff(x)$ 表示一定选节点 x 所代表的字符串，且该字符串是套娃的最外层的方案数。那么枚举其包含的每个回文子串 y，可得到 $ff(x) = (\sum ff(y)) + 1$。简化方法即分别记录两种前驱的前缀和进行运算，详见以下代码。最后枚举右端点，将当前右端点所对应的所有回文串作为最外层套娃的方案数计入答案中即可。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e6+5;
const ll mod = 998244353;

char s[N];
int n;
int ans;

struct PalindromesTree{
   int tot,last,l[N],son[N][26],c[N],f[N];
   ll ff[N];
   ll pref[N],prefa[N];

   int newnode(int x){
      l[++tot]=x;
      return tot;
   }

   void init(){
      tot=-1;last=0;
      f[newnode(0)]=1;
      f[newnode(-1)]=0;
   }

   int get(int x,int n){
      while(s[n]!=s[n-l[x]-1])x=f[x];
      return x;
   }

   void add(int x,int i){
      int y=get(last,i);
      if(!son[y][x]){
```

```
35          int now=newnode(l[y]+2),ne=get(f[y],i);
36          f[now]=son[ne][x];
37          son[y][x]=now;
38          //y是删去左右第一个字符的父亲节点
39          //f[now]是最长回文前/后缀代表的节点
40          //ff[now]即now作为最外层套娃且必选now的方案数
41          //pref是f作为前驱的ff前缀和
42          //prefa是fa作为前去的ff前缀和
43          ff[now]=(prefa[y]+1)%mod;
44          pref[now]=(pref[f[now]]+ff[now])%mod;
45          prefa[now]=(prefa[y]+2*pref[f[now]]+ff[now])%mod;
46          //在跳若干次父亲节点后，第一次跳f时，有跳前缀和跳后缀两种方案，所以此处要*2
47       }
48       last=son[y][x];
49       (ans+=pref[son[y][x]])%=mod;
50    }
51
52    void solve(){
53       for(int i=1;i<=n;++i)add(s[i]-'a',i);
54    }
55 }T;
56
57 int main(){
58    scanf("%s",s+1);
59    n=strlen(s+1);
60    T.init();
61    T.solve();
62    printf("%d\n",ans);
63 }
```

4.4 后缀自动机

4.4.1 半前缀计数

字符串的半前缀即前缀删除一个子串后得到的字符串，可以用 $s[1...i]+s[j...k]$ 表示。

问题即给定一个长度为 $n(n \leq 10^6)$ 的字符串，求其半前缀的个数。

对于这种本质不同的字符串计数问题，通常都是将某个串在它第一次或最后一次出现时统计。

本题中，如果将每个字符串在第一次出现的前缀中（即尽可能缩短用到的前缀长度），那么看起来就不太能做。但是如果将每一个串在最后一次出现的前缀中统计，就会非常简单：对于一个串，假设它可以表示为 $s[1...i]+s[j...k]$ 和 $s[1...i+a]+s[j+a...k]$，那么有 $s_{i+1}=s_j$。并且，如果它不能表示为 $s[1...i+1]+s[j+1...k]$ 的形式，必然也不能表示成 $s[1...i+a]+s[j+a...k], a>0$，而且有 $s_{i+1}\ s_j$。这也就是说，对于一个串 $t=s[1...i]+s[j...k]$，它是最后一次出现在某个前缀中，当且仅当 $s_{i+1}\ s_j$。所以我们要做的事情实际上就是对于每个后缀 $s[i+1...n]$，求出它包含的本质不同子串个数，减掉以 s_{i+1} 开头的不同子串个数即可。这个操作可以用 SAM 简单维护。

4.5 序列自动机

4.5.1 序列自动机的基本操作

序列自动机即一种建立在序列上的有穷自动机。有字符集，状态集合，起始状态，接收状态集合，转移函数五个要素。

其基本操作包括以下三种：

1. 统计一个串本质不同的子序列的个数。(复杂度为 $O(n*|S|)$)
2. 查找一个子序列是否在该串中出现过。(复杂度为 $O(n*|S|)$)
3. 找到两个串所有的公共子序列。(复杂度为 $O(n*n*|S|)$)

这三种操作的状态集合即位置集合 0 n，起始状态即位置 0，接收状态集合为所有状态，转移函数和第一种操作的具体实现如下：

```
void prepare(){
   for(int i=n;i>=1;--i){
      for(int j=0;j<26;++j)nex[i][j]=nex[i+1][j];
      nex[i][s[i]-'a']=i;
   }
}

int dfs(int x){
   if(vis[x])return f[x];
   f[x]=1;
   for(int i=0;i<26;++i)
   if(nex[x+1][i]){
      f[x]+=dfs(nex[x+1][i]);
   }
   vis[x]=true;
   return f[x];
}
/*
Sample "abab":
Empty
a
b
ab
ba
aa
bb
aba
bab
aab
abb
abab
*/
```

4.5.2 有特殊限制 (相邻有 1 才能删 1) 的 01 子序列计数

给你一个长度为 n 的 01 串 s。你可以进行不超过 $n-1$ 次操作。每次操作，你可以选择一个位置 $i(1\ i<|s|)$，令 $s_i:=max(s_i,s_{i+1})$，然后删掉 s_{i+1}。删掉后，左右两边会自动拼接起来，并

且 s 的长度会减 1。请求出，有多少种不同的串，能通过对 s 进行不超过 $n-1$ 次操作得到。答案对 10^9+7 取模。

首先我们发现生成的数列一定是原序列的一个子序列，然后又要求本质不同的子序列个数，这就让我们想到序列自动机。

我们先看一下对 a_i, a_{i+1} 操作的几种不同的后果：

$$00 \to 0$$

$$01 \to 1$$

$$10 \to 1$$

$$11 \to 1$$

可以发现，只有当两个 1 的时候才有可能删掉一个 1。

所以我们构造如下的一个自动机：（相比一般的序列自动机，起始状态，接收状态集合，转移函数均有所变化）

字符集：$0, 1$。

状态集合：类似于子序列自动机，序列中每个状态代表一个位置。

起始状态：第一个 1 的位置代表的状态。（因为最后一个 1 不可消除，可参考 101 仔细想想）

接受状态集合：所有 1 的位置代表的状态均为接受状态，其余都不是。（因为最后一个 1 不可消除，可参考 101 仔细想想）

转移函数：设当前状态为 $zeta$，则 $(zeta, 1)$ 为下一个 1 的位置，$(zeta, 0)$ 为下一个 $len[x] = len[] + 1$ 的状态 x（其中 $len[]$ 表示以该状态代表的位置为结尾，连续的 0 的个数，这个转移函数是整个题目的精髓）。比如一个序列 1000110000，现在在位置 4，想在这个后面加一个 0，那么我们必须把位置 $[2, 5]$ 的数全部清掉，然后把位置 $[6, 9]$ 的数加进来才能达到目的（可以仔细想一想）。

我们对一个输入的序列，先把头尾的 0 全部删掉（最后累计进答案里）（根据自动机的构造我们必须这样做），然后可以发现我们要计数的东西其实就是从初始节点开始到任何一个接受状态的路径的个数（此处与序列自动机求本质不同的子序列个数是一样的），由于我们已经保证了任何一个最后可能生成的序列在自动机上遍历的时候一定遍历到它第一次出现的位置（类似于序列自动机），所以本质不同的要求我们也满足了。最后对自动机这个 DAG 跑一遍 DP 就可以了。

对这种要求本质不同的有条件的子序列个数的题，我们构建自动机时要注意：

1. 自动机里的路径代表的子序列有且仅有题目要求的合法子序列。

2. 对于任意一个合法子序列，在自动机里跑一遍后必须是在它第一次出现的位置。

3. 与一般的子序列自动机不同，不一定每个状态都是接受状态。

4.6 Lyndon 分解

4.6.1 Duval 算法

Lyndon 串：对于字符串 x，如果 x 的字典序严格小于 x 的所有后缀的字典序，我们称 x 是简单串，或者 *Lyndon* 串。

近似 *Lyndon* 串：若 x 为 *Lyndon* 串，则 $xxxx$ 为近似 *Lyndon* 串，x 为 x 的前缀。

Lyndon 分解：将一个串 x 分作 $x_1x_2..x_k$，每一个部分都是 *Lyndon* 串，且 x_i x_{i+1}。

定理：

- *Lyndon* 串是循环同构中字典序最小的一个。

- *Lyndon* 分解唯一。

- 两个 *Lyndon* 串 a,b，若 $a<b$，有 ab 为 *Lyndon* 串。

考虑 i 之前的位置都已经被分解为 *Lyndon* 串，用指针 j 指向 i，k 指向 $i+1$。

- 若 $x[j]=x[k]$，加上 $x[k]$ 不会影响 i 开头的串为近似 *Lyndon* 串。所以比较后一位，两个指针都往后移；

- 若 $x[j]<x[k]$，我们不允许 $x_i<x_i+1$，所以包含这一位。这个串是一个整体，所以使 $j=i$，重新开始比较（$x[i:k]$ 是 *Lyndon* 串）；

- 若 $x[j]>x[k]$，就不行了，所以当前串就结束了；但是当前串可能是近似 *Lyndon* 串（*Lyndon* 串的长度为 $k-j$），所以一节一节来就行。

如果用后缀数组实现，可以求出所有后缀的 *Lyndon* 分解，其中 nex_i 表示对第 i 个后缀进行分解，得到的下一个 *Lyndon* 串的第一个位置（当前 *Lyndon* 串的第一个位置即 i）。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e6+5;

int n;
char s[N];
vector<int>ans;

void Lyndon(){
   for(int i=1;i<=n;){
      int j=i,k=i+1;
      while(k<=n&&s[j]<=s[k]){
         if(s[j]==s[k])j++,k++;
         else j=i,k++;
      }
      while(i<=j){
         i+=k-j;
         ans.push_back(i-1);
      }
   }
}

int main(){
   scanf("%s",s+1);
   n=strlen(s+1);
```

```
    Lyndon();
    for(int i=0;i<ans.size();++i)printf("%d ",ans[i]);
}
```

4.6.2 后缀数组

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
const int M = 20;

int n,Q,nex[N];
int sa[N],tong[N],wa[N],wb[N],wv[N],rk[N],h[N];
char s[N];

int cmp(int *r,int a,int b,int l){
    return r[a]==r[b]&&r[a+l]==r[b+l];
}

void da(int n,int m){
    int i,j,p,*x=wa,*y=wb;
    for(i=0;i<m;++i)tong[i]=0;
    for(i=0;i<n;++i)tong[x[i]=s[i]]++;
    for(i=1;i<m;++i)tong[i]+=tong[i-1];
    for(i=n-1;i>=0;--i)sa[--tong[x[i]]]=i;
    for(j=1,p=1;p<n;j<<=1,m=p){
        for(p=0,i=n-j;i<n;++i)y[p++]=i;
        for(i=0;i<n;++i)if(sa[i]>=j)y[p++]=sa[i]-j;
        for(i=0;i<n;++i)wv[i]=x[y[i]];
        for(i=0;i<m;++i)tong[i]=0;
        for(i=0;i<n;++i)tong[wv[i]]++;
        for(i=1;i<m;++i)tong[i]+=tong[i-1];
        for(i=n-1;i>=0;--i)sa[--tong[wv[i]]]=y[i];
        for(swap(x,y),p=1,x[sa[0]]=0,i=1;i<n;++i)
        x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
    }
    return;
}

void calh(int n){
    int i,j,k=0;
    for(i=1;i<=n;i++)rk[sa[i]]=i;
    for(i=0;i<n;h[rk[i++]]=k)
    for(k?k--:0,j=sa[rk[i]-1];s[i+k]==s[j+k];k++);
    return;
}

int main(){
    scanf("%s%d",s,&Q);
    n=strlen(s);
    da(n+1,255);
    calh(n);
```

```
    for(int i=n;i>=1;--i)rk[i]=rk[i-1];
    for(int i=1;i<=n;++i)sa[i]++;
    static int stack[N];
    int top=0;
    for(int i=n;i;--i){
        while(top&&rk[stack[top]]>rk[i])--top;
        if(top)nex[i]=stack[top];
        else nex[i]=n+1;
        stack[++top]=i;
    }
    for(int i=1;i<=n;++i)cout<<nex[i]<<" ";
}
```

4.7 KMP

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6+5;

int f[N];
char s1[N],s2[N];

void getfail(char *s){
    int n=strlen(s);
    for(int i=1;i<n;++i){
        int j=f[i];
        while(j&&s[j]!=s[i])j=f[j];
        f[i+1]=(s[i]==s[j]?j+1:0);
    }
    return;
}

void find(){
    int n1=strlen(s1),n2=strlen(s2);
    int j=0;
    for(int i=0;i<n1;++i){
        while(j&&s2[j]!=s1[i])j=f[j];
        if(s2[j]==s1[i])j++;
        if(j==n2)cout<<i-n2+2<<endl;
    }
}

int main(){
    scanf("%s%s",s1,s2);
    getfail(s2);
    find();
    for(int i=1;i<=strlen(s2);++i)cout<<f[i]<<" ";
}
```

4.8 EXKMP

$f[i]$ 表示为模式串 s 中以 i 为起点的后缀字符串和模式串 s 的最长公共前缀长度。

如果要求模式串 $s1$ 中的每个后缀字符串和模式串 $s2$ 的最长公共前缀长度，则把 $s1$ 接到 $s2$ 后面即可（中间要有分隔符）。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 4e7+5;

int n,m,f[N];
char s1[N],s2[N];

void getfail(char *s,int n){
    int a=0,p=0;
    f[0]=n;
    for(int i=1;i<n;++i){
        if(i>=p||i+f[i-a]>=p){
            p=max(p,i);
            while(p<n&&s[p]==s[p-i])p++;
            f[i]=p-i;
            a=i;
        }
        else f[i]=f[i-a];
    }
}

int main(){
    scanf("%s%s",s1,s2);
    n=strlen(s1);
    m=strlen(s2);
    ll ans1=0,ans2=0;
    getfail(s2,m);
    s2[m]='#';
    for(int i=1;i<=n;++i)s2[m+i]=s1[i-1];
    getfail(s2,m+n+1);
}
```

4.9 Manacher

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;

int n;
char a[N];
int R,s[N<<1],f[N<<1];

void manacher(){
    int p=0;
```

```
    s[0]='?';
    s[++R]='#';
    for(int i=1;i<=n;++i){
        s[++R]=a[i];
        s[++R]='#';
    }
    s[++R]='!';
    for(int i=1;i<=R;++i){
        if(f[p]+p>i)f[i]=min(f[p]+p-i,f[p*2-i]);
        else f[i]=1;
        while(s[i-f[i]]==s[i+f[i]])f[i]++;
        if(i+f[i]>p+f[p])p=i;
    }
}

int main(){
    while(scanf("%s",a+1)!=EOF){
        n=strlen(a+1),R=0;
        manacher();
    }
}
```

4.10 Hash 表

```
const int SIZE = 1e6+5;

struct Hash_Table{

    int cnt;
    int head[SIZE];

    struct nd{
        int ne;
        ll val,p;
    }e[SIZE];

    void insert(ll p,ll val){
        int x=(p|(p>>5))%SIZE;
        e[++cnt].p=p;
        e[cnt].val=val;
        e[cnt].ne=head[x];
        head[x]=cnt;
    }

    ll get(ll p){
        int x=(p|(p>>5))%SIZE;
        for(int i=head[x];i;i=e[i].ne)
        if(e[i].p==p){
            return e[i].val;
        }
        return -1;
    }
```

```
}H;
```

4.11 字符串 Hash

```
const int N = 1e5+5;

char s[N];

struct Hash_String{
    int n;
    int seed,mod;
    int base[N],f[N];

    void init(int _n,int _seed,int _mod){
        n=_n;
        seed=_seed;
        mod=_mod;
        base[0]=1;
        for(int i=1;i<N;++i)base[i]=1ll*base[i-1]*seed%mod;
        for(int i=1;i<N;++i)f[i]=(1ll*f[i-1]*seed+s[i])%mod;
    }

    int get(int l,int r){
        return (f[r]-1ll*f[l]*base[r-l]%mod+mod)%mod;
    }
}H[2][2];
```

4.12 最小表示法

```
string solve(string s){
    int n=s.size();
    ll i=0,j=1,k=0,t;
    while(i<n&&j<n&&k<n){
        t=s[(i+k)%n]-s[(j+k)%n];
        if(!t) k++;
        else {
            if(t>0) i+=k+1;
            else j+=k+1;
            if(i==j) j++;
            k=0;
        }
    }
    int ans=i<j?i:j;
    string rt;
    for(int tt=0;tt<n;tt++){
        rt.push_back(s[(ans+tt)%n]);
    }
    return rt;
}
```

4.13 最大字典序的字符串拼接顺序

给定 n 个字符串，求一种拼接顺序，使得得到的大字符串是所有可能性中字典序最小的。

有一种思路为：先把 n 个字符串按照字典顺序排序，然后将串起来的结果返回。这么做是错误的，比如：B, BA 按照字典排序结果是 B, BA，串起来的大写字符串为 "BBA"，但是字典顺序最小的大写字符串是 "BAB"，所以按照单个字符串的字典顺序进行排序的想法是行不通的。如果要排序，应该按照下文描述的标准进行排序。

假设有两个字符串，分别记为 a 和 b，a 和 b 拼起来的字符串表示为 $a.b$。那么如果 $a.b$ 的字典顺序小于 $b.a$，就把字符串 a 放在前面，否则把字符串 b 放在前面。

每两个字符串之间都按照这个标准进行比较，以此标准排序后，再依次串起来的字符串就是结果。这样做为什么对呢？当然需要证明。

证明的关键步骤是证明这种比较方式具有传递性。

假设有 $a\ b\ c$ 三个字符串，它们有如下关系：

$a.b < b.a$

$b.c < c.b$

如果能够根据上面两式证明出 $a.c < c.a$，说明这种比较方式具有传递性，证明过程如下：

字符串的本质是 K 进制数，比如，只由字符 $'a'$ $'z'$ 组成的字符串其实可以看作 26 进制数。那么字符串 $a.b$ 这个数可以看作 a 是它的高位，b 是低位，即 $a.b = a * K\ b\ + b$。为了让证明过程便于阅读，我们把 $K\ b$ 记为 $k(b)$。则原来的不等式可化简为：

$a.b < b.a => a * k(b) + b < b * k(a) + a$ 不等式 1

$b.c < c.b => b * k(c) + c < c * k(b) + b$ 不等式 2

现在要证明 $a.c < c.a$，即证明 $a * k(c) + c < c * k(a) + a$。

不等式 1 的左右两边同时减去 b，再乘以 c，变为 $a * k(b) * c < b * k(a) * c + a * c - b * c$。不等式 2 的左右两边同时减去 b，再乘以 a，变为 $b * k(c) * a + c * a - b * a < c * k(b) * a$。所以有如下不等式：

$b * k(c) * a + c * a - b * a < c * k(b) * a == a * k(b) * c < b * k(a) * c + a * c - b * c$

$=> b * k(c) * a + c * a - b * a < b * k(a) * c + a * c - b * c$

$=> b * k(c) * a - b * a < b * k(a) * c - b * c$

$=> a * k(c) - a < c * k(a) - c$

$=> a * k(c) + c < c * k(a) + a$

即 $a.c < c.a$，传递性证明完毕。

证明传递性后，还需要证明通过这种比较方式排序后，如果交换任意两个字符串的位置所得到的总字符串，将拥有更大的字典顺序。此处证明较简单，故省略。

总结：多利用进制的思想处理字符串问题；如 $a.b < b.a$，则同样可得 $a^{\infty} < b^{\infty}$，证明较为简单。

5 图论

5.1 连通分量

5.1.1 边双连通分量

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;

int n,m,head[N];
pair<int,int>g[N];
int tot,dfn[N],low[N],bel[N];
struct nd{
    int ne,to;
}e[N<<1];

void in(int x,int y){
    static int cnt=1;
    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
}

void tj(int x,int f=-1){
    static int totw,q[N];
    static bool inq[N];
    dfn[x]=low[x]=++totw;
    q[++q[0]]=x;inq[x]=1;
    for(int i=head[x];i;i=e[i].ne)
    if(i!=f){
        int y=e[i].to;
        if(!dfn[y])tj(y,i^1),low[x]=min(low[x],low[y]);
        else if(inq[y])low[x]=min(low[x],dfn[y]);
    }
    if(dfn[x]==low[x]){
        int y;++tot;
        do{
            y=q[q[0]--];
            inq[y]=0;
            bel[y]=tot;
        }while(y!=x);
    }
}

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1,x,y;i<=m;++i){
        scanf("%d%d",&x,&y);
        g[i]=make_pair(x,y);
        in(x,y);in(y,x);
    }
    for(int i=1;i<=n;++i)if(!dfn[i])tj(i);
    for(int i=1;i<=m;++i){
```

```
48        if(bel[g[i].first]==bel[g[i].second])continue;
49    }
50 }
```

5.1.2 点双连通分量

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 1e5+5;
5
6  int n,m,head[N];
7  vector<int>g[N];
8  int tot,dfn[N],low[N];
9  bool iscut[N];
10 struct nd{
11    int ne,to;
12 }e[N<<1];
13
14 void in(int x,int y){
15    static int cnt=1;
16    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
17 }
18
19 void tj(int x,int f=-1){
20    static int totw,q[N],inq[N];
21    dfn[x]=low[x]=++totw;
22    int c=0;
23    for(int i=head[x];i;i=e[i].ne)
24    if(e[i].to!=f){
25       int y=e[i].to;
26       if(!dfn[y]){
27          q[++q[0]]=i;
28          tj(y,x);c++;
29          if(low[y]>=dfn[x]){
30             int t;++tot;
31             iscut[x]=1;
32             do{
33                t=q[q[0]--];
34                if(inq[e[t].to]!=tot)inq[e[t].to]=tot,g[tot].push_back(e[t].to);
35                if(inq[e[t^1].to]!=tot)inq[e[t^1].to]=tot,g[tot].push_back(e[t^1].
                      to);
36             }while(t!=i);
37          }
38          low[x]=min(low[x],low[y]);
39       }
40       else low[x]=min(low[x],dfn[y]);
41    }
42    if(f==-1&&c<2)iscut[x]=0;
43 }
44
45 int main(){
```

```
    scanf("%d%d",&n,&m);
    for(int i=1,x,y;i<=m;++i){
        scanf("%d%d",&x,&y);
        in(x,y);in(y,x);
    }
    for(int i=1;i<=n;++i)if(!dfn[i])tj(i);
}
```

