

# Algorithm Templates

BiteTheDust

November 1, 2019

## Contents

<b>1</b>	<b>动态规划</b>	<b>4</b>
1.1	数位 dp	4
1.2	SOSdp	4
1.3	斜率优化 dp	5
1.4	可逆背包	7
1.5	树形 dp	7
1.6	UpDowndp	8
1.7	换根 dp	10
1.8	基环树 dp	11
<b>2</b>	<b>字符串</b>	<b>12</b>
2.1	Manacher	12
2.2	KMP	13
2.3	EXKMP	13
2.4	字符串 hash	14
2.5	后缀自动机	15
2.6	维护 endpos	16
2.7	回文自动机	19
2.8	AC 自动机	20
2.9	序列自动机	23
2.10	最小表示法	23
2.11	Lyndon 分解	23
<b>3</b>	<b>图论</b>	<b>24</b>
3.1	dijkstra	24
3.2	dinic	25
3.3	dij 费用流	28
3.4	zkw 费用流	30
3.5	KM 算法	31
3.6	带花树	32
3.7	并查集	34
3.8	割点	35
3.9	桥	36
3.10	强连通分量	37
3.11	树链剖分	38
3.12	LinkCutTree	40
3.13	任意根 lca	43

3.14	最大独立集	43
3.15	拓扑排序	44
3.16	树分治	45
3.17	DsuOnTree	46
3.18	虚树	48
<b>4</b>	<b>数据结构</b>	<b>50</b>
4.1	莫队分块	50
4.2	带修改莫队分块	52
4.3	树状数组	55
4.4	线段树 (重载 + 最大子列和)	56
4.5	可持久化线段树	57
4.6	动态开点线段树	58
4.7	李超树	60
4.8	整体二分	61
4.9	扩展整体二分	63
4.10	字典树	66
4.11	块状链表	66
<b>5</b>	<b>数论</b>	<b>70</b>
5.1	GCD、LCM、EXGCD	70
5.2	乘法逆元	70
5.3	MillerRobin、PollardRho	71
5.4	快速幂乘	72
5.5	矩阵快速幂	72
5.6	快速阶乘	73
5.7	欧拉函数	76
5.8	欧拉降幂	76
5.9	线性基	77
5.10	线性筛	79
5.11	杜教筛	80
5.12	组合数	82
5.13	模系解码、分数间最小分子	83
5.14	EXCRT	84
5.15	n 次同余	85
5.16	离散对数	87
5.17	BerlekampMassey、ReedsSloane	90
5.18	拉格朗日插值法	94
5.19	高斯消元	96
5.20	Dirichlet	97
5.21	类欧几里德	97
<b>6</b>	<b>其他</b>	<b>98</b>
6.1	01 分数规划	98
6.2	单纯形法	99
6.3	java 大数	102
6.4	mini 读入挂	104
6.5	FFT 快速傅里叶变换	104
6.6	FWT 快速沃尔什变换	105

6.7	NTT 快速数论变换	106
6.8	MTT 快速数论变换 (任意模数)	113
6.9	SG 函数	115
6.10	第二类斯特林数	116
6.11	离散化	117
6.12	STL	118
6.13	星期	119
6.14	博弈	119
<b>7</b>	<b>几何</b>	<b>120</b>
7.1	二维几何	120
7.2	三维几何	138
7.3	三维凸包	142
7.4	多边形交	147
<b>8</b>	<b>工具</b>	<b>148</b>
8.1	对拍	148
8.2	数据生成器	149
8.3	cb 环境配置	149

# 1 动态规划

## 1.1 数位 dp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  long long i,n,m,dp[20][300],a[20],T;
4  long long dfs(int len,bool maxi,int sum)
5  {
6      if(dp[len][sum]!=-1&&maxi==0)return dp[len][sum];
7      long long cnt=0;
8      if(!len)return sum%10==0;
9      int maxn=maxi?a[len]:9;
10     for(int i=0;i<=maxn;i++)cnt+=dfs(len-1,maxi&&i==a[len],sum+i);
11     return maxi?cnt:dp[len][sum]=cnt;
12 }
13 long long div(long long tmp)
14 {
15     memset(a,0,sizeof(a));
16     int p=0;
17     while(tmp)a[++p]=tmp%10,tmp/=10;
18     return dfs(p,1,0);
19 }
20 int main()
21 {
22     memset(dp,-1,sizeof(dp));
23     scanf("%lld",&T);
24     for(i=1;i<=T;i++)
25     {
26         scanf("%lld %lld",&n,&m);
27         printf("Case #%lld: ",i);
28         if(n)printf("%lld\n",div(m)-div(n-1));
29         else printf("%lld\n",div(m));
30     }
31     return 0;
32 }

```

## 1.2 SOSdp

```

1  for(int mask = 0; mask < (1<<N); ++mask){
2      for(int i = 0; i < (1<<N); ++i){
3          if((mask&i) == i){
4              F[mask] += A[i];
5          }
6      }
7  }
8  4^N
9  // iterate over all the masks
10 for (int mask = 0; mask < (1<<n); mask++){
11     F[mask] = A[0];
12     // iterate over all the subsets of the mask
13     for(int i = mask; i > 0; i = (i-1) & mask){

```

```

14     F[mask] += A[i];
15 }
16 }
17 3^N
18 //iterative version
19 for(int mask = 0; mask < (1<<N); ++mask){
20     dp[mask][-1] = A[mask]; //handle base case separately (leaf states)
21     for(int i = 0; i < N; ++i){
22         if(mask & (1<<i))
23             dp[mask][i] = dp[mask][i-1] + dp[mask^(1<<i)][i-1];
24         else
25             dp[mask][i] = dp[mask][i-1];
26     }
27     F[mask] = dp[mask][N-1];
28 }
29
30 //memory optimized, super easy to code.
31 for(int i = 0; i<(1<<N); ++i)
32     F[i] = A[i];
33 for(int i = 0; i < N; ++i) for(int mask = 0; mask < (1<<N); ++mask){
34     if(mask & (1<<i))
35         F[mask] += F[mask^(1<<i)];
36 }
37 N*2^N

```

### 1.3 斜率优化 dp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int i,i0,i1,n,k,ans;
4 long long dp[2005][5005];
5 struct node
6 {
7     long long w,h;
8 }a[5005];
9 struct line
10 {
11     long long k,b;
12 };
13 long long cal(line a,long long pos){return a.k*pos+a.b;}
14 double cross(line a,line b)
15 {
16     return ((double)a.b-b.b)/(b.k-a.k);
17 }
18 bool cmp(node a,node b)
19 {
20     return a.h<b.h;
21 }
22 deque<line>q;
23 map<long long,long long>mp;
24 int main()
25 {

```

```

26     scanf("%d %d",&n,&k);
27     long long sum=0;
28     for(i=1;i<=n;i++)scanf("%lld %lld",&a[i].w,&a[i].h),mp[a[i].h]+=a[i].w,sum+=a[i]
    ].h*a[i].w;
29     i=1;
30     for(auto it:mp)
31     {
32         a[i].h=it.first;
33         a[i].w=it.second;
34         i++;
35     }
36     n=mp.size();
37     sort(a+1,a+1+n,cmp);
38     for(i=1;i<=k;i++)
39     {
40         long long sumw=0;
41         while(!q.empty())q.pop_back();
42         for(i0=1;i0<=n;i0++)
43         {
44             sumw+=a[i0].w;
45             while(q.size()>1)
46             {
47                 line x=q.back();
48                 q.pop_back();
49                 line y=q.back();
50                 if(cross(x,{a[i0].h,dp[i-1][i0-1]-(sumw-a[i0].w)*a[i0].h})<=cross(y,x))
51                 {
52                     continue;
53                 }
54                 else
55                 {
56                     q.push_back(x);
57                     break;
58                 }
59             }
60             if(!(i==1&&i0!=1))q.push_back({a[i0].h,dp[i-1][i0-1]-(sumw-a[i0].w)*a[i0].
            h});
61             while(q.size()>1)
62             {
63                 line x=q.front();
64                 q.pop_front();
65                 line y=q.front();
66                 if(cal(x,sumw)<=cal(y,sumw))continue;
67                 else
68                 {
69                     q.push_front(x);
70                     break;
71                 }
72             }
73             dp[i][i0]=cal(q.front(),sumw);
74         }
75     }
76     printf("%lld\n",sum-dp[k][n]);

```

```

77     return 0;
78 }

```

## 1.4 可逆背包

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int mod=1e9+7;
4  int i,i0,n,m,dp[1005];
5  void add_dp(int x,int cnt,int n)
6  {
7      for(int i=n;i>=(cnt+1)*x;i--)dp[i]-=dp[i-(cnt+1)*x],dp[i]+=mod,dp[i]%=mod;
8      for(int i=x;i<=n;i++)dp[i]+=dp[i-x],dp[i]%=mod;
9  }
10 void del_dp(int x,int cnt,int n)
11 {
12     for(int i=n;i>=x;i--)dp[i]-=dp[i-x],dp[i]+=mod,dp[i]%=mod;
13     for(int i=(cnt+1)*x;i<=n;i++)dp[i]+=dp[i-(cnt+1)*x],dp[i]%=mod;
14 }
15 int main()
16 {
17     scanf("%d",&n);
18     dp[0]=1,m=0;
19     while(n--)
20     {
21         int op,x,y;
22         scanf("%d %d %d",&op,&x,&y); //添加价值为x的y个物品 在1000范围内
23         if(op==1)add_dp(x,y,1000);
24         if(op==2)del_dp(x,y,1000);
25         for(i=0;i<=1000;i++)m^=dp[i];
26     }
27     printf("%d\n",m);
28     return 0;
29 }

```

## 1.5 树形 dp

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int i,i0,n,m,dp[1005][2],ans;
4  vector<int>mp[1005];
5  void dfs(int now,int root)
6  {
7      for(int i:mp[now])
8      {
9          if(i==root)continue;
10         dfs(i,now);
11         if(dp[i][0]+1>dp[now][0])dp[now][1]=dp[now][0],dp[now][0]=dp[i][0]+1;
12         else if(dp[i][0]+1>dp[now][1])dp[now][1]=dp[i][0]+1;
13     }
14 }