5.1.3 强连通分量

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;

int n,m,head[N];
pair<int,int>g[N];
int tot,dfn[N],low[N],bel[N];
struct nd{
    int ne,to;
}e[N<<1];

void in(int x,int y){
    static int cnt;
    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
}

void tj(int x){
    static int totw,q[N];
    static bool inq[N];
    dfn[x]=low[x]=++totw;
    q[++q[0]]=x;inq[x]=1;
    for(int i=head[x];i;i=e[i].ne){
        int y=e[i].to;
        if(!dfn[y])tj(y),low[x]=min(low[x],low[y]);
        else if(inq[y])low[x]=min(low[x],dfn[y]);
    }
    if(low[x]==dfn[x]){
        int y;++tot;
        do{
            y=q[q[0]--];
            inq[y]=0;
            bel[y]=tot;
        }while(y!=x);
    }
}

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1,x,y;i<=m;++i){
        scanf("%d%d",&x,&y);
        g[i]=make_pair(x,y);
```

```
        in(x,y);
    }
    for(int i=1;i<=n;++i)if(!dfn[i])tj(i);
    for(int i=1;i<=m;++i){
        if(bel[g[i].first]==bel[g[i].second])continue;
    }
}
```

5.2 网络流

5.2.1 网络最大流

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
const int inf = 0x7fffffff;

struct nd{
    int ne,to,w;
}e[N<<1];

int n,m,S,T,ans,cnt=1,head[N],d[N];

void in(int x,int y,int w){
    e[++cnt].to=y;e[cnt].w=w;e[cnt].ne=head[x];head[x]=cnt;
    e[++cnt].to=x;e[cnt].w=0;e[cnt].ne=head[y];head[y]=cnt;
}

bool bfs(){
    queue<int>q;
    for(int i=1;i<=n/*T*/;++i)d[i]=0;
    q.push(S);d[S]=1;
    while(!q.empty()){
        int x=q.front();q.pop();
        for(int i=head[x];i;i=e[i].ne){
            int y=e[i].to;
            if(!d[y]&&e[i].w>0){
                d[y]=d[x]+1;
                q.push(y);
            }
        }
    }
    return d[T]!=0;
}

int dinic(int x,int mn){
    if(x==T||!mn)return mn;
    int flow=0;
    for(int i=head[x];i;i=e[i].ne){
        int y=e[i].to,tmpf;
        if(d[y]==d[x]+1&&(tmpf=dinic(y,min(e[i].w,mn)))>0){
            e[i].w-=tmpf;e[i^1].w+=tmpf;
```

```
            mn-=tmpf;flow+=tmpf;
            if(!mn)break;
        }
    }
    if(!flow)d[x]=0;
    return flow;
}

int main(){
    scanf("%d%d%d%d",&n,&m,&S,&T);
    for(int i=1,x,y,w;i<=m;++i){
        scanf("%d%d%d",&x,&y,&w);
        in(x,y,w);
    }
    while(bfs())ans+=dinic(S,inf);
    printf("%d\n",ans);
}
```

5.2.2 最小费用最大流

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5+5;

int n,m,S,T,head[N],pre[N],d[N],cnt=1;
ll ans1,ans2;

struct nd{
    int ne,to,w,f;
}e[N];

void in(int x,int y,int w,int f){
    ++cnt;e[cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;e[cnt].w=w;e[cnt].f=f;
    ++cnt;e[cnt].to=x;e[cnt].ne=head[y];head[y]=cnt;e[cnt].w=0;e[cnt].f=-f;
}

bool spfa(){
    for(int i=1;i<=n/*T*/;++i)d[i]=2e9;
    static bool inq[N];
    static queue<int>q;
    for(int i=1;i<=n/*T*/;++i)inq[i]=0;
    d[S]=0;
    q.push(S);inq[S]=1;
    while(!q.empty()){
        int x=q.front();q.pop();inq[x]=0;
        for(int i=head[x];i;i=e[i].ne){
            int y=e[i].to;
            if(d[y]>d[x]+e[i].f&&e[i].w>0){
                d[y]=d[x]+e[i].f;
                pre[y]=i;
                if(!inq[y])inq[y]=1,q.push(y);
```

```
            }
        }
    }
    return d[T]!=2e9;
}

void dinic(){
    int flow=2e9;
    for(int i=pre[T];i;i=pre[e[i^1].to])flow=min(flow,e[i].w);
    for(int i=pre[T];i;i=pre[e[i^1].to])e[i].w-=flow,e[i^1].w+=flow;
    ans1+=flow;
    ans2+=flow*d[T];
}

int main(){
    cin>>n>>m>>S>>T;
    for(int i=1,x,y,w,f;i<=m;++i)
    cin>>x>>y>>w>>f,in(x,y,w,f);
    while(spfa())dinic();
    cout<<ans1<<" "<<ans2<<endl;
}
```

5.2.3 二分图最大独立集方案

从左边所有的的未匹配点出发，走一条未匹配边-> 匹配边-> 未匹配边... 的路径，并将所有经过的点打上标记 (需要注意记忆化，而且记忆化后，由于路径实际可以重复，所以每次在找路径时应找所有可行路径，而不是只找一条路径)。所有左边的标记点和右边的未标记点构成了最大独立集。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 5e5+5;
const int inf = 0x7fffffff;

int n,S,T,a[N],d[N],col[N],head[N],cnt=1;
vector<int>g[N],v[N],vis[N];
bool mk[N],pi[N];

struct nd{
    int ne,to,w;
}e[N<<1];

void in(int x,int y,int w){
    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;e[cnt].w=w;
    e[++cnt].to=x;e[cnt].ne=head[y];head[y]=cnt;e[cnt].w=0;
}

bool bfs(){
    static queue<int>q;
    for(int i=1;i<=T;++i)d[i]=0;
    q.push(S);d[S]=1;
```

```
24      while(!q.empty()){
25          int x=q.front();q.pop();
26          for(int i=head[x];i;i=e[i].ne){
27              int y=e[i].to;
28              if(!d[y]&&e[i].w>0){
29                  d[y]=d[x]+1;q.push(y);
30              }
31          }
32      }
33      return d[T]!=0;
34
35  }
36
37  int dinic(int x=S,int mn=inf){
38      if(x==T||!mn)return mn;
39      int flow=0;
40      for(int i=head[x];i;i=e[i].ne){
41          int maxf,y=e[i].to;
42          if(d[y]==d[x]+1&&(maxf=dinic(y,min(mn,e[i].w)))>0){
43              mn-=maxf;flow+=maxf;
44              e[i].w-=maxf;e[i^1].w+=maxf;
45              if(!mn)break;
46          }
47      }
48      if(!flow)d[x]=0;
49      return flow;
50  }
51
52  void dfs(int x,int c){
53      col[x]=c;
54      for(int i=0;i<g[x].size();++i)
55      if(!col[g[x][i]]){
56          int y=g[x][i];
57          dfs(y,3-c);
58      }
59  }
60
61  void walk(int x){
62      mk[x]=1;
63      for(int i=0;i<v[x].size();++i)
64      if(!vis[x][i]){
65          vis[x][i]=1;
66          walk(v[x][i]);
67      }
68  }
69
70  int main(){
71      scanf("%d",&n);
72      for(int i=1;i<=n;++i)scanf("%d",&a[i]);
73      for(int i=1;i<=n;++i)
74      for(int j=i+1;j<=n;++j){
75          int t=a[i]^a[j];
76          if((t&-t)==t)g[i].push_back(j),g[j].push_back(i);
```

```
77      }
78      for(int i=1;i<=n;++i)if(!col[i])dfs(i,1);
79      S=n+1;T=n+2;
80      for(int i=1;i<=n;++i)col[i]--;
81      for(int i=1;i<=n;++i)
82      if(!col[i])in(S,i,1);
83      else in(i,T,1);
84      for(int i=1;i<=n;++i)
85      if(!col[i]){
86          for(int j=0;j<g[i].size();++j)
87          in(i,g[i][j],1);
88      }
89      int ans=0;
90      while(bfs())ans+=dinic();
91      printf("%d\n",n-ans);
92      for(int x=1;x<=n;++x)
93      if(!col[x]){
94          static int c=0;
95          for(int i=head[x];i;i=e[i].ne)
96          if(e[i].to!=S){
97              int y=e[i].to;
98              if(!e[i].w)pi[x]=1,v[y].push_back(x),vis[y].push_back(0);
99              else v[x].push_back(y),vis[x].push_back(0);
100         }
101     }
102     for(int i=1;i<=n;++i)if(!col[i]&&!pi[i])walk(i);
103     for(int i=1;i<=n;++i)if(col[i]^mk[i])printf("%d ",a[i]);
104 }
```

5.2.4 上下界可行流

先考虑无源汇的情况。将每条边的权值都设为 l_i，但是此时图中不满足每个点流出和流入相等的限制，所以对于每个点，如果它的流入大于流出，由超级源点 S 向其连一条流入-流出的边，表示流出还应增加多少；如果它的流出大于流入，由其向超级汇点 T 连一条流出-流入的边，表示流入还应增加多少，对于原来的每条边建成权值为 $r_i - l_i$ 的边，表示最多可以让起点的流出，终点的流入增加多少。然后再跑最大流。如果每个点在调整后都能满足流入 = 流出的限制 (显然 S 连出去的边的总权值等于连向 T 的边的总权值，所以限制等价于这些边都能流满)，则存在可行流。如果原图中有源汇，则为了让每个点仍保持流入等于流出的限制，可以先建一条由汇点到源点的上下界分别为 0 和 inf 的边，转换成无源汇的问题。(此时源点到汇点的最大流就等于这条边的流量)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 1e5+5;
5  const int inf = 0x7fffffff;
6
7  int Case;
8  int n,m,S,T,cnt;
9  int head[N],d[N];
```

```
10 int fin[N],fout[N],ide[N];
11 struct nd{
12    int ne,to,w;
13 }e[N<<1];
14 struct Edge{
15    int x,y,l,r;
16 }a[N];
17
18 void in(int x,int y,int w){
19    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;e[cnt].w=w;
20    e[++cnt].to=x;e[cnt].ne=head[y];head[y]=cnt;e[cnt].w=0;
21 }
22
23 bool bfs(){
24    static queue<int>q;
25    for(int i=1;i<=T;++i)d[i]=0;
26    q.push(S);d[S]=1;
27    while(!q.empty()){
28       int x=q.front();q.pop();
29       for(int i=head[x];i;i=e[i].ne){
30          int y=e[i].to;
31          if(!d[y]&&e[i].w>0){
32             d[y]=d[x]+1;
33             q.push(y);
34          }
35       }
36    }
37    return d[T]!=0;
38 }
39
40 int dinic(int x=S,int mn=inf){
41    if(x==T||!mn)return mn;
42    int flow=0;
43    for(int i=head[x];i;i=e[i].ne){
44       int y=e[i].to,maxf;
45       if(d[y]==d[x]+1&&(maxf=dinic(y,min(mn,e[i].w)))>0){
46          e[i].w-=maxf;e[i^1].w+=maxf;
47          mn-=maxf;flow+=maxf;
48          if(!mn)break;
49       }
50    }
51    if(!flow)d[x]=0;
52    return flow;
53 }
54
55 int main(){
56    scanf("%d",&Case);
57    while(Case--){
58       scanf("%d%d",&n,&m);
59       for(int i=1;i<=n+2;++i)head[i]=fin[i]=fout[i]=0;
60       cnt=1;
61       for(int i=1;i<=m;++i){
62          scanf("%d%d%d%d",&a[i].x,&a[i].y,&a[i].l,&a[i].r);
```

```
63          in(a[i].x,a[i].y,a[i].r-a[i].l);
64          fin[a[i].y]+=a[i].l;
65          fout[a[i].x]+=a[i].l;
66          ide[i]=cnt;
67       }
68       S=n+1;T=n+2;
69       int Maxflow=0;
70       for(int i=1;i<=n;++i)
71       if(fin[i]>=fout[i]){
72          in(S,i,fin[i]-fout[i]);
73          Maxflow+=fin[i]-fout[i];
74       }
75       else in(i,T,fout[i]-fin[i]);
76       //in(t,s,inf);
77       while(bfs())Maxflow-=dinic();
78       if(Maxflow!=0)puts("NO");
79       else{
80          puts("YES");
81          for(int i=1;i<=m;++i)printf("%d\n",a[i].l+e[ide[i]].w);
82       }
83    }
84 }
```

5.2.5 有源汇上下界最大最小流

最大流：得到有源汇上下界可行流后，在删去超级源点，超级汇点，汇点到源点边权为 inf 的边之后，再在残余网络上跑源点到汇点的最大流。最小流：得到有源汇上下界可行流后，在删去超级源点，超级汇点，汇点到源点边权为 inf 的边之后，再在残余网络上跑汇点到源点的最大流。因为在上下界最小流中，我们需要尽可能缩减残余网络中所有边的权值，需要找残余网络中源点到汇点的最小流，这等价于汇点到源点的最大流。(此时的最小流不为 0!! 因为删点后所有点都不一定满足流入等于流出的限制，所以即使是在最小流中，也可能有边不与源汇相连(流满的边视为不存在) 且有流量)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 1e5+5;
5  const int inf = 0x7fffffff;
6
7  int Case;
8  int n,m,S,T,cnt,tot;
9  int head[N],d[N],ide[N];
10 ll fin[N],fout[N],we[N];
11 struct nd{
12    int ne,to;ll w;
13 }e[N<<1];
14
15 void in(int x,int y,int w){
16    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;e[cnt].w=w;
17    e[++cnt].to=x;e[cnt].ne=head[y];head[y]=cnt;e[cnt].w=0;
```

```
}

bool bfs(){
    static queue<int>q;
    for(int i=1;i<=n+m+4;++i)d[i]=0;
    q.push(S);d[S]=1;
    while(!q.empty()){
        int x=q.front();q.pop();
        for(int i=head[x];i;i=e[i].ne){
            int y=e[i].to;
            if(!d[y]&&e[i].w>0){
                d[y]=d[x]+1;
                q.push(y);
            }
        }
    }
    return d[T]!=0;
}

ll dinic(int x=S,ll mn=inf){
    if(x==T||!mn)return mn;
    ll flow=0;
    for(int i=head[x];i;i=e[i].ne){
        int y=e[i].to;
        ll maxf;
        if(d[y]==d[x]+1&&(maxf=dinic(y,min(mn,e[i].w)))>0){
            e[i].w-=maxf;e[i^1].w+=maxf;
            mn-=maxf;flow+=maxf;
            if(!mn)break;
        }
    }
    if(!flow)d[x]=0;
    return flow;
}

void add(int x,int y,int l,int r){
    fin[y]+=l;
    fout[x]+=l;
    in(x,y,r-l);
}

int main(){
    while(scanf("%d%d",&n,&m)!=EOF){
        S=n+m+1;T=n+m+2;
        for(int i=1;i<=n+m+4;++i)head[i]=fin[i]=fout[i]=0;
        cnt=1;tot=0;
        for(int i=1,val;i<=m;++i){
            scanf("%d",&val);
            add(S,i,val,inf);
        }
        for(int i=1,c,d;i<=n;++i){
            scanf("%d%d",&c,&d);
            add(m+i,T,0,d);
```

```
71          for(int j=1,x,l,r;j<=c;++j){
72              scanf("%d%d%d",&x,&l,&r);x++;
73              add(x,m+i,l,r);
74              ++tot;
75              ide[tot]=cnt;
76              we[tot]=l;
77          }
78      }
79      S=n+m+3;T=n+m+4;
80      ll Maxflow=0;
81      for(int i=1;i<=n+m+2;++i)
82      if(fin[i]>=fout[i]){
83          in(S,i,fin[i]-fout[i]);
84          Maxflow+=fin[i]-fout[i];
85      }
86      else in(i,T,fout[i]-fin[i]);
87      in(n+m+2,n+m+1,inf);
88      int idrev=cnt;
89      while(bfs())Maxflow-=dinic();
90      if(Maxflow!=0)puts("-1");
91      else{
92          Maxflow=e[idrev].w;
93          e[idrev].w=e[idrev^1].to=0;
94          for(int i=head[S];i;i=e[i].ne){
95              e[i].w=e[i^1].w=0;
96          }
97          for(int i=head[T];i;i=e[i].ne){
98              e[i].w=e[i^1].w=0;
99          }
100         S=n+m+1;T=n+m+2;//swap(S,T);
101         while(bfs())Maxflow+=dinic();
102         printf("%d\n",Maxflow);
103         for(int i=1;i<=tot;++i)printf("%d\n",we[i]+e[ide[i]].w);
104     }
105     printf("\n");
106 }
107 }
```

5.2.6 用最小费用流中-inf 表示必选来实现上下界网络流

给定一个二分图，求一种最小价值的匹配，使得每个点的匹配次数都在 L_i, R_i 之间。显然这个可以通过上下界网络流实现。但此处要求的是最小费用流，而不是最小费用最大流，所以需要动态加边枚举流量来实现，而上下界网络流很难在残余网络上动态加边（如果每次都重构图的话会 TLE），所以此处需要用一个小 $trick$ 加上最小费用流来实现。该 $trick$ 为对于每个点，都将其向 T 连的边（或者 S 向其连的边）分为两种，一种流量为 L_i，费用为 $-inf$，第二种流量为 $R_i - L_i$，费用为 0。跑完费用流后再将答案加上 L_i 的和 $*inf$ 即可。由于第一种边费用极小，所以要优先选，当枚举到某一总流量时，如果此时即使优先选第一类便也没有选完，那么得到的答案一定大于 inf，则这种情况代表无解。（用这样的方法可以代替上下界网络流，但是费用流速度不如 $Dinic$；此题中需要在残余网络上动态加边，因此用这种方法比较方便）

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5+5;

int n,m,S,T,head[N],pre[N],d[N],cnt=1;
ll ans=1e9,ans1,ans2;

struct nd{
    int ne,to,w,f;
}e[N*20];

void in(int x,int y,int w,int f){
    ++cnt;e[cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;e[cnt].w=w;e[cnt].f=f;
    ++cnt;e[cnt].to=x;e[cnt].ne=head[y];head[y]=cnt;e[cnt].w=0;e[cnt].f=-f;
}

bool spfa(){
    for(int i=1;i<=T;++i)d[i]=2e9;
    static bool inq[N];
    static queue<int>q;
    for(int i=1;i<=T;++i)inq[i]=0;
    d[S]=0;
    q.push(S);inq[S]=1;
    while(!q.empty()){
        int x=q.front();q.pop();inq[x]=0;
        for(int i=head[x];i;i=e[i].ne){
            int y=e[i].to;
            if(d[y]>d[x]+e[i].f&&e[i].w>0){
                d[y]=d[x]+e[i].f;
                pre[y]=i;
                if(!inq[y])inq[y]=1,q.push(y);
            }
        }
    }
    return d[T]!=2e9;
}

void dinic(){
    int flow=2e9;
    for(int i=pre[T];i;i=pre[e[i^1].to])flow=min(flow,e[i].w);
    for(int i=pre[T];i;i=pre[e[i^1].to])e[i].w-=flow,e[i^1].w+=flow;
    ans1+=flow;
    ans2+=flow*d[T];
}

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1,x,y,w;i<=m;++i){
        scanf("%d%d%d",&x,&y,&w);
        in(x,y+n,1,w);
    }
```