```

```

15 void solve(int now,int root);
16 void change_root(int nowa,int nowb)
17 {
18     int tmpa[2]={dp[nowa][0],dp[nowa][1]},tmpb[2]={dp[nowb][0],dp[nowb][1]};
19     int son=dp[nowa][dp[nowb][0]+1]==dp[nowa][0]+1;
20     if(son>dp[nowb][0])dp[nowb][1]=dp[nowb][0],dp[nowb][0]=son;
21     else if(son>dp[nowb][1])dp[nowb][1]=son;
22     solve(nowb,nowa);
23     dp[nowb][0]=tmpb[0],dp[nowb][1]=tmpb[1];
24 }
25 void solve(int now,int root)
26 {
27     ans=max(ans,dp[now][0]);
28     for(auto i:mp[now])
29     {
30         if(i==root)continue;
31         change_root(now,i);
32     }
33 }
34 int main()
35 {
36     int T;
37     scanf("%d",&T);
38     while(T--)
39     {
40         scanf("%d",&n);
41         memset(dp,0,sizeof(dp));
42         for(i=1;i<=n;i++)mp[i].clear();
43         ans=0;
44         for(i=1;i<n;i++)
45         {
46             int a,b;
47             scanf("%d %d",&a,&b);
48             mp[a].push_back(b);
49             mp[b].push_back(a);
50         }
51         dfs(1,-1);
52         solve(1,-1);
53         printf("%d\n",n*2-ans-2);
54     }
55     return 0;
56 }

```

## 1.6 UpDowndp

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int i,i0,n,m,v[100005],dp[100005],ans[100005];
4 struct node
5 {
6     int to,v;
7 };

```



```

8 vector<node>mp[100005];
9 int dfs(int now,int root)
10 {
11     if(dp[now])return dp[now];
12     int cnt=v[now];
13     for(int i=0;i<mp[now].size();i++)
14     {
15         if(mp[now][i].to==root)continue;
16         cnt=max(cnt,dfs(mp[now][i].to,now)+v[now]-mp[now][i].v);
17     }
18     return dp[now]=cnt;
19 }
20 void dfs0(int now,int root,int maxn1,int maxn2,int maxi1,int maxi2)
21 {
22     if(maxn1<0)maxn1=0;
23     if(maxn2<0)maxn2=0;
24     maxn1+=v[now],maxn2+=v[now];
25     if(maxi1==now)maxi1=maxi2,maxn1=maxn2;
26     ans[now]=max(dp[now],maxn1);
27     for(int i=0;i<mp[now].size();i++)
28     {
29         if(mp[now][i].to==root)continue;
30         if(dp[mp[now][i].to]+v[now]-mp[now][i].v>maxn1)maxn2=maxn1,maxn1=dp[mp[now][i].to]+v[now]-mp[now][i].v,maxi1=mp[now][i].to;
31         else if(dp[mp[now][i].to]+v[now]-mp[now][i].v>maxn2)maxn2=dp[mp[now][i].to]+v[now]-mp[now][i].v,maxi2=mp[now][i].to;
32     }
33     for(int i=0;i<mp[now].size();i++)
34     {
35         if(mp[now][i].to==root)continue;
36         dfs0(mp[now][i].to,now,maxn1-mp[now][i].v,maxn2-mp[now][i].v,maxi1,maxi2);
37     }
38     return;
39 }
40 int main()
41 {
42     while(scanf("%d",&n)!=EOF)
43     {
44         for(i=1;i<=n;i++)mp[i].clear(),dp[i]=0;
45         for(i=1;i<=n;i++)scanf("%d",&v[i]);
46         for(i=1;i<n;i++)
47         {
48             int a,b,c;
49             scanf("%d %d %d",&a,&b,&c);
50             mp[a].push_back({b,c});
51             mp[b].push_back({a,c});
52         }
53         dfs(1,-1);
54         dfs0(1,-1,0,0,0,0);
55         for(i=1;i<=n;i++)
56         {
57             printf("%d",ans[i]);
58             if(i==n)printf("\n");

```

```

59         else printf(" ");
60     }
61 }
62 return 0;
63 }

```

## 1.7 换根 dp

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int i,i0,n,m,v[100005],dp[100005],ans[100005];
4  struct node{int to,d;};
5  vector<node>mp[100005];
6  void dfs(int now,int root)
7  {
8      dp[now]=v[now];
9      for(auto it:mp[now])
10     {
11         if(it.to==root)continue;
12         dfs(it.to,now);
13         dp[now]=max(dp[now],dp[it.to]+v[now]-it.d);
14     }
15 }
16 void solve(int now,int root);
17 void change_root(int nowa,int nowb,int d)
18 {
19     int tmpa=dp[nowa],tmpb=dp[nowb];
20     dp[nowa]=v[nowa];
21     for(auto it:mp[nowa])
22     {
23         if(it.to==nowb)continue;
24         dp[nowa]=max(dp[nowa],dp[it.to]+v[nowa]-it.d);
25     }
26     dp[nowb]=max(dp[nowb],dp[nowa]+v[nowb]-d);
27     solve(nowb,nowa);
28     dp[nowa]=tmpa,dp[nowb]=tmpb;
29 }
30 void solve(int now,int root)
31 {
32     ans[now]=dp[now];
33     for(int i=0;i<mp[now].size();i++)
34     {
35         if(mp[now][i].to==root)continue;
36         change_root(now,mp[now][i].to,mp[now][i].d);
37     }
38 }
39 int main()
40 {
41     while(scanf("%d",&n)!=EOF)
42     {
43         for(i=1;i<=n;i++)mp[i].clear(),dp[i]=0;
44         for(i=1;i<=n;i++)scanf("%d",&v[i]);

```

```

45     for(i=1;i<n;i++)
46     {
47         int a,b,c;
48         scanf("%d %d %d",&a,&b,&c);
49         mp[a].push_back({b,c});
50         mp[b].push_back({a,c});
51     }
52     dfs(1,-1);
53     solve(1,-1);
54     for(i=1;i<=n;i++)
55     {
56         printf("%d",ans[i]);
57         if(i==n)printf("\n");
58         else printf(" ");
59     }
60 }
61 return 0;
62 }

```

## 1.8 基环树 dp

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  long long i,i0,n,m,v[1000005],pre[1000005],dp[1000005][2][2],ans;
4  vector<int>mp[1000005];
5  vector<pair<int,int> >q;
6  int fin(int x)
7  {
8      int tmp=x,tmp2;
9      while(pre[tmp]!=tmp)tmp=pre[tmp];
10     while(x!=tmp)tmp2=pre[x],pre[x]=tmp,x=tmp2;
11     return tmp;
12 }
13 void uni(int x,int y){if(fin(x)!=fin(y))pre[fin(y)]=fin(x);}
14 void dfs(int now,int ex,int root)
15 {
16     dp[now][0][1]=dp[now][1][1]=v[now];
17     for(auto i:mp[now])
18     {
19         dfs(i,ex,root);
20         for(int i0=0;i0<2;i0++)
21         {
22             dp[now][i0][0]+=max(dp[i][i0][0],dp[i][i0][1]);
23             dp[now][i0][1]+=dp[i][i0][0];
24         }
25     }
26     if(ex==now)dp[now][1][0]=dp[now][0][1]=0;
27 }
28 int main()
29 {
30     scanf("%lld",&n);
31     ans=0;

```

```

32     for(i=1;i<=n;i++)pre[i]=i;
33     for(i=1;i<=n;i++)
34     {
35         scanf("%lld %lld",&v[i],&m);
36         if(fin(m)==fin(i))q.push_back({i,m});
37         else uni(m,i),mp[m].push_back(i);
38     }
39     for(i=0;i<q.size();i++)
40     {
41         dfs(q[i].first,q[i].second,q[i].first);
42         ans+=max(dp[q[i].first][1][0],max(dp[q[i].first][0][0],dp[q[i].first][0][1]))
43         ;
44     }
45     printf("%lld\n",ans);
46     return 0;
47 }

```

## 2 字符串

### 2.1 Manacher

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=100005;
4  char ma[maxn*2],s[maxn];
5  int mp[maxn*2];
6  void Manacher(char *s,int len)
7  {
8      int l=0;
9      ma[l++]='$',ma[l++]='#';
10     for(int i=0;i<len;i++)ma[l++]=s[i],ma[l++]='#';
11     ma[l]=0;
12     int mx=0,id=0;
13     for(int i=0;i<l;i++)
14     {
15         mp[i]=mx>i?min(mp[id*2-i],mx-i):1;
16         while(ma[i+mp[i]]==ma[i-mp[i]])mp[i]++;
17         if(i+mp[i]>mx)mx=i+mp[i],id=i;
18     }
19 }
20 int main()
21 {
22     while(scanf("%s",s)!=EOF)
23     {
24         int len=strlen(s);
25         Manacher(s,len);
26         int ans=0;
27         for(int i=0;i<2*len+2;i++)ans=max(ans,mp[i]-1);
28         printf("%d\n",ans);
29     }
30 }

```

## 2.2 KMP

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int a[1000005],b[100005],nex[100005],i,i0,n,m,T,ans,k;
4  void cal_nex(int *str,int len)
5  {
6      nex[0]=-1;
7      for (int q=1,k=-1;q<len;q++)
8      {
9          while(k!=-1&&str[k+1]!=str[q])k=nex[k];
10         if (str[k+1]==str[q])k++;
11         nex[q]=k;
12     }
13 }
14 int KMP(int *str,int slen,int *ptr,int plen)
15 {
16     cal_nex(ptr,plen);
17     for(int i=0,k=-1;i<slen;i++)
18     {
19         while(k>-1&&ptr[k+1]!=str[i])k=nex[k];
20         if(ptr[k+1]==str[i])k++;
21         if(k==plen-1)return i-plen+2;
22     }
23     return -1;
24 }
25 int main()
26 {
27     scanf("%d",&T);
28     while(T--)
29     {
30         scanf("%d %d",&n,&m);
31         for(i=0;i<n;i++)scanf("%d",&a[i]);
32         for(i=0;i<m;i++)scanf("%d",&b[i]);
33         printf("%d\n",KMP(a,n,b,m));
34     }
35     return 0;
36 }

```

## 2.3 EXKMP

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int n,T,nex[200005],ex[200005];
4  char s[200005];
5  void cal_nex(char *str,int len)
6  {
7      nex[0]=len,nex[1]=0;
8      while(nex[1]+1<len&&str[nex[1]]==str[nex[1]+1])nex[1]++;
9      for(int i=2,i0=1;i<len;i++)
10     {
11         if(i+nex[i-i0]<nex[i0]+i0)nex[i]=nex[i-i0];

```

```

12     else
13     {
14         nex[i]=max(0,nex[i0]+i0-i);
15         while(i+nex[i]<len&&str[nex[i]]==str[i+nex[i]])nex[i]++;
16         i0=i;
17     }
18 }
19 }
20 void EX_KMP(char *str,int slen,char *ptr,int plen)
21 {
22     cal_nex(ptr,plen);
23     ex[0]=0;
24     while(ex[0]<slen&&ex[0]<plen&&str[ex[0]]==ptr[ex[0]])ex[0]++;
25     for(int i=1,i0=0;i<slen;i++)
26     {
27         if(i+nex[i-i0]<ex[i0]+i0)ex[i]=nex[i-i0];
28         else
29         {
30             ex[i]=max(0,ex[i0]+i0-i);
31             while(i+ex[i]<slen&&ex[i]<plen&&str[i+ex[i]]==ptr[ex[i]])ex[i]++;
32             i0=i;
33         }
34     }
35 }
36 int main()
37 {
38     scanf("%d",&T);
39     while(T--)
40     {
41         scanf("%d %s",&n,s);
42         EX_KMP(s,n,s,n);
43         int ans=0;
44         for(int i=0;i<n;i++)ans+=ex[i],ans%=10007;
45         printf("%d\n",ans);
46     }
47     return 0;
48 }

```

## 2.4 字符串 hash

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int i,i0,n,k;
4  char s[100005];
5  const int pr=769;
6  unsigned long long dic[100005],pre[100005];
7  unsigned long long get(int l,int r){return pre[r]-pre[l-1]*dic[r-l+1];}
8  map<unsigned long long,int>mp;
9  int main()
10 {
11     scanf("%d %d %s",&n,&k,s+1);
12     dic[0]=1;

```

```

13     for(i=1;i<=n;i++)dic[i]=dic[i-1]*pr;
14     for(i=1;i<=n;i++)pre[i]=pre[i-1]*pr+(s[i]-'a');
15     for(i=1;i+k-1<=n;i++)mp[get(i,i+k-1)]++;
16     int ans=0;
17     for(auto i:mp)ans=max(ans,i.second);
18     printf("%d\n",ans);
19     return 0;
20 }

```

## 2.5 后缀自动机

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=1000005,ALP=26;
4  struct SuffixAutomaton
5  {
6      int next[maxn*2][ALP],p,cnt[maxn*2];
7      int fail[maxn*2];//表示节点i的后缀链接
8      int len[maxn*2];//表示节点i表示的等价集合的最长子串长度
9      int last;//指向新添加一个字母后所形成的最长子串所指向的节点
10     void init()
11     {
12         last=p=0,fail[0]=-1;
13         newnode(0);
14     }
15     int newnode(int l)
16     {
17         for(int i=0;i<ALP;i++)next[p][i]=-1;
18         len[p]=l,cnt[p]=0;
19         return p++;
20     }
21     void add(int c)
22     {
23         int cur=newnode(len[last]+1),t=last;
24         cnt[cur]++;
25         for(;t!=-1&&next[t][c]==-1;t=fail[t])next[t][c]=cur;
26         if(t==-1)fail[cur]=0;
27         else
28         {
29             int q=next[t][c];
30             if(len[t]+1==len[q])fail[cur]=q;
31             else
32             {
33                 int clone=newnode(len[t]+1);
34                 fail[clone]=fail[q];
35                 for(int i=0;i<26;i++)next[clone][i]=next[q][i];
36                 for(;t!=-1&&next[t][c]==q;t=fail[t])next[t][c]=clone;
37                 fail[q]=fail[cur]=clone;
38             }
39         }
40     }
41     last=cur;

```

```

42     }
43     int lcs(char* str,int slen)
44     {
45         int now=0,l=0,ans=0,sum=0;
46         for(int i=0;i<slen;i++)
47         {
48             int c=str[i]-'a';
49             while(now&&next[now][c]==-1)now=fail[now],l=len[now];
50             if(next[now][c]!=-1)now=next[now][c],l++;
51             ans=max(ans,l);
52             sum+=ans;
53         }
54         return sum;
55     }
56     int rk[maxn*2];
57     void sort()
58     {
59         for(int i=0;i<p;i++)rk[i]=i;
60         std::sort(rk,rk+p,[&](int x,int y){return len[x]>len[y];});
61         for(int i=0;i<p;i++)if(fail[rk[i]]!=-1)cnt[fail[rk[i]]]+=cnt[rk[i]];
62     }
63 }sam;
64 char s[maxn];
65 int main()
66 {
67     scanf("%s",s);
68     sam.init();
69     for(int i=0;s[i]!='\0';i++)sam.add(s[i]-'a');
70     int ans=0;
71     for(int i=0;i<sam.p;i++)if(sam.cnt[i]>1)ans=max(ans,sam.cnt[i]*sam.len[i]);
72     printf("%d\n",ans);
73     return 0;
74 }

```

## 2.6 维护 endpos

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=100005,ALP=26,lm=20;
4  int n,m;
5  namespace SegmentTree
6  {
7      #define mid (l+r)/2
8      int cnt=0;
9      struct node
10     {
11         int lson,rson,sum;
12     }tree[maxn*lm*2];
13     int new_node()
14     {
15         int p=++cnt;
16         tree[p].lson=tree[p].rson=0;

```



```

17     tree[p].sum=0;
18     return p;
19 }
20 void c_tree(int l,int r,int p,int x)
21 {
22     tree[p].sum++;
23     if(l!=r)
24     {
25         if(x<=mid)
26         {
27             tree[p].lson=new_node();
28             c_tree(l,mid,tree[p].lson,x);
29         }
30         else
31         {
32             tree[p].rson=new_node();
33             c_tree(mid+1,r,tree[p].rson,x);
34         }
35     }
36 }
37 int m_tree(int l,int r,int pa,int pb)
38 {
39     if(pa&&pb)
40     {
41         int t=new_node();
42         tree[t].lson=m_tree(l,mid,tree[pa].lson,tree[pb].lson);
43         tree[t].rson=m_tree(mid+1,r,tree[pa].rson,tree[pb].rson);
44         tree[t].sum=tree[pa].sum+tree[pb].sum;
45         return t;
46     }
47     else return pa|pb;
48 }
49 int q_tree(int l,int r,int p,int k)
50 {
51     if(tree[p].sum<k)return -1;
52     if(l==r)return l;
53     else
54     {
55         if(tree[tree[p].lson].sum>=k)return q_tree(l,mid,tree[p].lson,k);
56         else return q_tree(mid+1,r,tree[p].rson,k-tree[tree[p].lson].sum);
57     }
58 }
59 }
60 struct SuffixAutomaton
61 {
62     int next[maxn*2][ALP],p;
63     int root[maxn*2];
64     int fail[maxn*2]; //表示节点i的后缀链接
65     int len[maxn*2]; //表示节点i表示的等价集合的最长子串长度
66     int last; //指向新添加一个字母后所形成的最长子串所指向的节点
67     void init()
68     {
69         last=p=0,fail[0]=-1;

```