```
    S=2*n+2;
    T=2*n+3;
    for(int i=1;i<=n;++i){
        in(S,i,1,-1e9);
        in(S,i,1e9,0);
        in(i+n,T,1,-1e9);
        in(i+n,T,1e9,0);
    }
    S--;
    in(S,S+1,n-1,0);
    for(int i=1;i<=n+1;++i){
        in(S,S+1,1,0);
        while(spfa())dinic();
        ans=min(ans,ans2+(ll)2e9*n);
    }
    if(ans>=1e9)printf("NIE\n");
    else printf("%d\n",ans);
}
```

5.3 最小生成树

5.3.1 环中的最大边可以不在最小生成树中

对于环上的任意一条最大边，都存在最小生成树不包含它。

最小生成树不可能包含一个环上所有的最大边。

证明很简单：反证法。

若包含，则随意去掉一个环上的最大边 (x,y)，此时整棵树断成两部分，x,y 在不同的部分。由于是个环，那么环上一定有一条不包含 (x,y) 的 $x \to y$ 路径，连接两个部分，这条路径上显然有连接两部分的另一条边，选上它得到另一棵生成树。这条边边权严格小于 (x,y) 的权，所以原树不可能是最小生成树，推出矛盾，得证。

5.3.2 最小生成树的边权集合唯一

假设最小生成树 T 和 T' 按照权重排序后的边为 $(e_0,e_1,...)$ 和 $(e'_0,e'_1,...)$, 引入记号 E_i 和 E'_i, 分别为 T 和 T' 的第 0 到第 i 条边的集合

假设 e_k 和 e'_k 为第一对不是同一条边的位置，假设 e_k 为边 (u,v), e'_k 为 (u',v')，且此时 (u,v) 不在 T' 中

证明第一部分：

在 T' 中必然有一条 $u->v$ 的路径，记为 $P'(u,v)$：

1. $P'(u,v)$ 不可能只包含 E'_{k-1} 中的边：因为 k 为第一对不同的边故 $E_{k-1}=E'_{k-1}$，如果 $u->v$ 只有 E'_{k-1} 的边则在 T 中形成环路。

1. 假设边 e'_j 为 $\{P'(u,v)-E'_{k-1}\}$ 中一条，则 $weight(e'_j) <= weight(e_k)$ （否则可以在 T' 中把 e'_j 去掉换成 e_k 得到更小的生成树，与 T' 为最小生成树的前提矛盾），且 $j >= k$，故 $weight(e'_k) <= weight(e_k)$

结论 1：对称的可以推到 $weight(e_k) <= weight(e'_k)$, 因此 e_k 和 e'_k 的权重必然相等；

结论 2：任取边 e' 属于 $\{P'(u,v)-E'_{k-1}\}$, $weight(e')=weight(e_k)=weight(e'_k)$

证明第二部分：

依次考虑每一个不相等的位置

1. 设 $weight(e_k)$ <= $weight(e'_k)$，如果 T' 中也有 e_k，则此时用不到以上结论，直接在 T' 中交换 e_k 和 e'_k（因为 $E_{k-1}=E'_{k-1}$，此时 T' 中 e_k 一定在 e'_k 后面，由此同样可知 $weight(e_k)=weight(e'_k)$）2. 如果 T' 中不包含 e_k，可以将 $P'(u,v)-E'_{k-1}$ 中任意一条替换为 e_k，然后转到第 1 步

由以上结论可知，T' 可以在不改变边权集合的情况下变化为 T，所以 T' 与 T 的边权集合相同，结论得证

这个命题同样说明，如果无向图的边权都不相同，则最小生成树是唯一的。

5.3.3 Boruvka 算法求最小生成树

Boruvka 算法流程：

先对于每个点，选择在所有与之相连的边中，权值最小的边，并将这条边加入到最小生成树中。显然这样连出来的边会形成一个森林，并且连边后连通块个数至少减半。然后我们将每个连通块再看成一个点，重复以上算法即可。时间复杂度 $O(mlogn)$。

例题：

给定一个 n 个节点的完全图，每个节点有个编号 a_i，节点 i 和节点 j 之间边的权值为 a_i xor a_j，求该图的最小生成树的权值和。

从高位往低位贪心。把所有点按当前位为 0 还是 1 分为两类。显然在执行 *Boruvka* 算法时，这两类点内部会先形成联通块，然后这两类点间再连一条尽可能小的连边。因此，只要利用 *Trie* 树得到两类点间的最小边，然后对于两类点内部的连边情况再进行递归处理即可。总复杂度 $O(mlogn)$。

！注意！：对于异或问题，存在单调性的问题，利用数位动态规划的思想根据最高位取值为 0/1 将集合划分为两部分进行分治是很常用的做法。

5.4 欧拉回路

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5+1;

int n,m,head[N];
int tp,d[N],fr[N],to[N],ans[N];
bool vis[N];
struct nd{
   int ne,to,id;
}e[N*4];

void in(int x,int y,int w){
```

```
14      static int cnt=1;
15      e[++cnt].to=y;e[cnt].ne=head[x];e[cnt].id=w;head[x]=cnt;
16  }
17
18  void dfs(int x,int f){
19      for(int &i=head[x];i;i=e[i].ne){
20          if(!vis[i>>1]){
21              vis[i>>1]=1;
22              int now=e[i].id;
23              dfs(e[i].to,x);
24              ans[++ans[0]]=now;
25          }
26      }
27      return;
28  }
29  void dfs1(int x){
30      for(int &i=head[x];i;i=e[i].ne)
31      if(!vis[i]){
32              vis[i]=1;
33              int now=e[i].id;
34              dfs1(e[i].to);
35              ans[++ans[0]]=now;
36      }
37  }
38
39  int main(){
40      scanf("%d",&tp);
41      if(tp==1){//Undirected Graph
42          scanf("%d%d",&n,&m);
43          for(int i=1,x,y;i<=m;++i){
44              scanf("%d%d",&x,&y);
45              in(x,y,i);in(y,x,-i);
46              d[x]++;d[y]++;
47          }
48          for(int i=1;i<=n;++i)if(d[i]&1)return 0*printf("NO");
49          for(int i=1;i<=n;++i)if(head[i]){
50              dfs(i,-1);
51              break;
52          }
53          if(ans[0]!=m)return 0*printf("NO");
54          printf("YES\n");
55          for(int i=ans[0];i>=1;--i)printf("%d ",ans[i]);
56      }
57      else{//directed Graph
58          scanf("%d%d",&n,&m);
59          int tag=0;
60          for(int i=1,x,y;i<=m;++i){
61              scanf("%d%d",&x,&y);
62              in(x,y,i);
63              fr[x]++;to[y]++;
64          }
65          for(int i=1;i<=n;++i)if(fr[i]!=to[i])return 0*printf("NO");
66          for(int i=1;i<=n;++i)
```

```
67        if(head[i]){
68            dfs1(i);
69            break;
70        }
71        if(ans[0]!=m)return 0*printf("NO");
72        printf("YES\n");
73        for(int i=ans[0];i>=1;--i)printf("%d ",ans[i]);
74    }
75 }
```

5.5 仙人掌

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 2e5+5;
5  const int inf = 0x7fffffff;
6
7  int n,m,head[N];
8  vector<int>g[N];
9  int tot,dfn[N],low[N];
10 struct nd{
11     int ne,to;
12 }e[N<<1];
13
14 void in(int x,int y){
15     static int cnt=1;
16     e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
17 }
18
19 void tj(int x,int fa=-1){
20     static int totw,q[N];
21     dfn[x]=low[x]=++totw;
22     for(int i=head[x];i;i=e[i].ne)
23     if(e[i].to!=fa){
24         int y=e[i].to;
25         if(!dfn[y]){
26             q[++q[0]]=i;
27             tj(y,x);
28             if(low[y]>=dfn[x]){
29                 int t;++tot;
30                 do{
31                     t=q[q[0]--];
32                     g[tot].push_back(e[t].to);
33                 }while(t!=i);
34                 if(g[tot].size()==1)g[tot].push_back(e[t^1].to);
35             }
36             low[x]=min(low[x],low[y]);
37         }
38         else if(dfn[y]<dfn[x])q[++q[0]]=i,low[x]=min(low[x],dfn[y]);
39     }
40 }
```

```

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1,x,y;i<=m;++i){
        scanf("%d%d",&x,&y);
        in(x,y);in(y,x);
    }
    tot=n;tj(1);
    for(int i=n+1;i<=tot;++i)
    for(int j=0;j<g[i].size();++i)
    g[g[i][j]].push_back(i);
}
```

5.6 虚树

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e6+5;

int n;
int cnt,head[N];
int dfn[N],st[N],d[N],f[N][20];
int totc,vis[N];
int c[N];

struct nd{
    int ne,to;
}e[N<<1];

void in(int x,int y){
    if(!x||!y||x==y)return;
    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
}

void dfs_init(int x){
    static int totw;
    dfn[x]=++totw;
    for(int i=head[x];i;i=e[i].ne)
    if(e[i].to!=f[x][0]){
        int y=e[i].to;
        d[y]=d[x]+1;
        f[y][0]=x;
        dfs_init(y);
    }
}

bool cmp(int a,int b){
    return dfn[a]<dfn[b];
}

int lca(int x,int y){
```

```
38      if(d[x]<d[y])swap(x,y);
39      for(int i=19;~i;--i)
40      if(d[f[x][i]]>=d[y]){
41          x=f[x][i];
42      }
43      if(x==y)return x;
44      for(int i=19;~i;--i)
45      if(f[x][i]!=f[y][i]){
46          x=f[x][i],y=f[y][i];
47      }
48      return f[x][0];
49  }
50
51  void dfs(int x,int f=-1){
52      c[++c[0]]=x;
53      for(int i=head[x];i;i=e[i].ne)
54      if(vis[e[i].to]!=totc){
55          vis[e[i].to]=totc;
56          /*
57          do sth..
58          */
59          int y=e[i].to;
60          dfs(y,x);
61      }
62  }
63
64  void build(){
65      cnt=0;
66      c[0]=0;
67      ++totc;
68      int k;
69      static int q[N];
70      scanf("%d",&k);
71      for(int i=1;i<=k;++i)scanf("%d",q+i);
72      sort(q+1,q+k+1,cmp);
73      int L=0;
74      st[++L]=1;
75      for(int i=1;i<=k;++i){
76          int x=q[i],l=lca(x,st[L]);
77          while(1){
78              if(dfn[l]>=dfn[st[L-1]]){
79                  in(l,st[L--]);
80                  if(st[L]!=l)st[++L]=l;
81                  break;
82              }
83              in(st[L-1],st[L]),L--;
84          }
85          if(x!=st[L])st[++L]=x;
86      }
87      while(--L)in(st[L],st[L+1]);
88      vis[1]=totc;dfs(1);
89      for(int i=1;i<=c[0];++i)head[c[i]]=0;
90  }
```

```

int main(){
    scanf("%d",&n);
    for(int i=1;i<n;++i){
        int x,y;
        scanf("%d%d",&x,&y);
        in(x,y);
        in(y,x);
    }
    d[1]=1;dfs_init(1);
    for(int j=1;j<20;++j)
    for(int i=1;i<=n;++i)
    f[i][j]=f[f[i][j-1]][j-1];
    for(int i=1;i<=n;++i)head[i]=0;
    int Q;
    scanf("%d",&Q);
    while(Q--)build();
}
```

5.7 2-SAT

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e6+5;

int n,m,head[N],rev[N];
int tot,dfn[N],low[N],bel[N];
struct nd{
    int ne,to;
}e[N<<1];

void in(int x,int y){
    static int cnt;
    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
}

void tj(int x){
    static int totw,q[N];
    static bool inq[N];
    dfn[x]=low[x]=++totw;
    q[++q[0]]=x;inq[x]=1;
    for(int i=head[x];i;i=e[i].ne){
        int y=e[i].to;
        if(!dfn[y])tj(y),low[x]=min(low[x],low[y]);
        else if(inq[y])low[x]=min(low[x],dfn[y]);
    }
    if(dfn[x]==low[x]){
        int y;++tot;
        do{
            y=q[q[0]--];
            inq[y]=0;
```

```
            bel[y]=tot;
        }while(y!=x);
    }
}


int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;++i)rev[i]=i+n,rev[i+n]=i;
    for(int i=1,x,a,y,b;i<=m;++i){
        scanf("%d%d%d%d",&x,&a,&y,&b);
        in(rev[x+a*n],y+b*n);
        in(rev[y+b*n],x+a*n);
    }
    for(int i=1;i<=2*n;++i)if(!dfn[i])tj(i);
    for(int i=1;i<=n;++i)
    if(bel[i]==bel[rev[i]]){
        printf("IMPOSSIBLE\n");
        return 0;
    }
    printf("POSSIBLE\n");
    for(int i=1;i<=n;++i)printf("%d ",bel[i]>bel[rev[i]]);
}
```

5.8 带花树

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 505;


int n,m,head[N];
int tp[N],pre[N],match[N],flower[N];
int Q[N],size;
struct nd{
    int ne,to;
}e[N*N];

void in(int x,int y){
    static int cnt;
    e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
}

int blossom(int x){
    return x==flower[x]?x:flower[x]=blossom(flower[x]);
}

int lca(int x,int y){
    static int C,vis[N];
    ++C;
    for(;x;x=pre[match[x]]){
```

```
        x=blossom(x);
        vis[x]=C;
    }
    for(;y;y=pre[match[y]]){
        y=blossom(y);
        if(vis[y]==C)return y;
    }
}

void group(int x,int S){
    while (x!=S){
        int y=match[x],z=pre[y];
        if (blossom(z)!=S) pre[z]=y;
        if (tp[y]!=1)tp[Q[++size]=y]=1;
        if (tp[z]!=1)tp[Q[++size]=z]=1;
        flower[blossom(x)]=blossom(y);
        flower[blossom(y)]=blossom(z);
        x=z;
    }
}

bool find(int s){
    for(int i=1;i<=n;++i)tp[i]=pre[i]=0,flower[i]=i;
    tp[Q[size=1]=s]=1;
    for(int i=1;i<=size;++i){
        int x=Q[i];
        for(int i=head[x];i;i=e[i].ne){
            int y=e[i].to;
            if(tp[y]==2)continue;
            if(y==match[x])continue;
            if(blossom(x)==blossom(y))continue;
            if(!match[y]){
                pre[y]=x;
                for(int v=y,nxt,u;v;v=nxt){
                    u=pre[v],nxt=match[u];
                    match[u]=v;
                    match[v]=u;
                }
                return 1;
            }
            if(tp[y]==1){
                int LCA=lca(x,y);
                if(blossom(x)!=LCA)pre[x]=y;
                if(blossom(y)!=LCA)pre[y]=x;
                group(x,LCA);
                group(y,LCA);
            }
            else{
                tp[y]=2;pre[y]=x;
                tp[Q[++size]=match[y]]=1;
            }
        }
    }
```

```
    return 0;
}

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1,x,y;i<=m;++i){
        scanf("%d%d",&x,&y);
        in(x,y);in(y,x);
    }
    int ans=0;
    for(int i=1;i<=n;++i)if(!match[i])ans+=find(i);
    printf("%d\n",ans);
    for(int i=1;i<=n;++i)printf("%d ",match[i]);
    printf("\n");
}
```

5.9 支配树

支配树适用于单源有向图。

$semi[x]$ 是半支配点。从 $semi[x]$ 到 x 存在一条路径，掐头去尾（去除 $x, semi[x]$），经过的所有点的 dfn 都大于 $dfn[x]$，且 $semi[x]$ 为所有满足条件的点中 dfn 最小的。

$idom[x]$ 是最近支配点。删除 $idom[x]$ 后，从起点 S 无法到达 x，且 $idom[x]$ 为所有满足条件的点中离 x 最近的。

G 是原图，$revG$ 是反图，$G1$ 是支配树，$G2$ 是半支配树（半支配树应该没用）。

```
#include<bits/stdc++.h>
using namespace std;
const int N = 3e5+5;

int n,m,cnt,totw,S;
int fa[N],dfn[N],pre[N],p[N];
int semi[N],idom[N];
int mn[N];

struct nd{
    int ne,to;
}e[N*4];

struct graph{
    int head[N];

    void in(int x,int y){
        e[++cnt].to=y;e[cnt].ne=head[x];head[x]=cnt;
    }
}G,revG,G1,G2;

int find(int x){
    if(x==fa[x])return x;
    int t=fa[x];
    fa[x]=find(fa[x]);
    if(dfn[semi[mn[t]]]<dfn[semi[mn[x]]])mn[x]=mn[t];
```

```
27      return fa[x];
28  }
29
30  void dfs(int x){
31      dfn[x]=++totw;
32      p[totw]=x;
33      for(int i=G.head[x];i;i=e[i].ne)
34      if(!dfn[e[i].to]){
35          int y=e[i].to;
36          pre[y]=x;
37          dfs(y);
38      }
39  }
40
41  void tarjan(){
42      dfs(S);
43      for(int i=totw;i>=2;--i){
44          int res=n;
45          int x=p[i];
46          for(int j=revG.head[x];j;j=e[j].ne)
47          if(dfn[e[j].to]){
48              int y=e[j].to;
49              find(y);
50              res=min(res,dfn[semi[mn[y]]]);
51          }
52          fa[x]=pre[x];
53          semi[x]=p[res];
54          G2.in(semi[x],x);
55          int y=p[i-1];
56          for(int j=G2.head[y];j;j=e[j].ne){
57              int z=e[j].to;
58              find(z);
59              if(semi[mn[z]]==y)idom[z]=y;
60              else idom[z]=mn[z];
61          }
62      }
63      for(int i=2;i<=totw;++i){
64          int x=p[i];
65          if(idom[x]!=semi[x]){
66              idom[x]=idom[idom[x]];
67          }
68      }
69  }
70
71  int sz[N];
72
73  void dfs1(int x){
74      sz[x]=1;
75      for(int i=G1.head[x];i;i=e[i].ne){
76          dfs1(e[i].to);
77          sz[x]+=sz[e[i].to];
78      }
79  }
```

```

int main(){
    scanf("%d%d",&n,&m);
    S=1;
    for(int i=1,x,y;i<=m;++i){
        scanf("%d%d",&x,&y);
        G.in(x,y);
        revG.in(y,x);
    }
    for(int i=1;i<=n;++i)fa[i]=semi[i]=mn[i]=i;
    tarjan();
    for(int i=1;i<=n;++i)
    if(i!=S){
        G1.in(idom[i],i);
    }
    dfs1(S);
    for(int i=1;i<=n;++i)printf("%d ",sz[i]);
}
```

5.10 点分治

```
//点分治相关DP一定要考虑点分树！！
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5+1;
const ll mod = 1e9+7;

int n,m,cnt,head[N];
int rt,d[N],mx[N],size[N],tot,sum;
bool vis[N];
struct nd{
    int ne,to,w;
}e[N<<1];

void in(int x,int y,int w){
    e[++cnt].to=y;e[cnt].ne=head[x];e[cnt].w=w;head[x]=cnt;
}

int getrt(int x,int f=-1){
    size[x]=1;mx[x]=0;
    for(int i=head[x];i;i=e[i].ne)
    if(e[i].to!=f&&!vis[e[i].to]){
        getrt(e[i].to,x);
        size[x]+=size[e[i].to];
        if(size[e[i].to]>mx[x])
        mx[x]=size[e[i].to];
    }
    if(sum-size[x]>mx[x])mx[x]=sum-size[x];
    if(mx[rt]>mx[x])rt=x;
}

```

```
ll ans[3],c[3],cnt1[3];

int get_a(int x,int f=-1){
    c[d[x]%3]+=d[x];
    cnt1[d[x]%3]++;
    for(int i=head[x];i;i=e[i].ne)
    if(!vis[e[i].to]&&e[i].to!=f){
        d[e[i].to]=d[x]+e[i].w;
        get_a(e[i].to,x);
    }
}

void get(int x,int dis,ll t){
    c[1]=c[2]=c[0]=0;
    cnt1[1]=cnt1[2]=cnt1[0]=0;
    d[x]=dis;
    get_a(x);
    ans[0]+=t*(cnt1[0]*c[0]+cnt1[2]*c[1]+cnt1[1]*c[2]);
    ans[1]+=t*(cnt1[2]*c[2]+cnt1[0]*c[1]+cnt1[1]*c[0]);
    ans[2]+=t*(cnt1[1]*c[1]+cnt1[0]*c[2]+cnt1[2]*c[0]);
}

int dfs(int x){
    get(x,0,1);
    vis[x]=1;
    for(int i=head[x];i;i=e[i].ne)
    if(!vis[e[i].to]){
        get(e[i].to,e[i].w,-1);
        sum=size[e[i].to];rt=0;getrt(e[i].to,0);
        dfs(rt);
    }
}

int main(){
    while(scanf("%d",&n)!=EOF){
        mx[0]=0x3f3f3f3f;
        cnt=0;
        for(int i=1;i<=n;++i)head[i]=vis[i]=0;
        for(int i=0;i<3;++i)ans[i]=0;
        for(int i=1,x,y,w;i<n;++i){
            scanf("%d%d%d",&x,&y,&w);
            in(x+1,y+1,w);
            in(y+1,x+1,w);
        }
        sum=n;rt=0;getrt(1,0);
        dfs(rt);
        for(int i=0;i<3;++i)ans[i]%=mod;
        for(int i=0;i<3;++i)ans[i]=ans[i]*2%mod;
        printf("%lld %lld %lld\n",ans[0],ans[1],ans[2]);
    }
}
```

5.11 KM 算法

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 750+5;
const int inf = 0x3f3f3f3f;

int n,a[N][N];
int wx[N],wy[N];
int link[N],pre[N],slack[N];
bool vis[N];

void bfs(int x){
   int y=0;
   link[0]=x;
   do{
      x=link[y];
      vis[y]=1;
      int nexty;
      int mind=inf;
      for(int i=1;i<=n;++i)
      if(!vis[i]){
         if(slack[i]>wx[x]+wy[i]-a[x][i]){
            slack[i]=wx[x]+wy[i]-a[x][i];
            pre[i]=y;
         }
         if(slack[i]<mind){
            mind=slack[i];
            nexty=i;
         }
      }
      for(int i=0;i<=n;++i)
      if(vis[i]){
         wx[link[i]]-=mind;
         wy[i]+=mind;
      }
      else{
         slack[i]-=mind;
      }
      y=nexty;
   }while(link[y]);
   while(y){
      link[y]=link[pre[y]];
      y=pre[y];
   }
}

ll KM(int nnow){
   n=nnow;
   memset(wx,0,sizeof(wx));
   memset(wy,0,sizeof(wy));
   memset(link,0,sizeof(link));
```

```
    for(int i=1;i<=n;++i){
        memset(vis,0,sizeof(vis));
        memset(pre,0,sizeof(pre));
        memset(slack,0x3f,sizeof(slack));
        bfs(i);
    }
    ll ans=0;
    for(int i=1;i<=n;++i){
        ans+=wx[i]+wy[i];
    }
    return ans;
}

int main(){
    int n,m,k;
    scanf("%d%d%d",&n,&m,&k);
    vector<int>id[N<<1];
    for(int i=1;i<=n;++i)id[i].resize(m+1);
    int cntx=0,cnty=0;
    for(int i=1;i<=n;++i)
    for(int j=1;j<=m;++j)
    if(i+j&1){
        id[i][j]=++cntx;
    }
    else{
        id[i][j]=++cnty;
    }
    for(int i=1,x1,y1,x2,y2,w;i<=k;++i){
        scanf("%d%d%d%d%d",&x1,&y1,&x2,&y2,&w);
        if(x2+y2&1){
            swap(x1,x2);
            swap(y1,y2);
        }
        a[id[x1][y1]][id[x2][y2]]=max(w,a[id[x1][y1]][id[x2][y2]]);
    }
    printf("%lld\n",KM(cnty));