```

70     SegmentTree::cnt=0;
71     newnode(0);
72 }
73 int newnode(int l)
74 {
75     for(int i=0;i<ALP;i++)next[p][i]=-1;
76     len[p]=1;
77     root[p]=SegmentTree::new_node();
78     return p++;
79 }
80 int d[maxn];
81 void add(int c,int pos)
82 {
83     int cur=newnode(len[last]+1),t=last;
84     d[pos]=cur;
85     SegmentTree::c_tree(1,n,root[cur],pos);
86     for(;t!=-1&&next[t][c]==-1;t=fail[t])next[t][c]=cur;
87     if(t==-1)fail[cur]=0;
88     else
89     {
90         int q=next[t][c];
91         if(len[t]+1==len[q])fail[cur]=q;
92         else
93         {
94             int clone=newnode(len[t]+1);
95             fail[clone]=fail[q];
96             for(int i=0;i<26;i++)next[clone][i]=next[q][i];
97             for(;t!=-1&&next[t][c]==q;t=fail[t])next[t][c]=clone;
98             fail[q]=fail[cur]=clone;
99         }
100     }
101     last=cur;
102 }
103 int rk[maxn*2];
104 int Fa[maxn*2][lm+1];
105 void count()
106 {
107     for(int i=0;i<p;i++)rk[i]=i;
108     sort(rk,rk+p,[&](int x,int y){return len[x]>len[y]});
109     for(int i=0;i<p-1;i++)root[fail[rk[i]]]=SegmentTree::m_tree(1,n,root[fail[rk[i]]],root[rk[i]]);
110     for(int i=0;i<p;i++)Fa[i][0]=fail[i];
111     for(int k=1;k<=lm;k++)for(int i=0;i<p;i++)Fa[i][k]=Fa[Fa[i][k-1]][k-1];
112 }
113 int find(int l,int r)
114 {
115     int u=d[r],length=r-l+1;
116     if(len[fail[u]]+1>length)
117     {
118         for(int k=lm;k>=0;k--)if(len[fail[Fa[u][k]]]+1>length)u=Fa[u][k];
119         u=fail[u];
120     }
121     return root[u];

```

```

122     }
123 }sam;
124 char s[maxn];
125 int main()
126 {
127     int T;
128     scanf("%d",&T);
129     while(T--)
130     {
131         scanf("%d %d %s",&n,&m,s);
132         sam.init();
133         for(int i=0;i<n;i++)sam.add(s[i]-'a',i+1);
134         sam.count();
135         while(m--)
136         {
137             int l,r,k;
138             scanf("%d %d %d",&l,&r,&k);
139             int z=SegmentTree::q_tree(1,n,sam.find(l,r),k);
140             if(z!=-1)z+=l-r;
141             printf("%d\n",z);
142         }
143     }
144     return 0;
145 }

```

## 2.7 回文自动机

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=300005,ALP=26;
4  string t;
5  struct PalindromeAutomaton
6  {
7      int next[maxn][ALP],fail[maxn],s[maxn],n,p;
8      int cnt[maxn];//表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()
9      函数跑一遍以后才是正确的）
10     int num[maxn];//表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
11     int len[maxn];//len[i]表示节点i表示的回文串的长度（一个节点表示一个回文串），当前有p个
12     节点
13     int last;//指向新添加一个字母后所形成的最长回文串表示的节点
14     //num[last]为最后添加的一个字母所增加的回文子串个数。
15     int newNode(int l)
16     {
17         for(int i=0;i<ALP;i++)next[p][i]=0;
18         cnt[p]=num[p]=0,len[p]=1;
19         return p++;
20     }
21     void init()
22     {
23         p=last=n=0,s[n]=-1,fail[0]=1;
24         newNode(0),newNode(-1);
25     }

```

```

24     int get_fail(int x)
25     {
26         while(s[n-len[x]-1]!=s[n])x=fail[x];
27         return x;
28     }
29     void add(int c)
30     {
31         s[++n]=c;
32         int cur=get_fail(last);
33         if(!next[cur][c])
34         {
35             int now=newnode(len[cur]+2);
36             fail[now]=next[get_fail(fail[cur])][c];
37             next[cur][c]=now,num[now]=num[fail[now]] + 1;
38         }
39         last=next[cur][c];
40         cnt[last]++;
41     }
42     void count()
43     {
44         for(int i=p-1;i>=0;i--)cnt[fail[i]]+=cnt[i];
45     }
46     void dfs(int now)
47     {
48         for(int i=0;i<ALP;i++)
49         {
50             if(next[now][i])
51             {
52                 t+='a'+i;
53                 cout<<t<<endl;
54                 dfs(next[now][i]);
55                 t=t.substr(0,t.size()-1);
56             }
57         }
58     }
59 }pam;
60 char s[300005];
61 int main()
62 {
63     while(scanf("%s",s)!=EOF)
64     {
65         pam.init();
66         int len=strlen(s);
67         for(int i=0;i<len;i++)pam.add(s[i]-'a');
68         pam.dfs(0);
69     }
70     return 0;
71 }

```

## 2.8 AC 自动机

```

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

```

```
2 using namespace std;
3 const int MAXNODE=500005,MAXCHAR=26,MAXN=1000005,MAXM=10005;
4 int ans,T,n,success[MAXM];
5 char s[MAXN],t[MAXN];
6 struct AC_automaton
7 {
8     int ch[MAXNODE][MAXCHAR];
9     int sz;
10    int val[MAXNODE];
11    int fail[MAXNODE];
12    int last[MAXNODE];
13    int q[MAXNODE];
14    int head,tail;
15    void clear()
16    {
17        for(int i=0;i<MAXCHAR;++i)
18        {
19            ch[0][i]=0;
20        }
21        sz=head=tail=0;
22    }
23    int new_node()
24    {
25        sz++;
26        for(int i=0;i<MAXCHAR;++i)
27        {
28            ch[sz][i]=0;
29        }
30        fail[sz]=last[sz]=val[sz]=0;
31        return sz;
32    }
33    void insert(char t[])
34    {
35        int root=0;
36        for(int i=0; t[i]!='\0'; ++i)
37        {
38            if(!ch[root][t[i]-'a'])
39            {
40                ch[root][t[i]-'a']=new_node();
41            }
42            root=ch[root][t[i]-'a'];
43        }
44        ++val[root];
45    }
46    void getfail()
47    {
48        for(int i=0; i<MAXCHAR; i++)
49        {
50            if(ch[0][i])
51            {
52                q[++tail]=ch[0][i];
53            }
54        }
```

```
55     while(head!=tail)
56     {
57         int root=q[++head];
58         for(int i=0; i<MAXCHAR; i++)
59         {
60             if(!ch[root][i])
61             {
62                 ch[root][i]=ch[fail[root]][i];
63                 continue;
64             }
65             int child=ch[root][i];
66             fail[child]=ch[fail[root]][i];
67             last[child]=val[fail[child]]?fail[child]:last[fail[child]];
68             q[++tail]=child;
69         }
70     }
71 }
72 void addnum(int root)
73 {
74     if(root)
75     {
76         ans+=val[root];
77         addnum(last[root]);
78     }
79 }
80 void find(char s[])
81 {
82     int root=0;
83     for(int i=0; s[i]!='\0'; i++)
84     {
85         root=ch[root][s[i]-'a'];
86         if(val[root]) addnum(root);
87         else if(val[last[root]]) addnum(last[root]);
88     }
89 }
90 }AC;
91
92 int main()
93 {
94     scanf("%d",&T);
95     while(T--)
96     {
97         AC.clear();
98         ans=0;
99         scanf("%d",&n);
100         for(int i=1;i<=n;++i)
101         {
102             scanf("%s",&t);
103             AC.insert(t);
104             success[i]=0;
105         }
106         AC.getfail();
107         scanf("%s",&s);
```

```

108     AC.find(s);
109     printf("%d\n",ans);
110 }
111 return 0;
112 }

```

## 2.9 序列自动机

```

1  int nex[MAX][26];
2  void work(char *s,int len)
3  {
4      mem(nex[len],0);
5      for(int i=len;i;i--)
6      {
7          for(int j=0;j<26;j++)
8          {
9              nex[i-1][j]=nex[i][j];
10         }
11         nex[i-1][s[i]-'a']=i;
12     }
13 }

```

## 2.10 最小表示法

```

1  int getmin(char *s){
2      int n=strlen(s);
3      int i=0,j=1,k=0,t;
4      while(i<n && j<n && k<n){
5          t=s[(i+k)%n]-s[(j+k)%n];
6          if (!t) k++;
7          else{
8              if (t>0) i+=k+1;
9              else j+=k+1;
10             if (i==j) j++;
11             k=0;
12         }
13     }
14     return i<j?i:j;
15 }

```

## 2.11 Lyndon 分解

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  #define Abigail inline void
5  typedef long long LL;
6
7  const int N=1<<20;
8

```

```

9  char s[N+9];
10 int n,x[N+9],ts;
11
12 void Lyndon_word(){
13     int j,k;
14     for (int i=1;i<=n;){
15         for (j=i,k=i+1;k<=n&&s[k]>=s[j];++k) //一直循环直到串结束或情况3出现
16             s[k]>s[j]?j=i:++j; //情况1与情况2
17         for (;i<=j;i+=k-j)
18             x[++ts]=i+k-j-1; //记录右端点
19     }
20 }
21
22 Abigail into(){
23     scanf("%s",s+1);
24     n=strlen(s+1);
25 }
26
27 Abigail work(){
28     Lyndon_word();
29 }
30
31 Abigail outo(){
32     for (int i=1;i<=ts;++i)
33         printf("%d ",x[i]);
34 }
35
36 int main(){
37     into();
38     work();
39     outo();
40     return 0;
41 }

```

## 3 图论

### 3.1 dijkstra

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=200005;
4  int i,i0,n,m,r;
5  long long d[maxn];
6  bool vis[maxn];
7  struct edge
8  {
9      int to;
10     long long v;
11     bool operator<(edge a)const{return a.v<v;}
12 };
13 vector<edge>mp[maxn];
14 priority_queue<edge>q;

```



```

15 int main()
16 {
17     scanf("%d",&n);
18     for(i=1;i<=n;i++)
19     {
20         int a,b,c,d;
21         scanf("%d %d %d %d",&a,&b,&c,&d);
22         if(i==n)r=a;
23         mp[i].push_back({d,a});
24         for(int i0=1;i0*i0<=c;i0++)
25         {
26             if(c%i0==0)
27             {
28                 if(i+i0<=n)mp[i].push_back({i+i0,a+b});
29                 if(i+c/i0<=n)mp[i].push_back({i+c/i0,a+b});
30             }
31         }
32     }
33     int s=1,t=n;
34     for(i=1;i<=n;i++)d[i]=(long long)INT_MAX*(n+5);
35     d[s]=0;
36     q.push({s,d[s]});
37     while(!q.empty())
38     {
39         int to=q.top().to;
40         long long v=q.top().v;
41         q.pop();
42         if(!vis[to])
43         {
44             vis[to]=1;
45             for(auto i:mp[to])
46             {
47                 if(d[i.to]>v+i.v)
48                 {
49                     d[i.to]=v+i.v;
50                     q.push({i.to,d[i.to]});
51                 }
52             }
53         }
54     }
55     d[t]+=r;
56     if(d[t]>=(long long)INT_MAX*(n+5))d[t]=-1;
57     printf("%lld\n",d[t]);
58     return 0;
59 }

```

### 3.2 dinic

```

1 #include<cstdio>
2 #include<cstring>
3 #include<queue>
4

```

```

5 using namespace std;
6
7 const int VM=2520;
8 const int EM=500010;
9 const int INF=0x3f3f3f3f;
10
11 struct Edge{
12     int u,v,nxt;
13     int flow;
14 }edge[EM<<1];
15
16 int n,m,cnt,head[VM];
17 int src,des,dep[VM];
18
19 void addedge(int cu,int cv,int cf){
20     edge[cnt].u=cu; edge[cnt].v=cv; edge[cnt].flow=cf;
21     edge[cnt].nxt=head[cu]; head[cu]=cnt++;
22
23     edge[cnt].u=cv; edge[cnt].v=cu; edge[cnt].flow=0;
24     edge[cnt].nxt=head[cv]; head[cv]=cnt++;
25 }
26
27 int dir[4][2]={{-1,0},{1,0},{0,-1},{0,1}};
28
29 int legal(int i,int j,int k){
30     int x=i+dir[k][0];
31     int y=j+dir[k][1];
32     return x>=1 && x<=n && y>=1 && y<=m;
33 }
34
35 int BFS(){
36     queue<int> q;
37     while(!q.empty())
38         q.pop();
39     memset(dep,-1,sizeof(dep));
40     dep[src]=0;
41     q.push(src);
42     while(!q.empty()){
43         int u=q.front();
44         q.pop();
45         for(int i=head[u];i!=-1;i=edge[i].nxt){
46             int v=edge[i].v;
47             if(edge[i].flow>0 && dep[v]==-1){
48                 dep[v]=dep[u]+1;
49                 q.push(v);
50             }
51         }
52     }
53     return dep[des]!=-1;
54 }
55 int DFS(int u,int minx){
56     int ans=0;
57     if(u==des)

```

```

58     return minx;
59     for(int i=head[u];i!=-1 && ans<minx;i=edge[i].nxt){
60         int v=edge[i].v;
61         if(edge[i].flow>0 && dep[v]==dep[u]+1){
62             int tmp=min(edge[i].flow,minx-ans);
63             tmp=DFS(v,tmp);
64             ans+=tmp;
65             edge[i].flow-=tmp;
66             edge[i^1].flow+=tmp;
67         }
68     }
69     if(!ans)
70         dep[u]=-2;
71     return ans;
72 }
73
74 int Dinic(){
75     int ans=0,tmp;
76     while(BFS()){
77         while(1){
78             tmp=DFS(src,INF);
79             if(tmp==0)
80                 break;
81             ans+=tmp;
82         }
83     }
84     return ans;
85 }
86
87 int main()
88 {
89     while(~scanf("%d%d",&n,&m))
90     {
91         cnt=0;
92         memset(head,-1,sizeof(head));
93         int x,sum=0;
94         src=0; des=n*m+1;
95         for(int i=1;i<=n;i++){
96             for(int j=1;j<=m;j++){
97                 scanf("%d",&x);
98                 sum+=x;
99                 if((i+j)%2==0){
100                     addedge(src,(i-1)*m+j,x);
101                     for(int k=0;k<4;k++){
102                         if(legal(i,j,k))
103                             addedge((i-1)*m+j,(i+dir[k][0]-1)*m+(j+dir[k][1]),INF);
104                     }
105                 }else{
106                     addedge((i-1)*m+j,des,x);
107                     for(int k=0;k<4;k++){
108                         if(legal(i,j,k))
109                             addedge((i+dir[k][0]-1)*m+(j+dir[k][1]),(i-1)*m+j,INF);
110                     }

```

```

111     }
112 }
113 int maxflow=Dinic();
114 printf("%d\n",sum-maxflow);
115 }
116 return 0;
117 }

```

### 3.3 dij 费用流

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef pair<int,int>P;
4  const int MAX_V=5005;
5  const int INF=0x3f3f3f3f;
6  struct edge {
7      int to,cap,cost,rev;
8  };
9  int V;
10 vector<edge>G[MAX_V];
11 int h[MAX_V];
12 int dist[MAX_V];
13 int prevv[MAX_V],preve[MAX_V];
14 void add_edge(int from,int to,int cap,int cost) {
15     G[from].push_back((edge) {
16         to,cap,cost,(int)G[to].size()
17     });
18     G[to].push_back((edge) {
19         from,0,-cost,(int)G[from].size()-1
20     });
21 }
22 int min_cost_flow(int s,int t,int f) {
23     int res=0;
24     fill(h,h+V+1,0);
25     while(f>0) {
26         priority_queue<P,vector<P>,greater<P>>que;
27         fill(dist,dist+V+1,INF);
28         dist[s]=0;
29         que.push(P(0,s));
30         while(!que.empty()) {
31             P p=que.top();
32             que.pop();
33             int v=p.second;
34             if(dist[v]<p.first)continue;
35             for(int i=0; i < G[v].size(); i++) {
36                 edge &e=G[v][i];
37                 if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to]) {
38                     dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
39                     prevv[e.to]=v;
40                     preve[e.to]=i;
41                     que.push(P(dist[e.to],e.to));
42                 }

```

```

43     }
44 }
45 if(dist[t]==INF) {
46     return -1;
47 }
48 for(int v=1; v<=V; v++)h[v]+=dist[v];
49 int d=f;
50 for(int v=t; v!=s; v=prevv[v]) {
51     d=min(d,G[prevv[v]][preve[v]].cap);
52 }
53 f-=d;
54 res+=d*h[t];
55 for(int v=t; v!=s; v=prevv[v]) {
56     edge &e=G[prevv[v]][preve[v]];
57     e.cap-=d;
58     G[v][e.rev].cap+=d;
59 }
60 }
61 return res;
62 }
63 int a[2005];
64 int main() {
65     int T;
66     scanf("%d",&T);
67     while(T--)
68     {
69         int n,k;
70         scanf("%d %d",&n,&k);
71         V=n*2+3;
72         for(int i=1;i<=V;i++)G[i].clear();
73         int s=2*n+1,s0=s+1,t=s0+1;
74         int x,y,z,zz;
75         x=s,y=s0,z=k,zz=0;
76         add_edge(x,y,z,zz);
77         for(int i=1;i<=n;i++)
78         {
79             scanf("%d",&a[i]);
80             x=s0,y=i,z=1,zz=-a[i];
81             add_edge(x,y,z,zz);
82             x=i,y=i+n,z=1,zz=0;
83             add_edge(x,y,z,zz);
84             x=n+i,y=t,z=1,zz=0;
85             add_edge(x,y,z,zz);
86             for(int i0=1;i0<i;i0++)
87             {
88                 if(a[i0]>a[i])continue;
89                 x=i0+n,y=i,z=1,zz=-a[i];
90                 add_edge(x,y,z,zz);
91             }
92         }
93         printf("%d\n",-min_cost_flow(s,t,k));
94     }
95     return 0;

```

96 }

### 3.4 zkw 费用流

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  bool vis[200001];
4  int n,m,s,t,ans,ans0;
5  int nedge=-1,p[200001],c[200001],cc[200001],nex[200001],head[200001],dist[200001];
6  void addedge(int x,int y,int z,int zz){p[++nedge]=y;c[nedge]=z;cc[nedge]=zz;nex[
    nedge]=head[x];head[x]=nedge;}
7  bool spfa(int s,int t)
8  {
9      memset(vis,0,sizeof vis);
10     for(int i=0;i<=n;i++)dist[i]=1e9;
11     dist[t]=0,vis[t]=1;
12     deque<int>q;
13     q.push_back(t);
14     while(!q.empty())
15     {
16         int now=q.front();
17         q.pop_front();
18         for(int k=head[now];k>-1;k=nex[k])if(c[k^1]&&dist[p[k]]>dist[now]-cc[k])
19         {
20             dist[p[k]]=dist[now]-cc[k];
21             if(!vis[p[k]])
22             {
23                 vis[p[k]]=1;
24                 if(!q.empty()&&dist[p[k]]<dist[q.front()])q.push_front(p[k]);
25                 else q.push_back(p[k]);
26             }
27         }
28         vis[now]=0;
29     }
30     return dist[s]<1e9;//不要求流量最大则设置为<0
31 }
32 int dfs(int x,int low)
33 {
34     if(x==t)
35     {
36         vis[t]=1;
37         return low;
38     }
39     int used=0,a;
40     vis[x]=1;
41     for(int k=head[x];k>-1;k=nex[k])
42     {
43         if(!vis[p[k]]&&c[k]&&dist[x]-cc[k]==dist[p[k]])
44         {
45             a=dfs(p[k],min(c[k],low-used));
46             if(a)ans+=a*cc[k],c[k]-=a,c[k^1]+=a,used+=a;
47             if(used==low)break;