}
```

5.12 图的荫度

给定一个无向连通图，可能有重边，将边集划分为最少的森林（等价于用尽可能少的生成树覆盖所有边），此时森林的个数称为图的荫度。

总复杂度为 $O(m^2)$。

详见 2020 Multi-University Training Contest 10 B。

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

template <typename T> using Vector = std::vector<T>;
template <typename T> using Queue = std::queue<T>;
template <typename T> using Pair = std::array<T, 2>;
```

```
6
7  constexpr int NIL = INT_MAX;
8
9  int n;
10 Vector<Pair<int>> edges;
11 Vector<int> fid;
12 Vector<int> pre;
13
14 class ForestOracle {
15    Vector<int> edgelist;
16    Vector<Vector<int>> incident;
17    Vector<int> parent;
18    Vector<int> peid;
19
20    void dfs(int u, int p, int e = NIL) {
21       if (parent[u] != NIL) return;
22       parent[u] = p;
23       peid[u] = e;
24       for (int eid : incident[u])
25          dfs(edges[eid][0] ^ edges[eid][1] ^ u, u, eid);
26    }
27
28 public:
29    explicit ForestOracle(int eid) :
30       incident(n), parent(n), peid(n), edgelist{eid} {}
31
32    void insert(int eid) {
33       edgelist.push_back(eid);
34    }
35
36    void remove(int eid) {
37       edgelist.erase(std::find(edgelist.begin(), edgelist.end(), eid));
38    }
39
40    void prepare(int r) {
41       for (auto& inc : incident) inc.clear();
42       std::fill(parent.begin(), parent.end(), NIL);
43       std::fill(peid.begin(), peid.end(), NIL);
44       for (int eid : edgelist)
45          for (int u : edges[eid])
46             incident[u].push_back(eid);
47       dfs(r, r);
48    }
49
50    bool get_cycle(Queue<int>& q, int eid) {
51       int u = edges[eid][0], v = edges[eid][1];
52       if (parent[u] == NIL || parent[v] == NIL)
53          return false;
54       Vector<int> eids;
55       for (int vtx : {u, v}) {
56          while (parent[vtx] != vtx && pre[peid[vtx]] == NIL) {
57             eids.push_back(peid[vtx]);
58             vtx = parent[vtx];
```

```
59          }
60      }
61      std::reverse(eids.begin(), eids.end());
62      for (int e : eids) {
63          pre[e] = eid;
64          q.push(e);
65      }
66      return true;
67   }
68};
69
70Vector<ForestOracle> forests;
71
72void augment(int eid) {
73   if (forests.empty()) {
74      fid[eid] = 0;
75      forests.emplace_back(eid);
76      return;
77   }
78   Queue<int> q; q.push(eid);
79
80   fid[eid] = -1; // s.t. (fid[eid] + 1) mod k = 0
81   std::fill(pre.begin(), pre.end(), NIL);
82   while (q.size()) {
83      int u = q.front(); q.pop();
84      int nf = (fid[u] + 1) % forests.size();
85      if (!forests[nf].get_cycle(q, u)) {
86         while (u != eid) {
87            forests[nf].insert(u);
88            forests[fid[u]].remove(u);
89            std::swap(fid[u], nf);
90            u = pre[u];
91         }
92         forests[nf].insert(u);
93         fid[u] = nf;
94         return;
95      }
96   }
97   fid[eid] = forests.size();
98   forests.emplace_back(eid);
99}
100
101int main() {
102   int T; std::cin >> T;
103   while (T--) {
104      forests.clear();
105
106      int m;
107      std::cin >> n >> m;
108      edges.resize(m);
109      fid.resize(m);
110      pre.resize(m);
111      for (int i = 0; i < m; i++) {
```

```
            for (int& u : edges[i]) {
                std::cin >> u;
                u--;
            }
            for (auto& f : forests) f.prepare(edges[i][0]);
            augment(i);
        }
        std::cout << forests.size() << std::endl;
    }
    return 0;
}
```

5.13 无向图最大最小环

无向图最大环:

构造任意一个 dfs 树，图中只有树边和返祖边，则最大环一定由一条返祖边和若干条树边构成。复杂度 $O(n)$

无向图最小环:

$Floyd$ 算法，复杂度 $O(n^3)$

以每个点为根构造一个 bfs 树，得出根到每个点的最短距离，则经过根的最小环一定由一条非树边和这条边两顶点到根的最短路径构成 (不需要关心这两条路径是否可能有交，因为如果有交，删去交集可以得到一个更小的环，所以这个环一定不是最小环，这样求出来的最小环一定为简单环)。复杂度 $O(n^2)$

5.14 三元环, 四元环计数

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;
const ll mod = 1e9+7;

int n,m,id[N],pos[N];
vector<int> g[N],g1[N];

bool cmp(int x,int y){
    return g[x].size()<g[y].size();
}

ll cycle3(){
    static bool vis[N];
    ll ans=0;
    for(int x=1;x<=n;++x){
        for(int y:g1[x])vis[y]=1;
        for(int y:g1[x])
        for(int z:g1[y])
        if(vis[z]){
```

```
            ans++;
        }
        for(int y:g1[x])vis[y]=0;
    }
    return ans;
}

ll cycle4(){
    static int cnt[N];
    ll ans=0;
    for(int x=1;x<=n;++x){
        for(int y:g[x])
        for(int z:g1[y])
        if(pos[z]>pos[x]){
            ans=(ans+(cnt[z]++))%mod;
        }
        for(int y:g[x])
        for(int z:g1[y])
        if(pos[z]>pos[x]){
            cnt[z]=0;
        }
    }
    return ans;
}

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1,x,y;i<=m;++i){
        scanf("%d%d",&x,&y);
        g[x].push_back(y);
        g[y].push_back(x);
    }
    for(int i=1;i<=n;++i)id[i]=i;
    sort(id+1,id+n+1,cmp);
    for(int i=1;i<=n;++i)pos[id[i]]=i;
    for(int x=1;x<=n;++x)
    for(int y:g[x])
    if(pos[x]<pos[y]){
        g1[x].push_back(y);
    }
    cout<<cycle3()<<endl;
}
```

5.15 扫描线解决极大网格图上求联通性相关的问题

对于极大网格图上求联通性相关的问题，通常可以利用扫描线算法来解决问题。

例如给定一个 $n = m = 2 * 10^5$ 的网格图，在其中放置 $k = 10^6$ 个障碍。一个人每次可以走向上下左右四个格子 (不能走出边界)，求有多少个点对 $(x_1, y_1), (x_2, y_2)$ 可以互相到达。此时可以考虑逐行计算连通性与方案数，把一行内连续的非障碍看成线段，那么在考虑计算第 i 行的贡献时，问题转化为两个线段组 (每个线段组内线段不相交) 的交的问题，可以利用尺取法解决。

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6+5;
const int M = 2e5+5;
const int mod = 1e9+7;

int T;
int n,m,k,fa[N*3];
vector<int> g[N*2];

struct nd{
    int x,y;
}a[N];

struct line{
    int l,r;
    int id;
}l[2][M];

struct func{
    int c,f;
    friend func operator + (func a,func b){
        return {(a.c+b.c)%mod,int(((a.f+b.f)%mod+1ll*a.c*b.c%mod)%mod)};
    }
}f[N*3];

int gf(int x){
    return fa[x]==x?x:fa[x]=gf(fa[x]);
}

void work(int x,int y){
    if(!x||!y)return;
    int fa0=gf(l[0][x].id);
    int fa1=gf(l[1][y].id);
    if(fa0!=fa1&&min(l[0][x].r,l[1][y].r)>=max(l[0][x].l,l[1][y].l)){
        fa[fa1]=fa0;
        f[fa0]=f[fa0]+f[fa1];
    }
}

signed main(){
    scanf("%d",&T);
    while(T--){
        scanf("%d%d%d",&n,&m,&k);
        static int q[N*2];
        int top=0;
        for(int i=1;i<=k;++i){
            scanf("%d%d",&a[i].x,&a[i].y);
            if(a[i].x!=1)q[++top]=a[i].x-1;
            q[++top]=a[i].x;
        }
        q[++top]=n;
```

```
53        sort(q+1,q+top+1);
54        top=unique(q+1,q+top+1)-q-1;
55        for(int i=1;i<=top;++i)g[i].clear();
56        for(int i=1;i<=k;++i){
57            int p=lower_bound(q+1,q+top+1,a[i].x)-q;
58            g[p].push_back(a[i].y);
59        }
60        int r[2]={0,0};
61        int cnt_id=0;
62        for(int i=1;i<=top;++i){
63            sort(g[i].begin(),g[i].end());
64            r[1]=0;
65            for(int j=0,x,y;j<=g[i].size();++j){
66                if(!j)x=1;
67                else x=g[i][j-1]+1;
68                if(j==g[i].size())y=m;
69                else y=g[i][j]-1;
70                if(x>y)continue;
71                l[1][++r[1]]={x,y,++cnt_id};
72                int cnt=1ll*(y-x+1)*(q[i]-q[i-1])%mod;
73                f[cnt_id]={cnt,int(1ll*cnt*(cnt+1)/2%mod)};
74                fa[cnt_id]=cnt_id;
75            }
76            int rtmp[2]={1,1};
77            while(rtmp[0]<=r[0]||rtmp[1]<=r[1]){
78                bool f0=rtmp[0]<=r[0];
79                bool f1=rtmp[1]<=r[1];
80                if(f0&&f1){
81                    if(l[0][rtmp[0]].l<=l[1][rtmp[1]].l){
82                        work(rtmp[0],rtmp[1]-1);
83                        ++rtmp[0];
84                    }
85                    else{
86                        work(rtmp[0]-1,rtmp[1]);
87                        ++rtmp[1];
88                    }
89                }
90                else if(f0){
91                    work(rtmp[0],r[1]);
92                    ++rtmp[0];
93                }
94                else{
95                    work(r[0],rtmp[1]);
96                    ++rtmp[1];
97                }
98            }
99            r[0]=r[1];
100           for(int i=1;i<=r[0];++i)l[0][i]=l[1][i];
101       }
102       int ans=0;
103       for(int i=1;i<=cnt_id;++i)
104       if(gf(i)==i){
105           (ans+=f[i].f)%=mod;
```

```
106         }
107         printf("%d\n",ans);
108     }
109 }
```

6 数学

6.1 多项式

6.1.1 快速傅里叶变换

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef double db;
5  const int N = 3e5+1;
6  const db pi = acos(-1);
7
8  int n,m,rev[N];
9  int len1,len2;
10
11 struct C{
12     db x,y;
13     C(db _x=0,db _y=0){
14         x=_x;y=_y;
15     }
16     inline friend C operator + (C a,C b){
17         return C(a.x+b.x,a.y+b.y);
18     }
19     inline friend C operator - (C a,C b){
20         return C(a.x-b.x,a.y-b.y);
21     }
22     inline friend C operator * (C a,C b){
23         return C(a.x*b.x-a.y*b.y,a.x*b.y+a.y*b.x);
24     }
25     inline friend C conj(C a){
26         return C(a.x,-a.y);
27     }
28     inline friend C exp(C a){
29         db rho=exp(a.x);
30         db theta=a.y;
31         return C(rho*cos(theta),rho*sin(theta));
32     }
33 }a[N],b[N];
34
35 void fft(C *a,int f){
36     for(int i=0;i<n;++i)if(i<rev[i])swap(a[i],a[rev[i]]);
37     for(int i=1;i<n;i<<=1){
38         db alpha=pi/i;
39         if(f==-1)alpha=-pi/i;
40         for(int k=0;k<i;++k){
41             C w=exp(C(0,alpha*k));
```

```
42          for(int j=k;j<n;j+=(i<<1)){
43              C x=w*a[j+i];
44              a[j+i]=a[j]-x;
45              a[j]=a[j]+x;
46          }
47      }
48  }
49  return;
50  }
51
52  int main(){
53      scanf("%d%d",&len1,&len2);
54      for(int i=0;i<=len1;++i)scanf("%lf",&a[i].x);
55      for(int i=0;i<=len2;++i)scanf("%lf",&b[i].x);
56      for(n=1,m=0;n<=len1+len2;n<<=1,m++);
57      for(int i=0;i<n;++i)rev[i]=rev[i>>1]>>1|(i&1)<<(m-1);
58      fft(a,1);fft(b,1);
59      for(int i=0;i<n;++i)a[i]=a[i]*b[i];
60      fft(a,-1);
61      for(int i=0;i<=len1+len2;++i)printf("%d ",(int)(a[i].x/n+0.5));
62  }
```

6.1.2 快速数论变换

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int N = 2e5+5;
4   const int G = 3;
5   const int mod = 998244353;
6
7   int fac[N],rev[N];
8
9   int qpow(int a,int b){
10      int r=1;
11      while(b){
12          if(b&1)r=1ll*r*a%mod;
13          b>>=1;a=1ll*a*a%mod;
14      }
15      return r;
16  }
17
18  struct NTT{
19      int n,m,rev[N<<1];
20      int a[N<<1],b[N<<1];
21
22      void init(int len){
23          for(n=1,m=0;n<=len;n<<=1,m++);
24          for(int i=0;i<n;++i){
25              rev[i]=rev[i>>1]>>1|(1&i)<<(m-1);
26              a[i]=b[i]=0;
27          }
28      }
```

```
29
30      void ntt(int *a,int f){
31          for(int i=0;i<n;++i)if(i<rev[i])swap(a[i],a[rev[i]]);
32          for(int i=1;i<n;i<<=1){
33              int wn=qpow(G,(mod-1)/(i<<1));
34              if(f==-1)wn=qpow(wn,mod-2);
35              for(int j=0;j<n;j+=i<<1){
36                  int w=1;
37                  for(int k=0;k<i;++k,w=1ll*w*wn%mod){
38                      int x=a[j+k],y=1ll*a[j+k+i]*w%mod;
39                      a[j+k]=(x+y)%mod;a[j+k+i]=(x-y+mod)%mod;
40                  }
41              }
42          }
43          if(f==-1){
44              int rn=qpow(n,mod-2);
45              for(int i=0;i<n;++i)a[i]=1ll*a[i]*rn%mod;
46          }
47      }
48
49      void work(){
50          ntt(a,1);ntt(b,1);
51          for(int i=0;i<n;++i)a[i]=1ll*a[i]*b[i]%mod;
52          ntt(a,-1);
53      }
54  }B;
55
56  int main(){
57  }
```

6.1.3 快速沃尔什变换

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   const int N = 4e5+1;
5   const ll mod = 1e9+7;
6
7   int n;
8
9   ll qpow(ll a,ll b){
10      ll ret=1;
11      while(b){
12          if(b&1)ret=ret*a%mod;
13          b>>=1;
14          a=a*a%mod;
15      }
16      return ret;
17  }
18
19  struct FWT{
20      int n;
```

```
    ll a[N],b[N];

    void prepare(int len){
        for(n=1;n<=len;n<<=1);
    }

    void fwt(ll *a,int f){
        ll rev=1;
        if(f==-1)rev=qpow(2,mod-2);
        for(int i=1;i<n;i<<=1){
            for(int j=0;j<n;j+=i<<1){
                for(int k=0;k<i;++k){
                    ll x=a[j+k],y=a[j+k+i];
                    a[j+k]=(x+y)*rev%mod;a[j+k+i]=(x-y+mod)*rev%mod;
    //    xor a[j+k]=x+y;a[j+k+i]=x-y;
    //     a[j+k]=(x+y)/2;a[j+k+i]=(x-y)/2;
    //    and a[j+k]=x+y;
    //     a[j+k]=x-y;
    //    or a[j+k+i]=x+y;
    //     a[j+k+i]=y-x;
                }
            }
        }
    }
}F;

void solve(ll *a,ll *b,int n){
    F.prepare(n);
    F.fwt(a,1);F.fwt(b,1);
    for(int i=0;i<F.n;++i)(a[i]*=b[i])%=mod;
    F.fwt(a,-1);
}

int main(){
    cin>>n;
    for(int i=0;i<n;++i)scanf("%d",&F.a[i]);
    for(int i=0;i<n;++i)scanf("%d",&F.b[i]);
    solve(F.a,F.b,n);
    for(int i=0;i<F.n;++i)cout<<F.a[i]<<" ";
}
```

6.1.4 多项式求逆

```
#include<bits/stdc++.h>
#define inv(x) qpow(x,mod-2)
using namespace std;
typedef long long ll;
const int N = 2e5+1;
const ll G = 3;
const ll mod = 1004535809;

ll a[N*4],b[N*4],c[N*4];
```

```
int n,m,rev[N*4];
ll inv_n;

ll qpow(ll a,ll b){
    ll ret=1;
    while(b){
        if(b&1)ret=ret*a%mod;
        b>>=1;
        a=a*a%mod;
    }
    return ret;
}

void init(int l){
    for(n=1,m=0;n<=l;n<<=1,m++);
    for(int i=0;i<n;++i)rev[i]=rev[i>>1]>>1|(1&i)<<(m-1);
}

void NTT(ll *a,int f){
    for(int i=0;i<n;++i)if(i<rev[i])swap(a[i],a[rev[i]]);
    for(int i=1;i<n;i<<=1){
        ll wn=qpow(G,(mod-1)/(i<<1));
        if(f==-1)wn=inv(wn);
        for(int j=0;j<n;j+=i<<1)
        for(int k=0,w=1;k<i;++k,w=(ll)w*wn%mod){
            ll x=a[j+k],y=a[j+k+i]*w%mod;
            a[j+k]=(x+y)%mod;a[i+j+k]=(x-y+mod)%mod;
        }
    }
    if(~f)return;
    inv_n=inv(n);
    for(int i=0;i<n;++i)a[i]=a[i]*inv_n%mod;
}

void getinv(ll *a,int n_now){
    if(n_now==1)return void(b[0]=inv(a[0]));
    getinv(a,n_now+1>>1);
    init(n_now<<2);
    for(int i=0;i<n_now;++i)c[i]=a[i];
    for(int i=n_now;i<n;++i)c[i]=0;
    NTT(b,1);NTT(c,1);
    for(int i=0;i<n;++i){
        b[i]=b[i]*(2-c[i]*b[i]%mod)%mod;
        if(b[i]<0)b[i]+=mod;
    }
    NTT(b,-1);
    for(int i=n_now;i<n;++i)b[i]=0;
}

int main(){
    int n_now;
    cin>>n_now;
    for(int i=0;i<n_now;++i)cin>>a[i];
```

```
    getinv(a,n_now);
    init(n_now<<1);
    NTT(a,1);NTT(b,1);
    for(int i=0;i<n;++i)a[i]=a[i]*b[i]%mod;
    NTT(a,-1);
    for(int i=0;i<n_now;++i)cout<<a[i]<<" ";
}
```

6.1.5 快速阶乘

```
// luogu-judger-enable-o2
#include<cstdio>
#include<algorithm>
#include<cmath>
using namespace std;const int N=262144+10;typedef unsigned long long ll;
const int P=65536;const int SF=16;const int msk=65535;ll mod;ll PP;
typedef long double ld;const ld pi=acos(-1.0);
inline ll po(ll a,ll p){ll r=1;for(;p;p>>=1,a=a*a%mod)if(p&1)r=r*a%mod;return r;}
struct cmp
{
    ld r;ld v;
    friend cmp operator +(cmp a,cmp b){return (cmp){a.r+b.r,a.v+b.v};}
    friend cmp operator -(cmp a,cmp b){return (cmp){a.r-b.r,a.v-b.v};}
    friend cmp operator *(cmp a,cmp b){return (cmp){a.r*b.r-a.v*b.v,a.r*b.v+a.v*b.r
        };}
    void operator /=(const int& len){r/=len;v/=len;}
}rt[2][22][N],tr[N],tr1[N],tr2[N],tr3[N],tr4[N],tr5[N],tr6[N];
int rv[22][N];ll m13[N],m14[N],m23[N],m24[N];
inline void pre()
{
    for(int d=1;d<=18;d++)
        for(int i=1;i<(1<<d);i++)rv[d][i]=(rv[d][i>>1]>>1)|((i&1)<<(d-1));
    for(int d=1,t=1;d<=18;d++,t<<=1)
        for(int i=0;i<(1<<d);i++)rt[0][d][i]=(cmp){cos(pi*i/t),sin(pi*i/t)};
    for(int d=1,t=1;d<=18;d++,t<<=1)
        for(int i=0;i<(1<<d);i++)rt[1][d][i]=(cmp){cos(pi*i/t),-sin(pi*i/t)};
}inline void fft(cmp* a,int len,int d,int o)
{
    for(int i=1;i<len;i++)if(i<rv[d][i])swap(a[i],a[rv[d][i]]);cmp* w;int i;
    for(int k=1,j=1;k<len;k<<=1,j++)
        for(int s=0;s<len;s+=(k<<1))
            for(i=s,w=rt[o][j];i<s+k;i++,++w)
                {cmp a1=a[i+k]*(*w);a[i+k]=a[i]-a1;a[i]=a[i]+a1;}
    if(o)for(int i=0;i<len;i++)a[i]/=len;
}inline void dbdft(ll* a,int len,int d,cmp* op1,cmp* op2)
{
    for(int i=0;i<len;i++)tr[i]=(cmp){(ld)(a[i]>>SF),(ld)(a[i]&msk)};
    fft(tr,len,d,0);tr[len]=tr[0];
    for(cmp* p1=tr,*p2=tr+len,*p3=op1;p1!=tr+len;++p1,--p2,++p3)
        (*p3)=(cmp){p1->r+p2->r,p1->v-p2->v}*(cmp){0.5,0};
    for(cmp* p1=tr,*p2=tr+len,*p3=op2;p1!=tr+len;++p1,--p2,++p3)
        (*p3)=(cmp){p1->r-p2->r,p1->v+p2->v}*(cmp){0,-0.5};
```