```

```

48     }
49 }
50 return used;
51 }
52 int costflow()
53 {
54     int flow=0;
55     while(spfa(s,t))
56     {
57         vis[t]=1;
58         while(vis[t])
59         {
60             memset(vis,0,sizeof vis);
61             flow+=dfs(s,1e9);
62         }
63     }
64     return flow;
65 }
66 int main()
67 {
68     memset(nex,-1,sizeof nex),memset(head,-1,sizeof head);
69     scanf("%d%d%d%d",&n,&m,&s,&t);
70     for(int i=1;i<=m;i++)
71     {
72         int x,y,z,zz;
73         scanf("%d%d%d%d",&x,&y,&z,&zz);
74         addedge(x,y,z,zz),addedge(y,x,0,-zz);
75     }
76     ans=0;
77     ans0=costflow();
78     printf("%d %d\n",ans0,ans);
79     return 0;
80 }

```

### 3.5 KM 算法

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  const int maxn=305;
5  const int inf=1e9;
6
7  int n,minz;
8  int wx[maxn],wy[maxn],match[maxn];
9  int mp[maxn][maxn],slack[maxn],pre[maxn];
10 bool viy[maxn];
11
12 void Bfs(int k)
13 {
14     int py=0,px,yy=0,delta;
15     match[py]=k;
16     for(int i=0;i<n;i++)pre[i]=0,slack[i]=inf;

```

```

17     do
18     {
19         px=match[py],delta=inf,viy[py]=1;
20         for(int i=1; i<=n; ++i)
21         {
22             if(!viy[i])
23             {
24                 if(wx[px]+wy[i]-mp[px][i]<slack[i])slack[i]=wx[px]+wy[i]-mp[px][i],pre[
                    i]=py;
25                 if(slack[i]<delta)delta=slack[i],yy=i;
26             }
27         }
28         for(int i=0; i<=n; ++i)
29         {
30             if(viy[i])wx[match[i]]-=delta,wy[i]+=delta;
31             else slack[i]-=delta;
32         }
33         py=yy;
34     }while(match[py]!=0);
35     while(py)match[py]=match[pre[py]],py=pre[py];
36 }
37
38 int Km()
39 {
40     for(int i=1; i<=n; ++i)
41     {
42         wx[i]=0,wy[i]=0,match[i]=0;
43         for(int j=1; j<=n; ++j)wx[i]=max(wx[i],mp[i][j]);
44     }
45     for(int i=1;i<=n;++i)memset(viy,0,sizeof(viy)),Bfs(i);
46     int Ans=0;
47     for(int i=1;i<=n;++i)Ans+=wx[match[i]]+wy[i];
48     return Ans;
49 }
50
51 int main()
52 {
53     while(~scanf("%d",&n))
54     {
55         for(int i=1;i<=n;++i)for(int j=1;j<=n;++j)scanf("%d",&mp[i][j]);
56         printf("%d\n",Km());
57     }
58     return 0;
59 }

```

### 3.6 带花树

```

1 #include<bits/stdc++.h>
2 #define T 1205
3 using namespace std;
4 int head[T],lst[T*T*2],nxt[T*T];
5 int tot,n,m;

```



```

6 void ins(int x,int y)
7 {
8     lst[++tot]=y;
9     nxt[tot]=head[x];
10    head[x]=tot;
11 }
12 int ma[T],st[T],pr[T],fa[T],q[T],v[T];
13 int ans,TI,u,t;
14 int lca(int x,int y)
15 {
16     for(TI++;;swap(x,y))if(x)
17     {
18         if(v[x]==TI)return x;
19         v[x]=TI;
20         x=fa[pr[ma[x]]];
21     }
22 }
23 void up(int x,int y,int f)
24 {
25     while(fa[x]!=f)
26     {
27         pr[x]=y;
28         if(st[ma[x]]>0)st[q[++t]=ma[x]]=0;
29         if(fa[x]==x)fa[x]=f;
30         if(fa[ma[x]]==ma[x])fa[ma[x]]=f;
31         x=pr[y=ma[x]];
32     }
33 }
34 int match(int x)
35 {
36     for(int i=1;i<=n;i++)fa[i]=i,st[i]=-1;
37     st[q[t=1]=x]=0;
38     for(int l=1;l<=t;l++)for(int i=head[q[l]];i;i=nxt[i])if(st[lst[i]]<0)
39     {
40         st[lst[i]]=1;
41         pr[lst[i]]=q[l];
42         if(!ma[lst[i]])
43         {
44             for(int j=q[l],k=lst[i];j;j=pr[k=u])
45             {
46                 u=ma[j];
47                 ma[j]=k;
48                 ma[k]=j;
49             }
50             return 1;
51         }
52         st[q[++t]=ma[lst[i]]]=0;
53     }
54     else if(fa[lst[i]]!=fa[q[l]]&&!st[lst[i]])
55     {
56         int f=lca(lst[i],q[l]);
57         up(q[l],lst[i],f);
58         up(lst[i],q[l],f);

```

```

59     for(int j=1;j<=n;j++)fa[j]=fa[fa[j]];
60 }
61 return 0;
62 }
63 int main()
64 {
65     int cas;
66     scanf("%d",&cas);
67     while(cas--)
68     {
69         TI=tot=0;
70         memset(head,0,sizeof(head));
71         memset(ma,0,sizeof(ma));
72         memset(v,0,sizeof(v));
73         scanf("%d %d",&n,&m);
74         ans=-n;
75         for(int i=1;i<=n;i++)
76         {
77             ins(i,i+n),ins(i+n,i);
78             int num,tmp;
79             scanf("%d",&num);
80             while(num--)
81             {
82                 scanf("%d",&tmp);
83                 tmp+=2*n;
84                 ins(tmp,i),ins(i,tmp);
85                 ins(tmp,i+n),ins(i+n,tmp);
86             }
87         }
88         n=2*n+m;
89         for(int i=1;i<=n;i++)ans+=!ma[i]&&match(i);
90         printf("%d\n",ans);
91     }
92     return 0;
93 }

```

### 3.7 并查集

```

1  int fin(int x)
2  {
3      int tmp=x,tmp2;
4      while(pre[tmp]!=tmp)tmp=pre[tmp];
5      while(x!=tmp)tmp2=pre[x],pre[x]=tmp,x=tmp2;
6      return tmp;
7  }
8
9  int fin(int x){return (pre[x]==x)?x:pre[x]=fin(pre[x]);}
10 void uni(int x,int y){if(fin(x)!=fin(y))pre[fin(y)]=fin(x);}
11
12 struct DisjointSetUnion
13 {
14     int fa[maxn*2],rank[maxn*2];

```

```

15     stack<pair<int*,int>>stk;
16     void init(){for(int i=1;i<=maxn;i++)fa[i]=i,rank[i]=0;}
17     int find(int x){return (x^fa[x])?find(fa[x]):x;}
18     void join(int x,int y)
19     {
20         x=find(x),y=find(y);
21         if(rank[x]<=rank[y])
22         {
23             stk.push({fa+x,fa[x]}),fa[x]=y;
24             if(rank[x]==rank[y])stk.push({rank+y,rank[y]}),rank[y]++;
25         }
26         else stk.push({fa+y,fa[y]}),fa[y]=x;
27     }
28     void undo(){*stk.top().first=stk.top().second,stk.pop();}
29 }DSU;

```

### 3.8 割点

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=100005;
4  using namespace std;
5  int n,m,len=0,id=0,ans=0;
6  int last[maxn],low[maxn],dfn[maxn];
7  bool bz[maxn];
8  vector<int>mp[maxn];
9  void dfs(int x,int root)
10 {
11     int tot=0;
12     low[x]=dfn[x]=++id;
13     for(int y:mp[x])
14     {
15         if(!dfn[y])
16         {
17             dfs(y,root);
18             low[x]=min(low[x],low[y]);
19             if(low[y]>=dfn[x]&&x!=root) bz[x]=true;
20             if(x==root) tot++;
21         }
22         low[x]=min(low[x],dfn[y]);
23     }
24     if(x==root&&tot>=2) bz[root]=true;
25 }
26 int main()
27 {
28     int x,y,t;
29     scanf("%d %d",&n,&m);
30     for(int i=1;i<=m;i++)
31     {
32         scanf("%d %d",&x,&y);
33         mp[x].push_back(y);
34         mp[y].push_back(x);

```

```

35     }
36     for(int i=1;i<=n;i++)if(!dfn[i])dfs(i,i);
37     for(int i=1;i<=n;i++)if(bz[i])printf("%d\n",i);
38 }

```

### 3.9 桥

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=5005;
4  int i,i0,n,m,T,dfn[maxn],low[maxn],deep,cnt[maxn],sum,pre[maxn];
5  vector<int>mp[maxn];
6  int fin(int x){return (pre[x]==x)?x:pre[x]=fin(pre[x]);}
7  void uni(int x,int y){if(fin(x)!=fin(y))pre[fin(y)]=fin(x);}
8  void dfs(int now,int root)
9  {
10     dfn[now]=low[now]=++deep;
11     bool f=0;
12     for(int i:mp[now])
13     {
14         if(i==root&&!f)f=1;
15         else
16         {
17             if(!dfn[i])dfs(i,now);
18             low[now]=min(low[now],low[i]);
19             if(low[now]==low[i])uni(now,i);
20         }
21     }
22 }
23 int main()
24 {
25     while(scanf("%d %d",&n,&m)!=EOF)
26     {
27         sum=deep=0;
28         for(i=1;i<=n;i++)mp[i].clear(),dfn[i]=0,cnt[i]=0,pre[i]=i;
29         while(m--)
30         {
31             int a,b;
32             scanf("%d %d",&a,&b);
33             mp[a].push_back(b),mp[b].push_back(a);
34         }
35         for(i=1;i<=n;i++)if(!dfn[i])dfs(i,-1);
36         for(i=1;i<=n;i++)
37         {
38             for(int i0:mp[i])
39             {
40                 if(fin(i)!=fin(i0))printf("%d %d\n",fin(i),fin(i0));
41             }
42         }
43     }
44     return 0;
45 }

```

### 3.10 强连通分量

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=1005;
4  int i,i0,n,m,T,dfn[maxn],low[maxn],deep,col[maxn],sum,cnt[maxn],out[maxn];
5  bool vis[maxn];
6  vector<int>mp[maxn];
7  stack<int>stk;
8  void dfs(int now)
9  {
10     dfn[now]=low[now]=++deep;
11     stk.push(now);
12     vis[now]=1;
13     for(auto i:mp[now])
14     {
15         if(!dfn[i])
16         {
17             dfs(i);
18             low[now]=min(low[now],low[i]);
19         }
20         else if(vis[i])low[now]=min(low[now],low[i]);
21     }
22     if(dfn[now]==low[now])
23     {
24         sum++;
25         while(vis[stk.top()])
26         {
27             col[stk.top()]=sum;
28             vis[stk.top()]=0;
29             cnt[sum]++;
30             stk.pop();
31         }
32     }
33 }
34 int main()
35 {
36     while(scanf("%d %d",&n,&m)!=EOF)
37     {
38         sum=deep=0;
39         for(i=1;i<=n;i++)mp[i].clear(),dfn[i]=0,cnt[i]=0,out[i]=0;
40         while(m--)
41         {
42             int a,b;
43             scanf("%d %d",&a,&b);
44             mp[a].push_back(b);
45         }
46         for(i=1;i<=n;i++)if(!dfn[i])dfs(i);
47         int ans=0;
48         for(i=1;i<=n;i++)for(auto i0:mp[i])out[col[i]]+=(col[i]!=col[i0]);
49         for(i=1;i<=sum;i++)if(!out[i]&&cnt[i]>1)ans++;
50         printf("%d\n",ans);
51     }

```

```

52     return 0;
53 }

```

### 3.11 树链剖分

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define mid (l+r)/2
4  const int maxn=100005;
5  int i,i0,n,m,dep[maxn],siz[maxn],fa[maxn],son[maxn],idx[maxn],cnt,top[maxn];
6  vector<int>mp[maxn];
7  void dfs(int now,int root,int deep)
8  {
9      dep[now]=deep,fa[now]=root,siz[now]=1,son[now]=-1;
10     for(auto i:mp[now])
11     {
12         if(dep[i])continue;
13         dfs(i,now,deep+1);
14         siz[now]+=siz[i];
15         if(son[now]==-1||siz[son[now]]<siz[i])son[now]=i;
16     }
17 }
18 void dfs0(int now,int topf)
19 {
20     idx[now]=++cnt,top[now]=topf;
21     if(son[now]!=-1)dfs0(son[now],topf);
22     for(auto i:mp[now])if(!idx[i])dfs0(i,i);
23 }
24 int tree[maxn*4],lazy[maxn*4];
25 void p_tree(int l,int r,int p)
26 {
27     if(lazy[p]==-1)return;
28     lazy[p*2]=lazy[p],lazy[p*2+1]=lazy[p];
29     tree[p*2]=(mid-l+1)*lazy[p],tree[p*2+1]=(r-mid)*lazy[p];
30     lazy[p]=-1;
31 }
32 void c_tree(int l,int r,int p,int a,int b,int v)
33 {
34     if(l==a&&r==b)lazy[p]=v,tree[p]=(r-l+1)*v;
35     else
36     {
37         p_tree(l,r,p);
38         if(b<=mid) c_tree(l,mid,p*2,a,b,v);
39         else if(a>=mid+1)c_tree(mid+1,r,p*2+1,a,b,v);
40         else c_tree(l,mid,p*2,a,mid,v),c_tree(mid+1,r,p*2+1,mid+1,b,v);
41         tree[p]=tree[p*2]+tree[p*2+1];
42     }
43 }
44 int q_tree(int l,int r,int p,int a,int b)
45 {
46     if(l==a&&r==b)return tree[p];
47     else

```

```

48     {
49         p_tree(1,r,p);
50         if(b<=mid)return q_tree(1,mid,p*2,a,b);
51         else if(a>=mid+1)return q_tree(mid+1,r,p*2+1,a,b);
52         else return q_tree(1,mid,p*2,a,mid)+q_tree(mid+1,r,p*2+1,mid+1,b);
53     }
54 }
55 int q_range(int x,int y)
56 {
57     int ans=0;
58     while(top[x]!=top[y])
59     {
60         if(dep[top[x]]<dep[top[y]])swap(x,y);
61         ans+=q_tree(1,n,1,idx[top[x]],idx[x]);
62         x=fa[top[x]];
63     }
64     if(dep[x]>dep[y])swap(x,y);
65     ans+=q_tree(1,n,1,idx[x],idx[y]);
66     return ans;
67 }
68 void c_range(int x,int y,int k)
69 {
70     while(top[x]!=top[y])
71     {
72         if(dep[top[x]]<dep[top[y]])swap(x,y);
73         c_tree(1,n,1,idx[top[x]],idx[x],k);
74         x=fa[top[x]];
75     }
76     if(dep[x]>dep[y])swap(x,y);
77     c_tree(1,n,1,idx[x],idx[y],k);
78 }
79 int q_son(int x){return q_tree(1,n,1,idx[x],idx[x]+siz[x]-1);}
80 void c_son(int x,int k){c_tree(1,n,1,idx[x],idx[x]+siz[x]-1,k);}
81 void init()
82 {
83     memset(dep,0,sizeof(dep));
84     memset(idx,0,sizeof(idx));
85     memset(lazy,-1,sizeof(lazy));
86     cnt=0;
87 }
88 int main()
89 {
90     int r=1;
91     scanf("%d",&n);
92     for(i=2;i<=n;i++)
93     {
94         int x;
95         scanf("%d",&x),x++;
96         mp[i].push_back(x);
97         mp[x].push_back(i);
98     }
99     init();
100    dfs(r,-1,1);

```

```

101     dfs0(r,r);
102     scanf("%d",&m);
103     int now=0;
104     while(m--)
105     {
106         char op[10];
107         int x;
108         scanf("%s %d",op,&x),x++;
109         if(op[0]=='i')c_range(x,r,1);
110         if(op[0]=='u')c_son(x,0);
111         int tmp=q_son(r);
112         printf("%d\n",abs(now-tmp));
113         now=tmp;
114     }
115     return 0;
116 }

```

### 3.12 LinkCutTree

```

1  #include<cstdio>
2  #include<iostream>
3  #include<algorithm>
4  #define inf 1000000000
5  #define ll long long
6  using namespace std;
7  int read()
8  {
9      int x=0,f=1;char ch=getchar();
10     while(ch<'0' || ch>'9'){if(ch=='-')f=-1;ch=getchar();}
11     while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
12     return x*f;
13 }
14 int top,q[200005];
15 int n,m,ans=0;
16 int mx[200005],val[200005];
17 int p[200005],fa[200005],c[200005][2];
18 bool rev[200005];
19 struct Data{
20     int u,v,a,b;
21 }e[100005];
22 int find(int x)
23 {
24     return x==p[x]?x:p[x]=find(p[x]);
25 }
26 bool operator<(Data a,Data b)
27 {
28     return a.a<b.a;
29 }
30 bool isroot(int x)
31 {
32     return c[fa[x]][0]!=x&&c[fa[x]][1]!=x;
33 }

```