```
}inline void dbidft(cmp* tr,int len,int d,ll* a,ll* b)
{
    fft(tr,len,d,1);
    for(int i=0;i<len;i++)a[i]=(ll)(tr[i].r+0.5)%mod;
    for(int i=0;i<len;i++)b[i]=(ll)(tr[i].v+0.5)%mod;
}inline void poly_mul(ll* a,ll* b,ll* c,int len,int d)//以上都是任意模数fft的板子
{
    dbdft(a,len,d,tr1,tr2);dbdft(b,len,d,tr3,tr4);
    for(int i=0;i<len;i++)tr5[i]=tr1[i]*tr3[i]+(cmp){0,1}*tr2[i]*tr4[i];
    for(int i=0;i<len;i++)tr6[i]=tr2[i]*tr3[i]+(cmp){0,1}*tr1[i]*tr4[i];
    dbidft(tr5,len,d,m13,m24);dbidft(tr6,len,d,m23,m14);
    for(int i=0;i<len;i++)c[i]=m13[i]*PP%mod;
    for(int i=0;i<len;i++)(c[i]+=(m23[i]+m14[i])*P+m24[i])%=mod;
}namespace iter
{
    ll f[N];ll g[N];ll h[N];ll ifac[N];
    inline void ih()
    {
        ifac[0]=ifac[1]=1;
        for(int i=2;i<min((ll)N,mod);i++)ifac[i]=(mod-mod/i)*ifac[mod%i]%mod;
        for(int i=1;i<min((ll)N,mod);i++)(ifac[i]*=ifac[i-1])%=mod;
    }inline void calch(ll del,int cur,ll* ip,ll* op)
    {
        int d=0;int len=1;while(len<=cur+cur+cur)len<<=1,d++;
        for(int i=0;i<=cur;i++)f[i]=ip[i]*ifac[i]%mod*ifac[cur-i]%mod;
        for(int i=cur-1;i>=0;i-=2)f[i]=(mod-f[i])%mod;
        for(int i=0;i<=cur+cur;i++)g[i]=po((del+mod-cur+i)%mod,mod-2);
        for(int i=cur+1;i<len;i++)f[i]=0;for(int i=cur+cur+1;i<len;i++)g[i]=0;
        poly_mul(f,g,h,len,d);//卷积求出h'
        ll xs=1;ll p1=del-cur;ll p2=del;
        for(int i=p1;i<=p2;i++)(xs*=i)%=mod;
        for(int i=0;i<=cur;i++,p1++,p2++)//双指针求出系数
        {
            op[i]=h[i+cur]*xs%mod;
            (xs*=po(p1,mod-2))%=mod,(xs*=(p2+1))%=mod;
        }
    }
}ll val[N];ll fv1[N];ll fv2[N];
inline void solve(int n)//倍增
{
    int hb=0;for(int p=n;p;p>>=1)hb++;val[0]=1;
    for(int z=hb,cur=0;z>=0;z--)
    {
        if(cur!=0)//把d乘2
        {
            iter::calch(cur+1,cur,val,fv1);
            for(int i=0;i<=cur;i++)val[cur+i+1]=fv1[i];val[cur<<1|1]=0;
            iter::calch(cur*po(n,mod-2)%mod,cur<<1,val,fv2);
            cur<<=1;for(int i=0;i<=cur;i++)(val[i]*=fv2[i])%=mod;
        }if((n>>z)&1)//把d加1
        {
            for(int i=0;i<=cur;i++)(val[i]*=(ll)(n*i)+cur+1)%=mod;cur|=1;val[cur]=1;
            for(int i=1;i<=cur;i++)(val[cur]*=(ll)cur*n+i)%=mod;
```

```
95          }
96      }
97  }
98  int main()
99  {
100     pre();int n;scanf("%d%lld",&n,&mod);iter::ih();
101     int bl=sqrt(n);PP=(ll)P*P%mod;solve(bl);ll res=1;
102     for(int i=0,id=0;;i+=bl,id++)//分块
103     {
104         if((ll)i+bl>n){for(int j=i+1;j<=n;j++)(res*=j)%=mod;break;}
105         (res*=val[id])%=mod;
106     }printf("%lld",res);return 0;//拜拜程序~
107 }
```

6.2 博弈论

6.2.1 SG 函数 DP 中所有不同的后继状态都应可以通过不同的操作到达

为了规范化，求 SG 函数的方程一般由规模和禁手组成，一般不能限定下一次的操作类型（为了确保所有后继状态都可通过不同的操作来到达）

举个例子：

有一堆石子，两个人轮流从中拿 $w[0]/w[1]/w[2]$ 个石子，但是如果对方上一次拿了 $w[1]/w[2]$ 个，那么自己这一次就不能拿 $w[1]/w[2]$ 个，不能拿者输，求 SG 函数

1.

如果设 $f[i][j]$ 表示当前有 i 个石子，本轮操作的人决定要拿 $w[j]$ 个石子的 SG 值（此处限定了下一次操作的类型）

那么显然 $f[i][j]=mex\{f[i-w[j]][k](j=0\ or\ j\neq k)\}$

首先，我们要求的显然是不限定第一次操作要取多少个，只要总共取 n 个就行的 SG 值；但是 $f[n][0/1/2]$ 却分别表示限定第一次操作拿 $w[0/1/2]$ 个的 SG 值，很难推到我们想要的 SG 值

其次，这样求出来的 f 值是完全错误的！考虑 $f[i-w[1]][0/2]$ 分别为 $0/2$，那么 $f[i][1]=1$，代表此处先手必胜，具体获胜方法为取 $w[1]$ 个石子，但是此处你无法限定状态一定转移到 $f[i-w[1]][0]$（即你无法强迫下一论对方一定取 $w[0]$），你没法决定下一个人要选哪一种操作，它自然可以选 $f[i-w[1]][2]$ 保证他一定获胜，此处就与先手必胜矛盾。**通俗地讲，平时的 $f_i=mex\{f_k\}$ 可行是因为你可以通过拿 $i-k$ 个石子将局面限定到固定的后继状态 f_k 上，但是这里不可行是因为你拿 $w[j]$ 个也无法保证一定将局面限定到某个后继状态 $f[i-w[j]][k]$ 上。一般地讲，这是因为不同的后继状态可以通过相同的操作（此处即取 $w[j]$ 个石子）得到，使得当前轮到的人无法通过决定自己的本次操作到达所有的后继状态，此时再取 mex 就肯定不可行了**

2.

如果设 $f[i][j]$ 表示当前有 i 个石子，本轮操作的人不能拿 $w[j]$ 个石子（表示禁手）的 SG 值（$f[i][0]$ 表示本轮操作的人没有禁手）

那么显然 $f[i][j]=mex\ f[i-w[k]][k](j=0\ or\ j\neq k)$

那么显然 $f[n][0]$ 即为所求

6.2.2 后继状态存在偏序关系的 SG 函数转化为取 max

若后继状态存在偏序关系（即如果状态 S_1 可以转移到 S_2，状态 S_2 可以转移到状态 S_3 时，那么 S_1 一定可以转移到 S_3），假设 $SG(S_2)=x$，那么所有 0 $x-1$ 的 SG 值都在 S_2 的后继状态中出现过，又因为存在偏序关系，S_1 的后继状态包含 S_2 的所有后继状态。所以 0 $x-1$ 和 x 的 SG 值一定在 S_1 的后继状态出现过，所以 $SG(S_1)>x$。又因为 SG 是未在后继状态中出现过的最小自然数，所以 $SG(S_1)=max_{t\ 1}\ \ (SG_t)+1$。至此我们就将复杂度取 mex 问题转化为了更为简单的取 max 问题。

（由排列区间取 mex 转化为区间外取 min 和后继状态存在偏序关系的 SG 函数计算转化为取 max 可知，SG 函数复杂的 mex 运算可以通过问题的性质化为更简单的 min/max 操作）

6.2.3 排列的区间取 mex 问题转化为区间外取 min

给定一个长度为 n 的排列，有两种操作，分别为区间取 mex 与交换两个相邻位置的元素。

考虑到所给序列是一个排列，所以 0 $n-1$ 的每个元素要么在查询区间中出现，要么在区间外出现，所以区间内最小未出现的元素即区间外的最小元素（当区间包含整个序列时，答案应为 n，没有在区间外出现，这种情况特判即可）。记录前缀后缀最小值，查询时答案即为 $min(mnl[l-1],mnr[r+1])$。当交换相邻元素时，只有这两个相邻位置的前缀后缀最小值可能会发生改变。所以整体复杂度为 $O(n+Q)$

（由排列区间取 mex 转化为区间外取 min 和后继状态存在偏序关系的 SG 函数计算转化为取 max 可知，SG 函数复杂的 mex 运算可以通过问题的性质化为更简单的 min/max 操作）

6.3 斐波那契数列

6.3.1 斐波那契数列的性质

$f(1)=1,f(0)=0$

$f(n)=f(n-1)+f(n-2)$

$f(n)=((\frac{1+\sqrt{5}}{2})^n-(\frac{1-\sqrt{5}}{2})^n)/\sqrt{5}$

斐波那契数列任意两项和都不相等

斐波那契数列的第 $n+2$ 项代表了集合 $\{1\ 2\ ...n\}$ 中所有不包含相邻正整数的子集的个数

$f(n+m)=f(n-1)f(m)+f(n)f(m+1)$

$gcd(f(n),f(m))=f(gcd(n,m))$

$\frac{a_n}{a_{n+1}}$ 趋近于黄金比 0.618

$f(1)+f(3)+f(5)+...+f(2n-1)=f(2n)-f(2)+f(1)$

$f(2)+f(4)+f(6)+...+f(2n)=f(2n+1)-f(1)$

$f^2(1)+f^2(2)+...+f^2(n)=f(n)f(n+1)$

$\frac{f(2n)}{f(n)}=f(n-1)+f(n+1)$

$f(n-1)f(n+1)-f^2(n)=(-1)^n$

$f(1)+2f(2)+3f(3)+...+nf(n)=nf(n+2)-f(n+3)+2$

位数循环周期：

最后一位：$f(n+60)=f(n)$

最后二位：$f(n+300)=f(n)$

最后三位：$f(n+1500)=f(n)$

6.3.2 广义斐波那契数列的循环节

$f(n)=af(n-1)+bf(n-2), f(1)=c, f(2)=d$

求 $f(n)$ mod p 的最小循环节长度。

当 c 是模 p 的二次剩余时，枚举 $n=p-1$ 的因子。

当 c 是模 p 的二次非剩余时，枚举 $n=(p+1)(p-1)$ 的因子。

找到最小的因子 ans，使得

$$\begin{pmatrix} a & b \\ 1 & 0 \end{pmatrix}^{ans} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} (mod\ p)$$

具体证明见 https://blog.csdn.net/ACdreamers/article/details/25616461

6.4 卡特兰数

通项公式：$C(2*n,n)-C(2*n,n-1)$ 或者 $\frac{C(2*n,n)}{n+1}$

递推公式：$f_n=\sum_{k=0}^{n-1} f_k * f_{n-k-1}$

一个栈 (无穷大) 的进栈序列为，有多少种不同的出栈序列?

n 个节点的无标号二叉树有多少种?

一个凸 n 边形，每次连接两个顶点可以将其划分成三角形。请问有多少种不同的剖分方法?

前 10 项：

```
int catalan[]={
   1,
   2,
   5,
   14,
   42,
   132,
   429,
   1430,
   4862,
   16796
};
```

6.5 k 小线性基

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 64;
```

```
5
6  ll n,m,k;
7  struct Basis{
8     ll a[N],b[N];
9     vector<ll> s;
10
11    void insert(ll x){
12       for(int i=N-1;~i;--i)
13       if(x&1ll<<i)x^=a[i];
14       for(int i=N-1;~i;--i)
15       if((x&1ll<<i)&&!a[i]){
16          a[i]=x;
17          for(int j=i+1;j<N;++j)
18          if(a[j]&1ll<<i)a[j]^=a[i];
19          return;
20       }
21    }
22
23    void prepare(){
24       for(int i=0;i<64;++i)
25       if(a[i]){
26          s.push_back(a[i]);
27       }
28    }
29
30    ll kth(ll k){
31       if(s.size()!=n)k--;
32       ll ret=0;
33       if(k>=(1ll<<s.size()))return -1;
34       for(ll i=0;i<N;++i)
35       if(k&(1ll<<i))ret^=s[i];
36       return ret;
37    }
38 }B;
39
40 int main(){
41    cin>>n;
42    ll x;
43    for(int i=1;i<=n;++i){
44       cin>>x;
45       B.insert(x);
46    }
47    B.prepare();
48    cin>>m;
49    for(int i=1;i<=m;++i){
50       cin>>k;
51       cout<<B.kth(k)<<endl;
52    }
53 }
```

6.6 线性基的交

线性基的交：假设有 A,B 两个线性基，要求出一个线性基 C，使得 C 表示的线性空间为 A 所表示的线性空间和 B 所表示的线性空间的交。C 即为线性基 A,B 的交。

下面先说线性基怎么求交。如果我们有 A,B 两个线性基，要求它们的交线性基 C，那么显然 B 中所有能被 A 线性基表示的数，都要插入 C 中，之后如果 $A\ (B\ C)$ 线性无关（$B\ C$ 代表 B 中所有不能被 A 线性基表示的数），则 C 就是 A,B 的交线性基。（证明：若 $V1,V2$ 分别是 A,B 构成的线性空间，令 $W=B\ V1$ ，则需要证明的为：若 $A\ (B\ W)$ 线性无关，则 W 是 $V1\ V2$ 的一组基。考虑任意 $v\ V1\ V2$，那么 v 可以同时被 A 和 B 表出。考虑如何证明 v 可以被 W 线性表示。我们假设 v 不能可以被 W 线性表示，那么 v 一定可以被 S 和 T 共同线性表示，其中 $S\ W,T\ B\ W$，且其中 T 不为空。那么此时 T 一定与 A 线性相关（因为向量组 $S\ T$ 和 A 均与 $V1\ V2$ 等价，所以 $S\ T$ 与 A 等价，所以 A 可以表示出 T），与我们的前提不符。所以上述假设成立）

但问题是 $A\ (B\ C)$ 未必线性无关，所以我们采取下面这种做法。再额外创建一个线性基 D，同时 D 上每一位再额外保存一个值 v，一开始把 A 中所有的数字 $A[i]$ 插入 D，并把对应插入位置的 v 也改为 $A[i]$。之后从低位到高位，将 B 中每一个数字 $B[i]$ 插入 D，每次插入时，维护一个 u 值，u 一开始等于 1，插入时每访问到 D 的某一位，u 就等于 u 异或对应位上的 v。

如果 $B[i]$ 可以被 D 表示，那么插入一定失败，把 u 插入 C；如果 $B[i]$ 不可以被 D 表示，那么一定会将一个值 X 插入 D 中某一位，同时把对应位的 v 值改为 u。这样将 B 遍历完毕之后，C 就是 A 和 B 的交线性基。

时间复杂度为 $log(b^2)$，其中 b 为线性基的维数。

线性基的并：假设有 A,B 两个线性基，要求出一个线性基 C，使得 C 表示的线性空间为 A 所表示的线性空间和 B 所表示的线性空间的并。C 即为线性基 A,B 的并。（有题解说可以用容斥原理结合求交实现，但并真的可能存在么？比如说线性基 A 中只有 10，B 中只有 01，那么显然 C 表示的线性空间应该为 $\{10,01\}$，但是显然不存在 C 满足此条件，因为若 $01,10$ 都在线性空间中，那么 $01+10=11$ 必然在线性空间中）

```
struct Basis{
   int a[N];

   void insert(int x){
      for(int i=N-1;i>=0;--i)
      if(x&1<<i){
         if(a[i])x^=a[i];
         else{
            a[i]=x;
            return;
         }
      }
   }
};

void merge(Basis &C,Basis A,Basis B){
   Basis t1,t2;
   for(int i=0;i<N;++i){
```

```
        t1.a[i]=t2.a[i]=A.a[i];
    }
    for(int i=0;i<N;++i)
    if(B.a[i]){
        int x=B.a[i],tmp=0;
        for(int j=N-1;~j;--j)
        if(x&1<<j){
            if(!t1.a[j]){
                t1.a[j]=x;
                t2.a[j]=tmp;
                break;
            }
            x^=t1.a[j];
            tmp^=t2.a[j];
        }
        if(!x)C.insert(tmp);
    }
}
```

6.7 矩阵求逆

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef double db;
const int N = 2e2+1;

int n,m,H,h[N],d[N];
vector<int>g[N];
db a[N][N],c[N][N];

void gauss(int n){
    for(int i=1;i<=n;++i)c[i][i]=1;
    for(int i=1;i<=n;++i){
        int l=i;
        for(int j=i;j<=n;++j)if(abs(a[j][i])>abs(a[l][i]))l=j;
        for(int j=1;j<=n;++j)swap(a[l][j],a[i][j]),swap(c[l][j],c[i][j]);
        db p=a[i][i];
        for(int j=1;j<=n;++j)a[i][j]/=p,c[i][j]/=p;
        for(int j=1;j<=n;++j)
        if(i!=j){
            p=a[j][i];
            for(int k=1;k<=n;++k)a[j][k]-=p*a[i][k],c[j][k]-=p*c[i][k];
        }
    }
    return;
}
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i)
    for(int j=1;j<=n;++j)
    cin>>a[i][j];
```

```
    gauss(n);
}
```

6.8 杜教筛

求 $\sum_{i=1}^{n} u(i)$ 和 $\sum_{i=1}^{n} phi(i)$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e6+5;

int T;
int n;
struct fuc{
    ll u,phi;
    fuc(ll _u=0,ll _phi=0){
        u=_u;phi=_phi;
    }
    friend fuc operator - (fuc a,fuc b){
        return fuc(a.u-b.u,a.phi-b.phi);
    }
    friend fuc operator * (ll val,fuc a){
        return fuc(a.u*val,a.phi*val);
    }
}w[N];
unordered_map<int,fuc>w_map;

void prepare(){
    w[1]=fuc(1,1);
    static bool vis[N];
    static int p[N];
    for(int i=2;i<N;++i){
        if(!vis[i]){
            p[++p[0]]=i;
            w[i]=fuc(-1,i-1);
        }
        for(int j=1;j<=p[0]&&p[j]*i<N;++j){
            vis[i*p[j]]=1;
            if(i%p[j]==0){
                w[i*p[j]]=fuc(0,w[i].phi*p[j]);
                break;
            }
            w[i*p[j]]=fuc(-w[i].u,w[i].phi*(p[j]-1));
        }
    }
    for(int i=2;i<N;++i){
        w[i].u+=w[i-1].u;
        w[i].phi+=w[i-1].phi;
    }
}

```

```
fuc Solve(int n){
    if(n<N)return w[n];
    if(w_map.find(n)!=w_map.end())return w_map[n];
    fuc ret=fuc(1,1ll*n*(n+1)/2);
    for(int i=2,r;i<=n;i=r+1){
        r=n/(n/i);
        ret=ret-1ll*(r-i+1)*Solve(n/i);
    }
    return w_map[n]=ret;
}

int main(){
    prepare();
    cin>>T;
    while(T--){
        cin>>n;
        fuc ans=Solve(n);
        printf("%lld %lld\n",ans.phi,ans.u);
    }
}
```

6.9 高斯消元

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e3+5;

int n;
double a[N][N],f[N];

bool gauss(){
    for(int i=1;i<=n;++i){
        int l=i;
        for(int j=i+1;j<=n;++j)if(fabs(a[j][i])>fabs(a[l][i]))l=j;
        if(fabs(a[l][i])<1e-10)return false;
        for(int j=1;j<=n+1;++j)swap(a[i][j],a[l][j]);
        double p=a[i][i];
        for(int j=1;j<=n+1;++j)a[i][j]/=p;
        for(int j=1;j<=n;++j)
        if(i!=j){
            p=a[j][i];
            for(int k=1;k<=n+1;++k)a[j][k]-=p*a[i][k];
        }
    }
    for(int i=1;i<=n;++i)f[i]=a[i][n+1];
    return true;
}

int main(){
    cin>>n;
    for(int i=1;i<=n;++i)
    for(int j=1;j<=n+1;++j){
```

```
30        cin>>a[i][j];
31    }
32    if(!gauss()){
33        printf("No Solution\n");
34    }
35    else{
36        for(int i=1;i<=n;++i)printf("%.2lf\n",f[i]);
37    }
38    return 0;
39 }
```

6.10 组合数预处理

```
1  ll fac[N],rev[N];
2
3  ll qpow(ll a,ll b){
4      ll r=1;
5      while(b){
6          if(b&1)r=r*a%mod;
7          b>>=1;
8          a=a*a%mod;
9      }
10     return r;
11 }
12
13 void prepare(){
14     fac[0]=1;
15     for(int i=1;i<N;++i)fac[i]=fac[i-1]*i%mod;
16     rev[N-1]=qpow(fac[N-1],mod-2);
17     for(int i=N-2;~i;--i)rev[i]=rev[i+1]*(i+1)%mod;
18 }
19
20 ll C(int n,int m){
21     if(n<m||m<0)return 0;
22     return fac[n]*rev[m]%mod*rev[n-m]%mod;
23 }
```

6.11 各种情况下小球放盒子的方案数

k 个球	m 个盒子	是否允许有空盒子	方案数
各不相同	各不相同	是	m^k
各不相同	各不相同	否	m !Stirling2 (k,m)
各不相同	完全相同	是	$\sum_{i=1}^{m}$ Stirling2(k,i)
各不相同	完全相同	否	Stirling $2(k,m)$
完全相同	各不相同	是	$C(m+k-1,k)$
完全相同	各不相同	否	$C(k-1,m-1)$
完全相同	完全相同	是	$\frac{1}{(1-x)(1-x^2)\ldots(1-x^m)}$ 的x^k 项的系数
完全相同	完全相同	否	$\frac{x^m}{(1-x)(1-x^2)\ldots(1-x^m)}$ 的x^k 项的系数

6.12 康托展开与康托逆展开

```cpp
int contor(const vector<int>& permutation) {
    int num = 0;
    int len = permutation.size();
    for (int i = 0; i < len; ++i) {
        int cnt = 0; // 在 i 之后，比 i 还小的有几个
        for (int j = i + 1; j < len; ++j)
            if (permutation[i] > permutation[j]) ++cnt;
        num += cnt * fact[len - i - 1];
    }
    return num + 1;
}

vector<int> revContor(int bits, int num) {
    num = num - 1; //有 num - 1 个排列比目标序列要小
    vector<bool> vis(bits + 1, false);
    vector<int> permutation(bits, -1);

    int n, residue = num;
    for (int i = 0; i < bits; ++i) {
        n = residue / (fact[bits - i - 1]);
        residue = residue % (fact[bits - i - 1]);

        for (int j = 1; j <= bits; ++j) {
            if (!vis[j] && !(n--)) {
                vis[j] = true;
                permutation[i] = j;
                break;
            }
        }
    }
    return permutation;
}
```

6.13 BM 算法利用序列前 k 项猜第 n 项

如果 f_n 不是线性递推式，那么不妨试试 $f_n/n!, f_n/(n-k)!$ 等公式。

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e4+5;
const int mod = 998244353;

int k,n;

ll qpow(ll a,ll b){
    ll r=1;
```

```
    while(b){
        if(b&1)r=r*a%mod;
        b>>=1;
        a=a*a%mod;
    }
    return r;
}

namespace Linear_Seq{
    int k;
    ll a[N],h[N],s[N],t[N];
    vector<int> BM(vector<int> x){
        vector<int> ls,cur;
        int pn=0,lf,ld;
        for(int i=0;i<x.size();++i){
            ll t=-x[i]%mod;
            for(int j=0;j<cur.size();++j){
                t=(t+x[i-j-1]*(ll)cur[j])%mod;
            }
            if(!t)continue;
            if(!cur.size()){
                cur.resize(i+1);
                lf=i;
                ld=t;
                continue;
            }
            ll k=-t*qpow(ld,mod-2)%mod;
            vector<int> c(i-lf-1);
            c.push_back(-k);
            for(int j=0;j<ls.size();++j)c.push_back(ls[j]*k%mod);
            if(c.size()<cur.size())c.resize(cur.size());
            for(int j=0;j<cur.size();++j)c[j]=(c[j]+cur[j])%mod;
            if(i-lf+ls.size()>=cur.size()){
                ls=cur;
                lf=i;
                ld=t;
            }
            cur=c;
        }
        vector<int> &o=cur;
        for(int i=0;i<o.size();++i)o[i]=(o[i]%mod+mod)%mod;
        return o;
    }

    void mull(ll *p,ll *q){
        static ll w[N];
        for(int i=0;i<2*k;++i)w[i]=0;
        for(int i=0;i<k;++i)
        if(p[i]){
            for(int j=0;j<k;++j){
                w[i+j]=(w[i+j]+p[i]*q[j])%mod;
            }
        }
```

```
64        for(int i=2*k-1;i>=k;--i)
65        if(w[i]){
66           for(int j=k-1;~j;--j){
67              w[i-j-1]=(w[i-j-1]+w[i]*h[j])%mod;
68           }
69        }
70        for(int i=0;i<k;++i)p[i]=w[i];
71     }
72
73     ll calc(ll K){
74        for(int i=k;~i;--i)s[i]=t[i]=0;
75        s[0]=1;
76        if(k!=1)t[1]=1;
77        else t[0]=h[0];
78        for(;K;mull(t,t),K>>=1)
79        if(K&1){
80           mull(s,t);
81        }
82        ll ret=0;
83        for(int i=0;i<k;++i)ret=(ret+s[i]*a[i])%mod;
84        return (ret%mod+mod)%mod;
85     }
86
87     int get(vector<int> x,ll n){
88        if(n<x.size())return x[n];
89        vector<int> v=BM(x);//表示最短递推式
90        k=v.size();
91        if(!k)return 0;
92        for(int i=0;i<k;++i){
93           h[i]=v[i];
94           a[i]=x[i];
95        }
96        return calc(n);
97     }
98  }
99
100 int main(){
101    cin>>k>>n;
102    vector<int> v;
103    for(int i=0,x;i<k;++i){
104       cin>>x;
105       v.push_back(x);
106    }
107    cout<<Linear_Seq::get(v,n)<<endl;
108 }
```

6.14 单纯形

最大化 $\sum_j^n c_j x_j$

满足约束 $\sum_j^n a_{ij} x_j \le b_i$，其中 $i = 1, 2, ..., m$

且满足条件 $x_j \ge 0$，其中 $j = 1, 2, ..., n$

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double db;
const int N= 1e2+1;
const db eps = 1e-8;
const db inf = 1e15;

inline int read(){
    char c=getchar();int x=0,f=1;
    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;
}

int n,m,type;
db a[N][N],ans[N];
int id[N<<1];

void pivot(int l,int e){
    swap(id[n+l],id[e]);
    db t=a[l][e];a[l][e]=1;
    for(int j=0;j<=n;++j)a[l][j]/=t;
    for(int i=0;i<=m;++i)
    if(i!=l&&abs(a[i][e])>eps){
        t=a[i][e];a[i][e]=0;
        for(int j=0;j<=n;++j)a[i][j]-=a[l][j]*t;
    }
}

bool init(){
    while(1){
        int e=0,l=0;
        for(int i=1;i<=m;++i)if(a[i][0]<-eps&&(!l||(rand()&1)))l=i;
        if(!l)break;
        for(int j=1;j<=n;++j)if(a[l][j]<-eps&&(!e||(rand()&1)))e=j;
        if(!e){puts("Infeasible");return 0;}
        pivot(l,e);
    }
    return 1;
}

bool simplex(){
    while(1){
        int l=0,e=0; db mn=inf,mx=0;
        for(int j=1;j<=n;++j)if(a[0][j]-eps>mx)mx=a[0][j],e=j;
        if(!e)break;
        for(int i=1;i<=m;++i)
        if(a[i][e]>eps&&a[i][0]/a[i][e]<mn)
        mn=a[i][0]/a[i][e],l=i;
        if(!l){puts("Unbounded");return 0;}
        pivot(l,e);
```

```
    }
    return 1;
}

int main(){
    srand(time(NULL));
    n=read();m=read();type=read();
    for(int i=1;i<=n;++i)a[0][i]=read();
    for(int i=1;i<=m;++i){
        for(int j=1;j<=n;++j)a[i][j]=read();
        a[i][0]=read();
    }
    for(int i=1;i<=n;++i)id[i]=i;
    if(init()&&simplex()){
        printf("%.8lf\n",(double)-a[0][0]);
        if(type){
            for(int i=1;i<=m;++i)ans[id[n+i]]=a[i][0];
            for(int i=1;i<=n;++i)printf("%.8lf ",(double)ans[i]);
        }
    }
}
```

6.15 约瑟夫环

```
void Josephus(){
    //有n个人,其编号分别为1~n。这n个人按顺序排成一个圈。现在给定一个数k,从第一个人开始依次报数,数到k的人出列,然后又从下一个人开始又从1开始依次报数,数到k的人又出列...如此循环,最后出列的玩家获胜
    //n,k<=1e5:f[i]表示有i个玩家时，获胜者的编号
    for(int i=2;i<=n;++i)f[i]=(f[i-1]+k)%i;
    //k<=1e5,n<=1e9:ans表示获胜者编号
    ll ans=n*m;
    while(ans>n)ans+=(ans-n-1)/(k-1)-n;
}

void SPJosephus(){
    //反约瑟夫环游戏，除了报k以外的人都出列且n-1是m-1的倍数
    //n,k<=1e18:ans表示获胜者编号,num[i]表示第i轮所剩人数
    int i,ans;
    for(i=1;i<n/k;i*=k);
    ans=(n-i)/(k-1)*k;
}
```

6.16 prufer 序列

无根树转化为 $prufer$ 序列:

1. 每次找到编号最小的叶子节点 2. 删除该节点并在序列中添加与该节点相连的节点的编号

3. 重复 1, 2 操作，直到整棵树只剩下两个节点

$prufer$ 序列转化为无根树:

1. 每次取出 $prufer$ 序列中最前面的元素 u 2. 在点集 V（V 初始化为 $\{1,2,...,n\}$）中找到编号最小的没有在 $prufer$ 序列中出现的元素 v 3. 给 u,v 连边然后在序列中删除 u，在 V 中删除 v 4. 重复 $1,2,3$, 直到序列为空。此时在点集 V 中剩下两个节点，给它们连边即可

其中点集 V 用 set 维护可以做到 $P(nlogn)$ 的复杂度

证明：

注意到，如果一个节点 A 不是叶子节点，那么它至少有两条边；但在无根树转化为 $prufer$ 序列的过程结束后，整个图只剩下一条边，因此节点 A 的至少一个相邻节点被去掉过，节点 A 的编号将会在这棵树对应的 $prufer$ 编码中出现。反过来，在 $prufer$ 编码中出现过的数字显然不可能是这棵树（初始时）的叶子。于是我们看到，没有在 $prufer$ 编码中出现过的数字恰好就是这棵树（初始时）的叶子节点。把序列的第一个元素和 V 中最小编号的未出现点删除后，即在原树中删除了一个叶子节点和对应的边，剩下的新树和剩下的 $prufer$ 序列以及 V 仍然满足以上性质（只是多了个必须在 V 中出现的限制，这是因为不在 V 中出现的点此时已经不在新树中了）（如果想不明白对应关系的话，可以考虑画两个图，第一个图即原树（可以看作 V），第二个图是利用 $prufer$ 序列构造的树，每在原树中删除一个顶点一条边，就在第二个图中加入一条边）

性质：

1. $prufer$ 序列中节点 i 出现的次数为 d_i-1 2. 一棵 n 个节点的无根树唯一地对应了一个长度为 $n-2$ 的数列，数列中的每个数都在 1 到 n 的范围内（考虑转换中的 4 步，每一步都一定可行，第 3 步中序列的大小总比 V 大 2，所以总能找到满足的 v）3. n 个节点的有标号无根树有 $n^{(n-2)}$ 个，也可以看作完全无向图的生成树个数 4. n 个节点的度数分别为 $d_1,d_2,...,d_n$ 的无根树共有 $\frac{(n-2)!}{\Pi(d_i-1)!}$ 个，因为此时 $prufer$ 序列中节点 i 出现了 d_i-1 次（又因为该性质的存在，$prufer$ 序列常用于 dp 计算有度数限制的有标号无根树计数中）5. n 个节点的有根树有 $n^{(n-2)}*n=n^{(n-1)}$ 6. 考虑有 n 个节点，m 棵树的有根树森林计数我们建一个虚点 $n+1$，然后构造一棵 $n+1$ 个点的无根树，并假定以 $n+1$ 号点为根，然后把虚点拆掉，它的儿子就会形成若干棵有根树森林，我们只需要保证 $n+1$ 号点的度数恰好为 m 即可。所以方案数为 $C_{n-1}^{m-1}\ n^{n-m}$ 7. 接下来我们考虑不限制有多少棵树的有根树森林。可以由上面建虚点的方法直接得到方案数 $(n+1)^{n-1}$

6.17 本源勾股数

本源勾股数 (x,y,z) 满足：

1. $x^2+y^2=z^2$
2. $gcd(x,y,z)=1$
3. $x,y,z\in Z^+$

本源勾股数 (x,y,z) 可用以下公式推导：

1. $x=m^2-n^2$
2. $y=2mn$
3. $z=m^2+n^2$

其中 (m,n) 满足：

1. $m > n$
2. $gcd(m, n) = 1$
3. $m + n \equiv 1(mod\ 2)$
4. $m, n \in Z^+$

其中寻找互质数对 (m, n) 需要用到 $Stern - Brocot\ Tree$

前若干项：$(3, 4, 5), (5, 12, 13), (8, 15, 17), (7, 24, 25), (20, 21, 29), (9, 40, 41)$

6.18 五边形数求整数划分

给定数字 n，将其划分为若干个整数的和，其中每个整数只能用 k 次以内。

https://blog.csdn.net/zhoufenqin/article/details/9821617

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 100010;
int dp[maxn];
const int mod = 1000000000+7;
ll five(ll x){return (3*x*x-x)/2;}
int add_mod(int a,int b){return a+b>=mod?a+b-mod:a+b;}
void init(){
    dp[0]=1;
    for(int i=1;i<maxn;i++){
        for(int j=1;;j++){
            ll k=five(j);
            if(k<=i){
                dp[i]=add_mod(dp[i],(j&1)?dp[i-k]:mod-dp[i-k]);
            }
            else{
                break;
            }
            k=five(-j);
            if(k<=i){
                dp[i]=add_mod(dp[i],(j&1)?dp[i-k]:mod-dp[i-k]);
            }else{
                break;
            }
        }
    }
}
int q[maxn];
void initq(){
    q[0]=0;
    for(int i=1;i<maxn;i++){
        if(i&1) q[i]=five(i/2+1);
        else q[i]=five(-(i/2));
    }
}
int solve(ll n,ll k){
```

```
    ll ans=0;
    for(int i=0;;i++){
        if(q[i]*k>n) break;
        int t=i;
        if(t&1) t=t/2+1;
        else t=t/2;
        ans=add_mod(ans,(t&1)?mod-dp[n-q[i]*k]:dp[n-q[i]*k]);
    }
    return ans;
}
int main(){
    init();initq();
    int T;scanf("%d",&T);
    while(T--){
        int n,k;scanf("%d%d",&n,&k);
        printf("%d\n",solve(n,k));
    }
}
```

6.19 生成函数

普通生成函数:

常用于多重集组合问题。

物品 n 种，每种数量分别为 $k_1, k_2, ..., k_n$ 个，求从中可重复地选出 m 个物品的组合方法数。

设 $f_k(x) = 1 + x + x^2 + x^3 + ... + x^k$

则答案即为 $\prod f_{k_i}(x)$ 的第 m 项 $[x^m]$ 的系数。

若 $k- > \infty$，则 $f_k(x) = \frac{1}{1-x}$（现已加入多项式全家桶）

指数型生成函数:

常用于多重集排列问题。

物品 n 种，每种数量分别为 $k_1, k_2, ..., k_n$ 个，求从中可重复地选出 m 个物品的排列方法数。

设 $f_k(x) = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + ... + \frac{x^k}{k!}$

则答案即为 $\prod f_{k_i}(x)$ 的第 m 项 $[\frac{x^m}{m!}]$ 的系数。

若 $k- > \infty$，则 $f_k(x) = e^x$（现已加入多项式全家桶）

6.20 求数列中任意两数 lcm 最大值

给定序列 a，求 $max\{lcm(a_i, a_j)\}(n, a_i \leq 10^5)$

首先考虑将序列排序去重，并在过程中计算任意相等两数 lcm 的最大值

考虑枚举 gcd 的值，将所有 gcd 的倍数加入新序列内，问题转化为求序列中任意互质数对乘积的最大值

考虑维护一个栈，并将新序列中的元素从大到小依次加入，对于新加入的元素 x，如果栈中存在 y 与其互质，那么栈中所有比 y 小的元素都可以删除，因为此时 $x * y$ 一定大于接下来可能出现的答案 $x' * y'$。(y' 是栈中比 y 小的数，而 x' 是接下来处理的元素，由于从大到小处理，所以 x' 一定小于 x) 所以当栈中存在与 x 互质的元素时，可以一直删除栈顶并更新答案

考虑如何查询栈中是否存在元素与 x 互质，设 cnt_d 表示栈中的元素有多少个是 d 的倍数，则经过简单的推导可知与 x 互质的数的个数为 $\sum_{d|x}\mu(d)*cnt_d$。因此在栈中插入 x 与查询 x 的复杂度均为 $O(d(i))$

以上两种算法结合，可以将判定是否存在互质对的问题转化为求最大乘积互质对的问题

因为每个数要入栈 $d(i)$ 次，且每次入栈的复杂度小于 $O(d(i))$，所以总复杂度为 $O(\sum d(i)^2)$

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+1;

int n;
int a[N],u[N],cnt[N];
vector<int>g[N],d[N];
ll ans;

void prepare(){
   static bool vis[N];
   static int p[N];
   u[1]=1;
   for(int i=2;i<N;++i){
      if(!vis[i]){
         p[++p[0]]=i;
         u[i]=-1;
      }
      for(int j=1;j<=p[0]&&p[j]*i<N;++j){
         vis[i*p[j]]=1;
         if(i%p[j]==0)break;
         else u[i*p[j]]=-u[i];
      }
   }
   for(int i=1;i<N;++i)
   for(int j=1;i*j<N;++j)
   d[i*j].push_back(i);
}

void add(int x,int c){
   for(int i=0;i<d[x].size();++i){
      cnt[d[x][i]]+=c;
   }
}

bool ask(int x){
   int ret=0;
   for(int i=0;i<d[x].size();++i){
      ret+=cnt[d[x][i]]*u[d[x][i]];
   }
   return ret>0;
}

int main(){
   prepare();
```

```
    cin>>n;
    for(int i=1;i<=n;++i)cin>>a[i];
    sort(a+1,a+n+1);
    for(int i=1;i<n;++i)
    if(a[i]==a[i+1]){
        ans=max(ans,(ll)a[i]);
    }
    n=unique(a+1,a+n+1)-a-1;
    for(int i=1;i<=n;++i){
        for(int j=0;j<d[a[i]].size();++j){
            int x=d[a[i]][j];
            g[x].push_back(a[i]/x);
        }
    }
    for(int i=1;i<N;++i){
        static int stack[N];
        static int top;
        while(top){
            add(stack[top--],-1);
        }
        for(int j=(int)g[i].size()-1;~j;--j){
            int x=g[i][j];
            while(top&&ask(x)){
                ans=max(ans,1ll*x*stack[top]*i);
                add(stack[top--],-1);
            }
            add(stack[++top]=x,1);
        }
    }
    cout<<ans<<endl;
}
```

6.21 GCD 问题中验证解的存在性转计数

存在性 (0/1) 转计数是一种不太常见的套路，通常用于数论问题或者动态规划

例如查询数列 a 中是否存在 a_i 与查询的 x 互质，那么可以通过 $\sum_{d|x}\mu(d)*cnt_d$ 计算出数列中与 x 互质的数的个数 (其中 cnt_d 表示数列中有多少个数是 d 的倍数)

6.22 欧拉降幂公式

$$a^b=\begin{cases}a^{b\ mod\ \phi(n)}\ mod\ n & \text{n,a 互质}\\ a^b\ mod\ n & b<\phi(n)\\ b\ mod\ \phi(n)+\phi(n)\ mod\ n & b\geq\phi(n)\end{cases}$$

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

```

```cpp
int T;
ll a,b,m;

ll ph(ll x){
    ll res=x,a=x;
    for(ll i=2;i*i<=x;i++){
        if(a%i==0){
            res=res/i*(i-1);
            while(a%i==0)a/=i;
        }
    }
    if(a>1)res=res/a*(a-1);
    return res;
}

ll qpow(ll a,ll b,ll mod){
    ll r=1;
    while(b){
        if(b&1)r=r*a%mod;
        b>>=1;a=a*a%mod;
    }
    return r;
}

ll f(ll p,ll b){
    if(p==1)return 0;
    if(!b)return 1;
    ll k=ph(p);
    if(__gcd(a,p)==1)return qpow(a,f(k,b-1),p);
    long double t=0;
    for(int i=0;i<b;++i){
        t=pow(a,t);
        if(t>=k)return qpow(a,f(k,b-1)+k,p);
    }
    return qpow(a,t,p);
}

int main(){
    int T;
    scanf("%d",&T);
    while(T--){//a^a^a^a...%m(b nums in total)
        scanf("%lld%lld%lld",&a,&b,&m);
        if(!b)printf("%lld\n",1%m);
        else printf("%lld\n",f(m,b)%m);
    }
    return 0;
}
```

6.23 扩展中国剩余定理

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

```
typedef __int128 ll;
const int N = 1e5+1;

int n;
ll mod[N],r[N];

ll exgcd(ll a,ll b,ll &x,ll &y){
    if(!b){
        x=1,y=0;
        return a;
    }
    ll gcd=exgcd(b,a%b,x,y);
    ll tmp=x;
    x=y;
    y=tmp-a/b*y;
    return gcd;
}

ll EXCRT(){
    ll lcm=mod[1],last_r=r[1];
    ll lcm_a,x,y,k;
    for(int i=2;i<=n;++i){
        lcm_a=((r[i]-last_r)%mod[i]+mod[i]);
        k=lcm;
        ll gcd=exgcd(lcm,mod[i],x,y);
        x=(x*lcm_a/gcd%(mod[i]/gcd)+(mod[i]/gcd))%(mod[i]/gcd);
        lcm=lcm*mod[i]/gcd;
        last_r=(last_r+k*x)%lcm;
    }
    return (last_r%lcm+lcm)%lcm;
}

int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i){
        long long a,b;
        scanf("%lld%lld",&a,&b);
        mod[i]=a;
        r[i]=b;
    }
    long long ans=EXCRT();
    printf("%lld\n",ans);
}
```

6.24 位运算生成下一个含有 k 个 1 的二进制数

```
int getnex(int x){
    int b=x&-x;
    int t=x+b;
    int c=t^x;
    int m=(c>>2)/b;
    return t|m;
```

```
}
```

6.25 年月日与星期的转换

```
int getdata(int y,int m,int d){
    if(m==1||m==2){
        y--;
        m+=12;
    }
    return (d+2*m+3*(m+1)/5+y+y/4-y/100+y/400)%7+1;
}
```

6.26 广义斐波那契数列的循环节

$f(n) = af(n-1) + bf(n-2), f(1) = c, f(2) = d$

求 $f(n)$ mod p 的最小循环节长度。

当 c 是模 p 的二次剩余时，枚举 $n = p - 1$ 的因子。

当 c 是模 p 的二次非剩余时，枚举 $n = (p + 1)(p - 1)$ 的因子。

找到最小的因子 ans，使得

$$\begin{pmatrix} a & b \\ 1 & 0 \end{pmatrix}^{ans} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} (mod\ p)$$

具体证明见 https://blog.csdn.net/ACdreamers/article/details/25616461

二次剩余的定义：

当存在某个 X，使得式子 $X^2 = d(mod\ p)$ 成立时，则称 d 是模 p 的二次剩余。

当对任意 X，式子都不成立时，称 d 是模 p 的二次非剩余。

6.27 MillerRabin 和 PollardRho 判断大质数并分解大数质因数

```
#include<bits/stdc++.h>
#define Rand(x) (1ll*rand()*rand()%(x)+1)
using namespace std;
typedef __int128 ll;

namespace Miller_Rabin{
    const int N = 12;
    const ll p[N]={2,3,5,7,11,13,17,19,61,2333,4567,24251};

    ll qpow(ll a,ll b,ll p){
        ll ans=1;
        for(;b;a=a*a%p,b>>=1)
        if(b&1){
            ans=ans*a%p;
        }
        return ans;
    }

```

```
19      bool check(ll x,ll p){
20          if(x%p==0||qpow(p%x,x-1,x)^1)return true;
21          ll t,k=x-1;
22          while((k^1)&1){
23              t=qpow(p%x,k>>=1,x);
24              if(t^1&&t^x-1)return true;
25              if(!(t^x-1))return false;
26          }
27          return false;
28      }
29
30      bool miller_rabin(ll x){
31          if(x<2)return false;
32          for(int i=0;i<N;++i){
33              if(x==p[i])return true;
34              if(check(x,p[i]))return false;
35          }
36          return true;
37      }
38  }
39
40  namespace Pollard_Rho{
41      vector<long long> ans;//注意存到ans里的质因子都需要是long long范围
42
43      ll gcd(const ll a,const ll b){
44          return b?gcd(b,a%b):a;
45      }
46
47      ll mul(const ll x,const ll y,const ll X){
48          ll k=(1.0L*x*y)/(1.0L*X)-1;
49          ll t=x*y-k*X;
50          while(t<0)t+=X;
51          return t;
52      }
53
54      ll pollard_rho(const ll x,const ll y){
55          int t=0,k=1;
56          ll v0=Rand(x-1),v=v0,d,s=1;
57          while(1){
58              v=(mul(v,v,x)+y)%x;
59              s=mul(s,abs(v-v0),x);
60              if(!(v^v0)||!s)return x;
61              if(++t==k){
62                  if((d=gcd(s,x))^1)return d;
63                  v0=v;
64                  k<<=1;
65              }
66          }
67      }
68
69      void Resolve(ll x){
70          if(x==1)return;
71          if(Miller_Rabin::miller_rabin(x)){
```

```
72          ans.push_back(x);
73          return;
74      }
75      ll y=x;
76      while((y=pollard_rho(x,Rand(x)))==x);
77      while(!(x%y))x/=y;
78      Resolve(x);
79      Resolve(y);
80   }
81}
82
83signed main(){
84   srand(19260817);
85   int T;
86   long long x,res;
87   scanf("%d",&T);
88   while(T--){
89      scanf("%lld",&x);
90      Pollard_Rho::ans.clear();
91      Pollard_Rho::Resolve(x);
92      for(int i=0;i<Pollard_Rho::ans.size();++i){
93         printf("%lld ",Pollard_Rho::ans[i]);
94      }
95      printf("\n");
96   }
97}
```

7 动态规划

7.1 背包

7.1.1 树形依赖背包

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 const int N = 2e2+5;
5 const int M = 2e2+5;
6
7 int n,m,fa[N],w[N];
8 int totw,st[N],ed[N],p[N];
9 int f[N][M];
10 vector<int>g[N];
11
12 void dfs(int x){
13    st[x]=++totw;
14    p[totw]=x;
15    for(int i=0;i<g[x].size();++i)dfs(g[x][i]);
16    ed[x]=totw;
17 }
18
19 int main(){
```

```
    while(scanf("%d%d",&n,&m)!=EOF){
        if(!n&&!m)return 0;
        for(int i=0;i<=n;++i)g[i].clear();
        for(int i=1;i<=n;++i){
            scanf("%d%d",&fa[i],&w[i]);
            g[fa[i]].push_back(i);
        }
        totw=-1;dfs(0);
        for(int j=0;j<=m;++j)f[n+1][j]=0;
        for(int i=n;i>=1;--i){
            int x=p[i];
            for(int j=0;j<=m;++j)
            f[i][j]=max(j?f[st[x]+1][j-1]+w[x]:0,f[ed[x]+1][j]);
        }
        printf("%d\n",f[1][m]);
    }
}
```

7.1.2 可撤销背包

可撤销操作只适用于求方案背包，不适用于求最大值背包。若想要求最大值背包，且想要求出排除每种物品时的最大值背包，则可以通过分治的方法实现。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e3+5;
const int M = 2e3+5;
const int mod = 10;

int n,m,w[N];
int f[M];

int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;++i)scanf("%d",&w[i]);
    f[0]=1;
    for(int i=1;i<=n;++i){
        for(int j=m;j>=w[i];--j)f[j]=(f[j]+f[j-w[i]])%mod;
    }
    for(int i=1;i<=n;++i){
        static int g[M];
        for(int j=0;j<w[i];++j)g[j]=f[j];
        for(int j=w[i];j<=m;++j)g[j]=(f[j]-g[j-w[i]]+mod)%mod;
        for(int j=1;j<=m;++j)printf("%d",g[j]);
        printf("\n");
    }
}
```

7.2 SOS 动态规划

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 20;
typedef long long ll;

ll f[1<<N],a[1<<N];

int main(){
   for(int i=0;i<(1<<N);++i){
      f[i]=a[i];
   }
   for(int i=0;i<N;++i){
      for(int j=0;j<(1<<N);++j)
      if(j&(1<<i)){
         f[j]+=f[j^(1<<i)];
      }
   }
}
```

7.3 动态逆序对 (CDQ 分治)

```cpp
#include<bits/stdc++.h>
#define lowbit(x) (x)&(-x)
using namespace std;
typedef long long ll;
const int N = 2e5+1;

struct nd{
   int x,y,t;
   bool f;
}a[N],b[N];
ll pos[N],tr[N*2+1],taimu,n,m;
ll ans[N];

void add(int x,int d){
   for(int i=x;i<=n;i+=lowbit(i))tr[i]+=d;
}

ll sum(int x){
   int ret=0;
   for(int i=x;i>=1;i-=lowbit(i))ret+=tr[i];
   return ret;
}

bool cmpt(nd a,nd b){
   return a.t<b.t;
}

bool cmpx(nd a,nd b){
   return a.x<b.x;
}
```

```
31
32 int CDQ(int l,int r){
33    if(l==r)return 0;
34    int mid=l+r>>1,cnt=0;
35    for(int i=l;i<=mid;++i)b[++cnt]=a[i],b[cnt].f=0;
36    for(int i=mid+1;i<=r;++i)b[++cnt]=a[i],b[cnt].f=1;
37    sort(b+1,b+cnt+1,cmpx);
38    for(int i=1;i<=cnt;++i){
39       if(!b[i].f)add(b[i].y,1);
40       else ans[b[i].t]+=sum(n)-sum(b[i].y);
41    }
42    for(int i=1;i<=cnt;++i)if(!b[i].f)add(b[i].y,-1);
43    for(int i=cnt;i>=1;--i){
44       if(!b[i].f)add(b[i].y,1);
45       else ans[b[i].t]+=sum(b[i].y);
46    }
47    for(int i=1;i<=cnt;++i)if(!b[i].f)add(b[i].y,-1);
48    CDQ(l,mid);
49    CDQ(mid+1,r);
50 }
51
52 int main(){
53    scanf("%lld%lld",&n,&m);
54    taimu=n+1;
55    for(int i=1;i<=n;++i){
56       scanf("%d",&a[i].y);
57       a[i].x=i;pos[a[i].y]=i;
58    }
59    for(int i=1,x;i<=m;++i)scanf("%d",&x),a[pos[x]].t=--taimu;
60    for(int i=1;i<=n;++i)if(!a[i].t)a[i].t=--taimu;
61    sort(a+1,a+n+1,cmpt);
62    CDQ(1,n);
63    ll ansa=0;
64    for(int i=1;i<=n;++i)ansa+=ans[i];
65    for(int i=n;i>=n-m+1;--i){
66       printf("%lld\n",ansa);
67       ansa-=ans[i];
68    }
69 }
```

7.4 启发式分治

启发式分治是为了解决序列上的一类特殊分治问题，一般的分治每次将区间的中点作为划分点将区间划分成两部分递归下去，而此类特殊分支问题的划分点选取与问题的要求有关，无法自行规定，所以对于每一次分治 (l,r) 进行处理的复杂度也不能高达 $O(r-l+1)$，否则整个问题的复杂度将会退化为 $O(n^2)$。

举例说明：设整个序列 $(a_i >= 0)$ 的权值为 *good* 的区间个数，设区间 *good* 当且仅当区间最大值的两倍大于等于区间和。

此时问题可以用从大到小枚举最大值并划分区间来解决，进而推得利用分治进行解决，对于每个分治的区间 (l,r)，首先找到最大值的位置 p，那么满足条件的子区间 (L,R) 一定满足

$L <= p <= R$。所以我们可以枚举 L 的位置，然后在前缀和中二分出满足条件的 R 的最小位置，再将分治为 $(l, p-1), (p+1, r)$ 两个子区间进行处理，做到单次分治 $O((p-l+1)logn)$ 的时间复杂度，但这样整体复杂度会退化到 $O(n^2logn)$。所以我们利用启发式的思想考虑问题，容易发现在枚举那一步时不一定要枚举 L 的位置，也可以结合枚举 R 的位置二分 L 做到单次分治 $O(min(p-l+1, r-p)logn)$ 的复杂度，从而做到 $O(nlogn)$ 的整体复杂度（一共调用分支函数 n 次而不是 $nlogn$ 次，因为划分点不会在区间分治后得到的两个子区间中出现）。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 3e5+5;

int Case;
int n,a[N];
ll ans,b[N];

struct ST{
   int tot;
   int ls[N<<1],rs[N<<1],mx[N<<1];

   void up(int x){
   }

   int build(int l,int r){
      int x=++tot;
      if(l==r){
         mx[x]=l;
         return x;
      }
      int mid=l+r>>1;
      ls[x]=build(l,mid);
      rs[x]=build(mid+1,r);
      if(a[mx[ls[x]]]>a[mx[rs[x]]])mx[x]=mx[ls[x]];
      else mx[x]=mx[rs[x]];
      return x;
   }

   int getmx(int x,int l,int r,int L,int R){
      if(L<=l&&r<=R)return mx[x];
      int mid=l+r>>1;
      if(R<=mid)return getmx(ls[x],l,mid,L,R);
      if(L>mid)return getmx(rs[x],mid+1,r,L,R);
      int ret1=getmx(ls[x],l,mid,L,R);
      int ret2=getmx(rs[x],mid+1,r,L,R);
      return a[ret1]>a[ret2]?ret1:ret2;
   }

   void init(){
      for(int i=1;i<=tot;++i){
         mx[i]=0;
         ls[i]=rs[i]=0;
```

```
        }
        tot=0;
        int rt=build(1,n);
    }
}T;

void Solve(int l,int r){
    if(l>=r)return;
    int p=T.getmx(1,1,n,l,r);
    int mid=l+r>>1;
    if(p<=mid){
        for(int i=l-1;i<=p-1;++i){
            int pos=lower_bound(b+p,b+r+1,b[i]+a[p]*2)-b;
            ans+=(r-pos+1);
        }
    }
    else{
        for(int i=p;i<=r;++i){
            int pos=upper_bound(b+l-1,b+p,b[i]-a[p]*2)-b-1;
            ans+=(pos-l+2);
        }
    }
    Solve(l,p-1);
    Solve(p+1,r);
}

int main(){
    scanf("%d",&Case);
    while(Case--){
        scanf("%d",&n);
        for(int i=1;i<=n;++i){
            scanf("%d",&a[i]);
            b[i]=b[i-1]+a[i];
        }
        T.init();
        ans=0;
        Solve(1,n);
        printf("%lld\n",ans);
    }
}
```

7.5 随机游走

```
#include <bits/stdc++.h>
#define fo(i,a,b) for(int i=a;i<=b;++i)
#define fod(i,a,b) for(int i=a;i>=b;--i)
#define N 5005
#define LL long long
#define mo 998244353
using namespace std;
LL ny[N],f[N],g[N];
int m1,n,m,t,rd[N],n1;
```

```
LL ksm(LL k,LL n)
{
    LL s=1;
    for(;n;n>>=1,k=k*k%mo) if(n&1) s=s*k%mo;
    return s;
}
LL ap[2*N],fp[2][N],gp[N];
int d[2][N];
vector<int> a1[N];
#define inc(x,v) (x=(x+v)%mo)
void bfs()
{
    memset(fp,0,sizeof(fp));
    LL sum=1;
    fp[0][1]=1,ap[0]=1;
    int p,r;LL v;
    vector<int>::iterator it;
    fo(i,0,2*n-1)
    {
        int i1=i&1;
        fo(j,1,n) fp[1^i1][j]=gp[j]=0;
        fo(k,1,n-1)
        {
            if(fp[i1][k]&&k!=n)
            {
                v=fp[i1][k]*ny[rd[k]]%mo*ny[2]%mo;
                for(it=a1[k].begin();it!=a1[k].end();it++)
                {
                    p=*it;
                    inc(fp[1^i1][p],v);
                }
            }
        }
        fo(k,1,n) gp[k]=fp[1^i1][k];
        fo(k,1,n)
            if(gp[k])
            {
                v=gp[k]*ny[rd[k]]%mo;
                for(it=a1[k].begin();it!=a1[k].end();it++)
                {
                    p=*it;
                    inc(fp[1^i1][p],v);
                }
            }
        sum=(sum-fp[1^i1][n]+mo)%mo;
        ap[i+1]=sum;
        fp[1^i1][n]=0;
    }
}

LL rc[4*N],rp[4*N],le,le1,rw[4*N];
void BM()
{
```

```
63      le=le1=0;
64      memset(rc,0,sizeof(rc));
65      memset(rp,0,sizeof(rp));
66      int lf=0;LL lv=0;
67      fo(i,0,n1)
68      {
69          LL v=0;
70          fo(j,1,le) inc(v,rc[j]*ap[i-j]%mo);
71          if(v==ap[i]) continue;
72          if(le==0)
73          {
74              le=i+1;
75              fo(j,1,le) rc[j]=rp[j]=0;
76              le1=0,lf=i,lv=(ap[i]-v)%mo;
77              continue;
78          }
79          v=(ap[i]-v+mo)%mo;
80          LL mul=v*ksm(lv,mo-2)%mo;
81
82          fo(j,1,le) rw[j]=rc[j];
83
84          inc(rc[i-lf],mul);
85          fo(j,i-lf+1,i-lf+le1) inc(rc[j],(mo-mul*rp[j-(i-lf)]%mo)%mo);
86          if(le<i-lf+le1)
87          {
88              swap(le1,le);
89              le=i-lf+le,lf=i,lv=v;
90              fo(j,1,le1) rp[j]=rw[j];
91          }
92
93          v=0;
94          fo(j,1,le) inc(v,rc[j]*ap[i-j]%mo);
95      }
96  }
97
98  int main()
99  {
100     ny[1]=1;
101     fo(i,2,5000) ny[i]=(-ny[mo%i]*(LL)(mo/i)%mo+mo)%mo;
102     cin>>t;
103     while(t--)
104     {
105         scanf("%d%d",&n,&m);
106         fo(i,1,n)
107         {
108             int x,y;
109             scanf("%d%d",&x,&y);
110             a1[i].clear();
111         }
112         memset(rd,0,sizeof(rd));
113         m1=0;
114         fo(i,1,m)
115         {
```

```
            int x,y;
            scanf("%d%d",&x,&y);
            a1[x].push_back(y),a1[y].push_back(x);
            rd[x]++,rd[y]++;
        }
        n1=2*n;
        bfs();
        BM();
        rc[0]=1,rp[0]=0;
        fo(i,1,le) rc[i]=(mo-rc[i])%mo,rp[i]=0;
        fo(i,0,le)
            fo(j,0,n1) if(i+j<=le) inc(rp[i+j],rc[i]*ap[j]%mo);
        LL ans=0,sv=0;
        fo(i,0,le+n1) inc(ans,rp[i]);
        fo(i,0,le) inc(sv,rc[i]);
        printf("%lld\n",ans*ksm(sv,mo-2)%mo);
    }
}
```

7.6 数位动态规划中的区间异或

问题为求 $\sum_{i=L}^{R} f(i \oplus x)^2$

先将区间分成两个前缀的差，现在只需要考虑 $[0, R] \oplus x$

从高到低枚举最高位 w

设 x' 表示 x 去掉第 w 位的值

设 x^w 表示只保留 x 的 $[0, w-1]$ 位的值

如果 $R < 2^w$，递归到更小的 w 进行处理

如过 $R \geq 2^w$，那么可以考虑 $[0, R] = [0, 2^w - 1] + [2^w, R]$

$[0, 2^w - 1] \oplus x$

$= [0, 2^w - 1] \oplus x^w \oplus (x \oplus x^w)$

$= [0, 2^w - 1] \oplus (x \oplus x^w)$

$= [(x \oplus x^w), (x \oplus x^w) + 2^w - 1]$

（这是因为异或的逆元即为自己本身，$a \oplus x = b$ 等价于 $b \oplus x = a$，所以 $[0, 2^w - 1]$ 中的数在异或后两两互换，所以 $[0, 2^w - 1] \oplus x^w = [0, 2^w - 1]$）

$[2^w, R] \oplus x = [0, R - 2^w] \oplus (x \oplus 2^w)$(此处显然有 $R - 2^w < 2^w$)，最高位变小，这部分递归到更小的 w 用相同的方法处理即可

这样就能把 $[0, R] \oplus x$ 分成 $O(logn)$ 个连续的区间，问题转化为求 $\sum_{i=0}^{R} f(i)^2$

！注意!：对于异或问题，存在单调性的问题，利用数位动态规划的思想根据最高位取值为 0/1 将集合划分为两部分进行分治是很常用的做法。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;
const ll mod = 998244353;
const int M = 30;
```

```
7
8  int Q;
9
10 int get(int l,int r){
11    return 0;
12 }
13
14 int Solve(int R,int w,int x){
15    if(R==-1)return 0;
16    if(w==-1)return get(x,x);//若开始传入的R=R',x=x',则此处R=0,x=x'^R',此处为递归的终止
          条件
17    if((1<<w)>n)return Solve(R,w-1,x);
18    int x1=(x>>w)<<w;//只保留大于等于m的二进制位
19    int l=x1,r=x1+(1<<w)-1;
20    int ret=get(l,r);
21    return (ret+Solve(R-(1<<w),w-1,x^(1<<w)))%mod;
22 }
23
24 int main(){
25    scanf("%d",&Q);
26    for(int i=1,l,r,x;i<=Q;++i){
27       scanf("%d%d%d",&l,&r,&x);
28       printf("%d\n",(Solve(r,M-1,x)-Solve(l-1,M-1,x)+mod)%mod);
29    }
30 }
```

7.7 序列单调递增的最小代价

已知序列 $a_1, a_2, ..., a_n$，对于 a_i，可以花费 b_i 的代价将 a_i 增加 1，花费 c_i 的代价将 a_i 减小 1，求将序列 a 变为非严格单调递增的最小代价。

最朴素的动态规划：设 $f_{i,j}$ 表示只考虑前 i 个位置，且 a_i 的值变为 j 时的最小代价。

$$f_{i,j} = min_{k<=j} f_{i-1,k} + (a_i - j) * c_i \ (a_i \geq j)$$

$$f_{i,j} = min_{k<=j} f_{i-1,k} + (j - a_i) * b_i \ (a_i \leq j)$$

设 $pre_{i,j} = min_{k<=j} f_{i,k}$

则上文中动态规划方程变为：

$$f_{i,j} = pre_{i-1,j} + (a_i - j) * c_i \ (a_i \geq j)$$

$$f_{i,j} = pre_{i-1,j} + (j - a_i) * b_i \ (a_i \leq j)$$

可以将 j 看作横坐标，$f_{i,j}$ 看为纵坐标，则通过归纳法（证明的过程实际就是下述维护函数的过程）可证明该函数是凹函数，斜率单调不降。

因此，可以将解题过程分为两步：

1. 在函数中将 $f_{i-1,j}$ 替换为 $pre_{i-1,j}$，由于 f 为凹函数，因此可以通过在栈中维护函段将所有栈顶斜率大于 0 的线段替换为一个斜率等于 0 的线段。

2. 将函数 pre_{i-1} 与函数 $|a_i - j| * (a_i \geq j\ ?\ c_i\ :\ b_i)$（显然该函数为 V 形）求和，得到 f_i。可以通过在栈中维护斜率变换处和斜率变化值来维护，显然将 pre 与 V 形函数求和只会在 a_i 处插入一个斜率变换值 $b_i + c_i$。（注意同时要维护第一条线段的斜率和最后一条线段的斜率）

重复以上两步 n 次，得到 f_n 的函数图象，然后算出 $f_{n,min\{a_i\}}$ 的函数值（显然，将所有 a_i 变为最小值即可），再通过函数图象递推出所有斜率变化处的函数值，取最小值即可得到答案。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+1;

int T;
int n;
ll a[N],b[N],c[N];
multiset<pair<ll,ll>> s;

int main(){
   scanf("%d",&T);
   while(T--){
      scanf("%d",&n);
      for(int i=1;i<=n;++i)scanf("%lld",&a[i]);
      for(int i=1;i<=n;++i)scanf("%lld",&b[i]);
      for(int i=1;i<=n;++i)scanf("%lld",&c[i]);
      ll lk=0,rk=0;
      s.clear();
      s.insert({0,0});
      ll now=0;
      for(int i=1;i<=n;++i){
         while(rk>0){
            assert(s.size());
            auto p=prev(s.end());
            ll x=p->first,val=p->second;
            s.erase(p);
            rk-=val;
            if(rk<=0){
               s.insert({x,-rk});
               rk=0;
               break;
            }
         }
         lk-=c[i];
         rk+=b[i];
         s.insert({a[i],b[i]+c[i]});
         now+=a[i]*c[i];
      }
      ll ans=now,last=0,k=lk;
      for(auto p:s){
         now+=k*(p.first-last);
         last=p.first;
```

```
            k+=p.second;
            ans=min(ans,now);
        }
        assert(k==rk);
        printf("%lld\n",ans);
    }
}
```

7.8 基于单调序列的数位动态规划

定义序列 $a_{1...n}$ 的权值为 $\sum a_i$

求满足 $l_i \le a_i \le r_i$ 且 a_i 单调不降的所有序列 $a_{1...n}$ 的权值和。

考虑最高位，那么一定存在一个 k，使得 $a[1...k]$ 最高位是 0，$a[k+1...n]$ 最高位是 1，那么可以分成两个子区间去做。

设 $f[i][L][R][LIML][LIMR]$ 表示：在 $i+1...60$ 位已经定下来的情况下（此时 $a[L...R]$ 的 $i+1..60$ 位的值一样），区间 $[L...R]$ 只考虑 $1...i$ 位的答案。

$LIML$ 表示只考虑 $i+1..60$ 位 $a[L...R]$ 是否等于区间中最大的 l_x；

$LIMR$ 表示只考虑 $i+1..60$ 位 $a[L...R]$ 是否等于区间中最小的 r_x

！注意！：对于异或问题，存在单调性的问题，利用数位动态规划的思想根据最高位取值为 0/1 将集合划分为两部分进行分治是很常用的做法。

```
struct atom{
    int sumw,way;
    atom(int _sumw=0,int _way=0){
        sumw=_sumw;
        way=_way;
    }
    friend atom operator +(atom a,atom b){
        return atom((a.sumw+b.sumw)%mod,(a.way+b.way)%mod);
    }
    friend atom operator *(atom a,atom b){
        return atom((a.sumw*1ll*b.way+a.way*1ll*b.sumw)%mod,a.way*1ll*b.way%mod);
    }
};

void Solve(){
    for(int i=1;i<=n;++i){
        mir[i][i]=r[i];
        mal[i][i]=l[i];
        for(int j=i+1;j<=n;++j){
            mir[i][j]=min(mir[i][j-1],r[j]);
            mal[i][j]=max(mal[i][j-1],l[j]);
        }
    }
    for(int l=1;l<=n;++l)
    for(int r=l;r<=n;++r)
    for(int el=0;el<2;++el)
    for(int er=0;er<2;++er){
        f[0][l][r][el][er]=atom(0,1);
```

```
    }
    for(int i=1;i<=60;++i){
        for(int l=1;l<=n;++l)
        for(int r=l;r<=n;++r)
        for(int el=0;el<2;++el)
        for(int er=0;er<2;++er){
            atom ans=atom(0,0);
            for(int k=l;k<=r-1;++k){
                //第i位a[l...k]填0,a[k+1..r]填1
                //如果(i+1,60)位mal[l][k]小于mal[l][r],那么由于LIML=1,所以a[l...r]的(i
                    +1,60)位均等于mal[l][r],所以a[l...k]的(i+1,60)位小于mal[l][k]，所以子
                    区间的LIML一定为0
                int lel=(el&&(mal[l][k]>>i)==(mal[l][r]>>i));
                int ler=(er&&(mir[l][k]>>i)==(mir[l][r]>>i));
                int rel=(el&&(mal[k+1][r]>>i)==(mal[l][r]>>i));
                int rer=(er&&(mir[k+1][r]>>i)==(mir[l][r]>>i));
                //判断第i位a[l...k]填0,a[k+1..r]填1是否可行以及子区间LIML,LIMR的变化
                if(lel&&(mal[l][k]&(1ll<<(i-1))))continue;
                if(rer&&(!(mir[k+1][r]&(1ll<<(i-1)))))continue;
                ler&=((mir[l][k]&(1ll<<(i-1)))==0);
                rel&=((mal[k+1][r]&(1ll<<(i-1)))>0);

                ans=(ans+f[i-1][l][k][lel][ler]*f[i-1][k+1][r][rel][rer]*atom((r-k)*1ll
                    *((1ll<<(i-1))%mod)%mod,1));
            }
            //第i位全填0
            if(el&&(mal[l][r]&(1ll<<(i-1))));
            else{
                int nl=el;
                int nr=er&&((mir[l][r]&(1ll<<(i-1)))==0);
                ans=(ans+f[i-1][l][r][nl][nr]*atom(0,1));
            }
            //第i位全填1
            if(er&&(!(mir[l][r]&(1ll<<(i-1)))));
            else{
                int nl=el&&(mal[l][r]&(1ll<<(i-1)));
                int nr=er;
                ans=(ans+f[i-1][l][r][nl][nr]*atom((r-l+1)*1ll*((1ll<<(i-1))%mod)%mod
                    ,1));
            }
            f[i][l][r][el][er]=ans;
        }
    }
    printf("%d\n",f[60][1][n][1][1].sumw);
}
```

7.9 最小化圆上关键点到同一点距离和

圆上有 n 个关键点，最小化他们到同一点的距离和。

考虑在圆上顺时针移动所求点，在移动的过程中一些关键点到该点距离增加，一些关键点到该点距离减少，一些关键点距离先增加再减少或是先减少再增加。考虑以在移动过程中每个关

键点增加/减少的分界处 (显然这些分界处即为每个关键点在圆上相对的点和自己本身) 为单位枚举移动所求点，那么在移动过程中关键点到该点的距离要么增加，要么减少，只需要维护这两种情况分别有多少个点即可。(显然所求点一定在分界处上, 因为在分界处之间的距离和变化单调，即分界处之间的距离和一定小于两边某一个分界处的距离和)

把 n 个关键点移动到圆上 n 等分位置也可以用相同的方法，先将点按极角排序，默认第 i 个点移动到第 i 个 n 等分位置即可 (即 $a_i-=360*i/n$, 在此次操作后第 i 个等分位置移动到了第 0 个等分位置，a_i 移动到了 $a_i-360*i/n$)

正确的做法是将第 i 个关键点跟第 i 个等分点绑定，然后考虑移动一起所有等分点的过程。

以下是把 n 个关键点移动到圆上 n 等分位置, 最小化距离和的算法:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5+5;

int n;
ll a[N],b[N],q[N*2];
multiset<pair<ll,int>> s;

int main(){
   cin>>n;
   ll r=360000000ll*n;
   ll dec=100000ll*n;
   for(int i=1;i<=n;++i){
      double b;
      cin>>b;
      a[i]=b*dec+0.6;
   }
   sort(a+1,a+n+1);
   ll tot=0;
   for(int i=1;i<=n;++i){
      ll s=(a[i]-360000000ll*(i-1)+r)%r;
      b[i]=s;
   }
   sort(b+1,b+n+1);
   for(int i=n;i>=1;--i)b[i]=b[i]-b[1];
   for(int i=1;i<=n;++i)tot+=min(b[i],r-b[i]);
   ll cnt1=0,cnt2=0;
   ll ans=tot;
   for(int i=1;i<=n;++i){
      ll x1=b[i];
      ll x2=(x1+r/2)%r;
      if(x1<r/2){
         cnt1++;
         s.insert({x1,1});
         s.insert({x2,-1});
      }
      else{
         cnt2++;
         s.insert({x1,1});
```

```
41          s.insert({x2,-1});
42      }
43      q[++q[0]]=x1;
44      q[++q[0]]=x2;
45   }
46   sort(q+1,q+q[0]+1);
47   q[++q[0]]=r;
48   ll last=0;
49   for(auto p:s){
50      tot-=cnt1*(p.first-last);
51      tot+=cnt2*(p.first-last);
52      last=p.first;
53      ans=min(ans,tot);
54      if(p.second==1)cnt1--,cnt2++;
55      else cnt1++,cnt2--;
56   }
57   double res=1.*ans/dec;
58   res=res/180*acos(-1);
59   printf("%.10lf\n",res);
60 }
```

7.10 简化值域只有 01 的动态规划方程

当 DP 方程维度太多且值域为 0/1 时，可以考虑将其中一维放到值域上。

例如 $f[i][j][k]$ 表示考虑前 i 个物品，分给第一个人 j 个，分给第二个人 k 个是否可行；

可以改成 $f[i][j]$ 表示考虑前 i 个物品，分给第一个人 j 个，最多能分给第二个人多少个。(前提是如果分给第二个人 k 个可行，那么 $k-1$ 个也一定可行)

7.11 偏序问题中简化取绝对值或取最大值

$a_i - a_j >= |b_i - b_j|$ 等价于 $a_i - a_j >= b_i - b_j$ 且 $a_i - a_j >= b_j - b_i$

$x >= max(b_i, b_j)$ 等价于 $x >= b_i$ 且 $x >= b_j$

没必要通过讨论 b_i 与 b_j 的大小关系来去掉绝对值或者 max 符号

7.12 对于最小化操作次数的构造题

对于最小化操作次数的构造题，可以考虑优先计算答案的下界，并由此入手构造方案

7.13 数位动态规划中从高位到低位和从低位到高位的区别

求满足以下条件的 (x, y) 整数对数：(适用高位到低位 DP)

1. x $[0, A]$, y $[0, B]$
2. $x xor y$ W

求满足以下条件的 (x, y) 整数对数：(适用低位到高位 DP)

1. x $[0,A]$, y $[0,B]$
2. $xxory$ W
3. $x-y$ K

大多数情况下，从高位到低位进行 DP 都更为便捷。但第二道例题中有加法（也可以看作减法），也就是有进位（也可以看作有借位），此时从低位到高位显然更便捷。

对于第二道例题：设 $f[bit][xA][yB][xKy][yKx][xyw][cxK][cyK]$ 表示在已确定最低的 bit 位的情况下，是否有 x A，是否有 y B，是否有 $x+K$ y，是否有 $y+K$ x，是否有 $xxory$ W，$x+K$ 是否产生了进位，$y+K$ 是否产生了进位时的答案。

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

int T;
int a[31],b[31],k[31],w[31];
ll f[31][2][2][2][2][2][2][2];

ll dfs(int n,bool xa,bool yb,bool xky,bool ykx,bool xyw,int cxk,int cyk){
   if(n==31){
      return xa&&yb&&xky&&ykx&&xyw;
   }
   if(~f[n][xa][yb][xky][ykx][xyw][cxk][cyk]){
      return f[n][xa][yb][xky][ykx][xyw][cxk][cyk];
   }
   ll ret=0;
   for(int x=0;x<2;++x)
   for(int y=0;y<2;++y){
      ret+=dfs(n+1,
      (x==a[n])?xa:(x<a[n]),
      (y==b[n])?yb:(y<b[n]),
      ((x+k[n]+cxk)%2==y)?xky:((x+k[n]+cxk)%2>y),
      ((y+k[n]+cyk)%2==x)?ykx:((y+k[n]+cyk)%2>x),
      ((x^y)==w[n])?xyw:((x^y)<w[n]),
      x+k[n]+cxk>=2,
      y+k[n]+cyk>=2);
   }
   return f[n][xa][yb][xky][ykx][xyw][cxk][cyk]=ret;
}

void trans(int n,int *a){
   for(int i=0;i<31;++i){
      a[i]=n&1;
      n>>=1;
   }
}

int main(){
   cin>>T;
   while(T--){
      int A,B,K,W;
      cin>>A>>B>>K>>W;
```

```
43          trans(A,a);
44          trans(B,b);
45          trans(K,k);
46          trans(W,w);
47          memset(f,-1,sizeof(f));
48          cout<<dfs(0,1,1,1,1,1,0,0)<<endl;
49      }
50  }
```

8 计算几何

8.1 直线与多边形

直线的交

凸多边形的判定

点在多边形内的判定

线段在多边形内的判定

多边形的重心

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   typedef double db;
5   #define zero(x) (((x)>0?(x):-(x))<eps)
6   #define _sign(x) ((x)>eps?1:((x)<-eps?2:0))
7   const int N = 1e3+5;
8   const int offset = 1e8;
9   const db eps = 1e-8;
10
11  struct point{
12      db x,y;
13      point(db _x=0,db _y=0){
14          x=_x;y=_y;
15      }
16  };
17  struct line{
18      point a,b;
19      line(point _a=point(0,0),point _b=point(0,0)){
20          a=_b;b=_b;
21      }
22  };
23
24  //p0->p1与p0->p2的叉积
25  double xmult(point p1,point p2,point p0){
26      return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
27  }
28
29  //直线的交点
30  point intersection(line u,line v){
31      point ret=u.a;
```

```
32      double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
33             /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
34      ret.x+=(u.b.x-u.a.x)*t;
35      ret.y+=(u.b.y-u.a.y)*t;
36      return ret;
37  }
38
39  //判定是否为凸多边形,顶点按顺时针或逆时针给出,允许相邻边共线
40  int is_convex(int n,point* p){
41      int s[3]={1,1,1};
42      for(int i=0;i<n;++i)s[_sign(xmult(p[(i+1)%n],p[(i+2)%n],p[i]))]=0;
43      return s[1]|s[2];
44  }
45
46  //判点在凸多边形内,顶点按顺时针或逆时针给出,在多边形边上返回2
47  int inside_convex(point q,int n,point* p){
48      int s[3]={1,1,1};
49      for(int i=0;i<n;++i)s[_sign(xmult(p[(i+1)%n],q,p[i]))]=0;
50      if(!s[0])return 2;
51      return s[1]|s[2];
52  }
53
54  //判断点a,b是否在直线l的两侧
55  int opposite_side(point p1,point p2,line l){
56      return xmult(l.a,p1,l.b)*xmult(l.a,p2,l.b)<-eps;
57  }
58
59  //判断点是否在线段上
60  inline int dot_online_in(point p,point l1,point l2){
61      return zero(xmult(p,l1,l2))&&(l1.x-p.x)*(l2.x-p.x)<eps&&(l1.y-p.y)*(l2.y-p.y)<
            eps;
62  }
63
64  //判定点是否在任意多边形内,顶点按顺时针或逆时针给出
65  //on_edge表示点在多边形边上时的返回值,offset为多边形坐标上限
66  int inside_polygon(point q,int n,point* p,int on_edge=1){
67      point q2;
68      int i=0,count;
69      while(i<n){
70          for(count=i=0,q2.x=rand()+offset,q2.y=rand()+offset;i<n;++i)
71          if(dot_online_in(q,p[i],p[(i+1)%n]))return on_edge;
72          else if(zero(xmult(q,q2,p[i])))break;
73          else if(opposite_side(p[i],p[(i+1)%n],line(q,q2))&&opposite_side(q,q2,line(p[
                i],p[(i+1)%n])))count++;
74      }
75      return count&1;
76  }
77
78  //判定线段是否在任意多边形内,顶点按顺时针或逆时针给出,与边界相交返回1
79  int inside_polygon(point l1,point l2,int n,point* p){
80      point t[N],tt;
81      int k=0;
82      if(!inside_polygon(l1,n,p)||!inside_polygon(l2,n,p))return 0;
```

```
83      for(int i=0;i<n;++i){
84          if(opposite_side(l1,l2,line(p[i],p[(i+1)%n]))&&opposite_side(p[i],p[(i+1)%n],
                line(l1,l2)))return 0;
85          else if(dot_online_in(l1,p[i],p[(i+1)%n]))t[k++]=l1;
86          else if(dot_online_in(l2,p[i],p[(i+1)%n]))t[k++]=l2;
87          else if(dot_online_in(p[i],l1,l2))t[k++]=p[i];
88      }
89      for(int i=0;i<k;i++)
90      for(int j=i+1;j<k;j++){
91          tt.x=(t[i].x+t[j].x)/2;
92          tt.y=(t[i].y+t[j].y)/2;
93          if(!inside_polygon(tt,n,p))return 0;
94      }
95      return 1;
96  }
97
98  //三点重心
99  point barycenter(point a,point b,point c){
100     line u,v;
101     u.a.x=(a.x+b.x)/2;
102     u.a.y=(a.y+b.y)/2;
103     u.b=c;
104     v.a.x=(a.x+c.x)/2;
105     v.a.y=(a.y+c.y)/2;
106     v.b=b;
107     return intersection(u,v);
108 }
109
110 //多边形重心
111 point barycenter(int n,point* p){
112     point ret,t;
113     db t1=0,t2;
114     ret.x=ret.y=0;
115     for(int i=1;i<n-1;++i)
116     if(fabs(t2=xmult(p[0],p[i],p[i+1]))>eps){
117         t=barycenter(p[0],p[i],p[i+1]);
118         ret.x+=t.x*t2;
119         ret.y+=t.y*t2;
120         t1+=t2;
121     }
122     if(fabs(t1)>eps)ret.x/=t1,ret.y/=t1;
123     return ret;
124 }
```

8.2 凸包

如要求上凸壳或下凸壳，则将排序方式改为以 (x, y) 排序即可。

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef double db;
```

```cpp
const int N = 1e5+1;
const db inf = 1e9;

struct P{
    db x,y,k;
    P(db _x=0,db _y=0,db _k=0){x=_x;y=_y;k=_k;}
    friend bool operator < (P a,P b){
        return a.k<b.k;
    }
    friend db operator * (P a,P b){
        return a.x*b.y-a.y*b.x;
    }
    friend P operator - (P a,P b){
        return P(a.x-b.x,a.y-b.y);
    }
}q[N],a[N];

int n;

bool jud(P a,P b,P t){
    return (t-a)*(t-b)>0;
}

db dis(P a,P b){
    P t=a-b;
    return sqrt(t.x*t.x+t.y*t.y);
}

db work(){
    int R=0;
    for(int i=1;i<=3;++i)q[++R]=a[i];
    for(int i=4;i<=n;++i){
        while(R>1&&jud(q[R],q[R-1],a[i]))R--;
        q[++R]=a[i];
    }
    q[R+1]=q[1];
    db ret=0;
    for(int i=1;i<=R;++i)ret+=dis(q[i],q[i+1]);
    return ret;
}

int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;++i)scanf("%lf%lf",&a[i].x,&a[i].y);
    int now=0;
    a[0].x=a[0].y=inf;
    for(int i=1;i<=n;++i)if(a[i].x<a[now].x||(a[i].x==a[now].x&&a[i].y<a[now].y))now
        =i;
    swap(a[1],a[now]);
    for(int i=n;i;--i)a[i].x-=a[1].x,a[i].y-=a[1].y;
    for(int i=1;i<=n;++i)a[i].k=atan2(a[i].y,a[i].x);
    sort(a+2,a+n+1);
    printf("%.2lf\n",work());
```

```
}
```

8.3 半平面交

```
#include<bits/stdc++.h>
using namespace std;
typedef double db;
const int N = 1e5+1;

int n,cnt;

struct P{
    db x,y;
    P(db _x=0,db _y=0){x=_x;y=_y;}
    friend P operator - (P a,P b){return P(a.x-b.x,a.y-b.y);}
    friend db operator * (P a,P b){return a.x*b.y-b.x*a.y;}
}ans[N];

struct L{
    P a,b;
    db k;
    L(P _a=P(0,0),P _b=P(0,0),db _k=0){a=_a;b=_b;k=_k;}
    friend bool operator < (L a,L b){
        if(a.k==b.k)return (b.b-a.a)*(b.a-a.a)<=0;
        return a.k<b.k;
    }
}a[N],q[N];

P inter(L a,L b){
    db k1,k2,t;
    k1=(a.b-b.a)*(b.b-b.a);
    k2=(b.b-b.a)*(a.a-b.a);
    t=k1/(k1+k2);
    P ans;
    ans.x=a.b.x+(a.a.x-a.b.x)*t;
    ans.y=a.b.y+(a.a.y-a.b.y)*t;
    return ans;
}

bool jud(L a,L b,L t){
    P p=inter(a,b);
    return (t.a-p)*(t.b-p)<0;
}

void hpi(){
    sort(a+1,a+n+1);
    int tot=0;
    for(int i=1;i<=n;++i)if(a[i].k!=a[tot].k)a[++tot]=a[i];
    n=tot;
    int L=0,R=1;
    q[0]=a[1];q[1]=a[2];
    for(int i=3;i<=n;++i){
```

```
49          while(L<R&&jud(q[R],q[R-1],a[i]))R--;
50          while(L<R&&jud(q[L],q[L+1],a[i]))L++;
51          q[++R]=a[i];
52      }
53      while(L<R&&jud(q[R],q[R-1],q[L]))R--;
54      while(L<R&&jud(q[L],q[L+1],q[R]))L++;
55      q[R+1]=q[L];
56      for(int i=L;i<=R;++i)ans[++cnt]=inter(q[i],q[i+1]);
57  }
58
59  int main(){
60      scanf("%d",&n);
61      for(int i=1;i<=n;++i)scanf("%lf%lf%lf%lf",&a[i].a.x,&a[i].a.y,&a[i].b.x,&a[i].b.
            y);
62      a[++n]=L(P(0,0),P(10000,0),0);a[++n]=L(P(10000,0),P(10000,10000),0);
63      a[++n]=L(P(10000,10000),P(0,10000),0);a[++n]=L(P(0,10000),P(0,0),0);
64      for(int i=1;i<=n;++i)a[i].k=atan2((a[i].b.y-a[i].a.y),(a[i].b.x-a[i].a.x));
65      hpi();
66      db res=0;
67      if(cnt>=3){
68          ans[++cnt]=ans[1];
69          for(int i=1;i<=cnt;++i)res+=ans[i]*ans[i+1];
70          res=fabs(res)/2;
71      }
72      printf("%.1lf",res);
73      return 0;
74  }
```

8.4 最小圆覆盖

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   typedef double db;
5   const int N = 1e5+1;
6
7   int n,id[N];
8   db x[N],y[N],R;
9
10  db dis(int i,int j){
11      return sqrt((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j]));
12  }
13
14  bool incircle(int i){
15      return dis(i,0)<=R;
16  }
17
18  int main(){
19      scanf("%d",&n);
20      for(int i=1;i<=n;++i)scanf("%lf%lf",&x[i],&y[i]);
21      for(int i=1;i<=n;++i)id[i]=i;
22      random_shuffle(id+1,id+n+1);
```

```
23      for(int i=1;i<=n;++i)
24      if(!incircle(i)){
25          x[0]=x[i];y[0]=y[i];R=0;
26          for(int j=1;j<i;++j)
27          if(!incircle(j)){
28              x[0]=(x[i]+x[j])/2;y[0]=(y[i]+y[j])/2;R=dis(i,0);
29              for(int k=1;k<j;++k)
30              if(!incircle(k)){
31                  db a1=x[j]-x[i],b1=y[j]-y[i],c1=(a1*a1+b1*b1)/2;
32                  db a2=x[k]-x[i],b2=y[k]-y[i],c2=(a2*a2+b2*b2)/2;
33                  db d=a1*b2-a2*b1;
34                  x[0]=x[i]+(c1*b2-c2*b1)/d;
35                  y[0]=y[i]+(a1*c2-a2*c1)/d;
36                  R=dis(i,0);
37              }
38          }
39      }
40      printf("%.3lf\n",R);
41  }
```