```

34 void update(int x)
35 {
36     int l=c[x][0],r=c[x][1];
37     mx[x]=x;
38     if(val[mx[l]]<val[mx[x]])mx[x]=mx[l];
39     if(val[mx[r]]<val[mx[x]])mx[x]=mx[r];
40 }
41 void pushdown(int x)
42 {
43     int l=c[x][0],r=c[x][1];
44     if(rev[x])
45     {
46         rev[x]^=1;rev[l]^=1;rev[r]^=1;
47         swap(c[x][0],c[x][1]);
48     }
49 }
50 void rotate(int &x)
51 {
52     int y=fa[x],z=fa[y],l,r;
53     if(c[y][0]==x)l=0;else l=1;r=l^1;
54     if(!isroot(y))
55     {
56         if(c[z][0]==y)c[z][0]=x;
57         else c[z][1]=x;
58     }
59     fa[x]=z;fa[y]=x;fa[c[x][r]]=y;
60     c[y][l]=c[x][r];c[x][r]=y;
61     update(y);update(x);
62 }
63 void splay(int &x)
64 {
65     top=0;q[++top]=x;
66     for(int i=x;!isroot(i);i=fa[i])q[++top]=fa[i];
67     while(top)pushdown(q[top--]);
68     while(!isroot(x))
69     {
70         int y=fa[x],z=fa[y];
71         if(!isroot(y))
72         {
73             if(c[y][0]==x^c[z][0]==y)rotate(x);
74             else rotate(y);
75         }
76         rotate(x);
77     }
78 }
79 void access(int x)
80 {
81     for(int t=0;x;t=x,x=fa[x])
82         splay(x),c[x][1]=t,update(x);
83 }
84 void makeroot(int x)
85 {
86     access(x);splay(x);rev[x]^=1;

```

```

87 }
88 void link(int x,int y)
89 {
90     makeroot(x);fa[x]=y;
91 }
92 void cut(int x,int y)
93 {
94     makeroot(x);access(y);splay(y);
95     c[y][0]=fa[x]=0;update(y);
96 }
97 int query(int x,int y)
98 {
99     makeroot(x);access(y);splay(y);
100     return mx[y];
101 }
102 int main()
103 {
104     n=read();m=read();
105     for(int i=1;i<=n;i++)p[i]=i,val[i]=inf;
106     for(int i=1;i<=m;i++)
107     {
108         e[i].u=read();e[i].v=read();e[i].a=read();e[i].b=read();
109     }
110     sort(e+1,e+m+1);
111     int tot=0;
112     val[0]=inf;
113     for(int i=1;i<=m;i++)
114     {
115         int u=e[i].u,v=e[i].v,a=e[i].a,b=e[i].b;
116         if(find(u)==find(v))
117         {
118             int t=query(u,v);
119             if(val[t]<e[i].b)
120             {
121                 cut(t,e[t-n].u);
122                 cut(t,e[t-n].v);
123             }
124             else
125             {
126                 if(find(1)==find(n))
127                 {
128                     //printf("%d %d %d\n",i,e[i].a,val[query(1,n)]);
129                     if(i==m)ans+=max(val[query(1,n)]-e[i].a+1,0);
130                     else ans+=max(min(val[query(1,n)],e[i+1].a-1)-e[i].a+1,0);
131                 }
132                 continue;
133             }
134         }
135         else p[find(u)]=find(v);
136         val[n+i]=e[i].b;mx[n+i]=n+i;
137         link(u,n+i);link(v,n+i);
138         if(find(1)==find(n))
139         {

```

```

140         int t=query(1,n);
141         //printf("%d %d %d\n",i,e[i].a,val[t]);
142         if(i==m)ans+=max(val[query(1,n)]-e[i].a+1,0);
143         else ans+=max(min(val[query(1,n)],e[i+1].a-1)-e[i].a+1,0);
144     }
145 }
146 printf("%d\n",ans);
147 return 0;
148 }

```

### 3.13 任意根 lca

```

1 int r,x,y;
2 int rx=lca(r,x),ry=lca(r,y);
3 if(rx==ry)printf("%d\n",lca(x,y));
4 else printf("%d\n",d(rx,r)<d(ry,r)?rx:ry);

```

### 3.14 最大独立集

```

1 #include<stdio>
2 #include<cstring>
3 #define N 50
4 /*
5 最大团 = 补图G的最大独立集数
6 ——>最大独立集数 = 补图G'最大团
7 */
8 //最大团模板
9 bool a[N][N]; //a为图的邻接表(从1开始)
10 int ans, cnt[N], group[N], n, m, vis[N]; //ans表示最大团, cnt[N]表示当前最大团的节点数,
    group[N]用以寻找一个最大团集合
11 bool dfs( int u, int pos ) //u为当前从顶点开始深搜, pos为深搜深度 (即当前深搜树所在第几层
    的位置)
12 {
13     int i, j;
14     for( i = u+1; i <= n; i++) //按递增顺序枚举顶点
15     {
16         if( cnt[i]+pos <= ans ) return 0; //剪枝
17         if( a[u][i] )
18         {
19             // 与目前团中元素比较, 取 Non-N(i)
20             for( j = 0; j < pos; j++ ) if( !a[i][ vis[j] ] ) break;
21             if( j == pos )
22             { // 若为空, 则皆与 i 相邻, 则此时将i加入到 最大团中
23                 vis[pos] = i; //深搜层次也就是最大团的顶点数目, vis[pos] = i表示当前第pos小的
                    最大团元素为i (因为按递增顺序枚举顶点)
24                 if( dfs( i, pos+1 ) ) return 1;
25             }
26         }
27     }
28     if( pos > ans )
29     {

```

```

30         for( i = 0; i < pos; i++ )
31             group[i] = vis[i]; // 更新最大团元素
32         ans = pos;
33         return 1;
34     }
35     return 0;
36 }
37 void maxclique()//求最大团
38 {
39     ans=-1;
40     for(int i=n;i>0;i--)
41     {
42         vis[0]=i;
43         dfs(i,1);
44         cnt[i]=ans;
45     }
46 }
47 int main()
48 {
49     int T;
50     //scanf("%d",&T);
51     while(~scanf("%d",&n))
52     {
53         if(n==0) break;
54         //scanf("%d%d",&n,&m );
55         int x, y;
56         memset( a, 0, sizeof(a));
57         /*for(int i = 0; i < m; i++)
58         {
59             scanf("%d%d",&x,&y);
60             a[x][y] = a[y][x] = 1;
61         }*/
62         //相邻顶点间有边相连，模型转换成求 无向图 最大独立集。
63         //要求原图的最大独立集，转化为求原图的补图的最大团(最大团顶点数量 = 补图的最大独立集)
64         for(int i = 1; i <= n; i++)//求原图的补图
65             for(int j = 1; j <= n; j++)
66                 scanf("%d",&a[i][j]);
67         maxclique();//求最大团
68         if( ans < 0 ) ans = 0;//ans表示最大团
69         printf("%d\n", ans );
70         /*for(int i = 0; i < ans; i++)
71             printf( i == 0 ? "%d" : " %d", group[i]);*/group[N]用以寻找一个最大团集合
72         if( ans > 0 ) puts("");*/
73     }
74 }

```

### 3.15 拓扑排序

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int i,i0,n,m,cnt[505];

```

```

4 vector<int>mp[505],ans;
5 priority_queue<int,vector<int>,greater<int>>q;
6 int main()
7 {
8     while(scanf("%d %d",&n,&m)!=EOF)
9     {
10         ans.clear();
11         for(i=1;i<=n;i++)mp[i].clear();
12         while(m--)
13         {
14             int a,b;
15             scanf("%d %d",&a,&b);
16             mp[a].push_back(b);
17             cnt[b]++;
18         }
19         for(i=1;i<=n;i++)if(!cnt[i])q.push(i);
20         while(!q.empty())
21         {
22             m=q.top(),q.pop();
23             ans.push_back(m);
24             for(auto i:mp[m])if(--cnt[i])q.push(i);
25         }
26         for(i=0;i<ans.size();i++)printf("%d%c",ans[i],i==ans.size()-1?'\\n':' ');
27     }
28     return 0;
29 }

```

### 3.16 树分治

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int i,i0,n,k,msiz[100005],siz[100005],sum[100005],ans[100005];
4 bool vis[100005];
5 vector<int>mp[100005];
6 int getRoot(int now,int root,int Siz)
7 {
8     int res=-1;
9     msiz[now]=0,siz[now]=1;
10    for(int i:mp[now])
11    {
12        if(vis[i]||i==root)continue;
13        int x=getRoot(i,now,Siz);
14        msiz[now]=max(msiz[now],siz[i]),siz[now]+=siz[i];
15        if(res==-1||msiz[x]<msiz[res])res=x;
16    }
17    msiz[now]=max(msiz[now],Siz-siz[now]);
18    if(res==-1||msiz[now]<msiz[res])res=now;
19    return res;
20 }
21 void add(int p, int x){while(p<=n)sum[p]+=x,p+=p&-p;}
22 int ask(int p){int res=0;while(p)res+=sum[p],p-=p&-p;return res;}
23 int range_ask(int l, int r){return ask(r)-ask(l-1);}

```

```

24 void cal(int now,int root,int dep,int sign)
25 {
26     add(dep+1,sign);
27     if(k-dep>0)for(int i:mp[now])if(!vis[i]&&root!=i)cal(i,now,dep+1,sign);
28 }
29 void dfs(int now,int root,int dep)
30 {
31     if(k-dep>=0)ans[now]+=range_ask(1,k-dep+1);
32     if(k-dep>0)for(int i:mp[now])if(!vis[i]&&root!=i)dfs(i,now,dep+1);
33 }
34 int calSiz(int now,int root)
35 {
36     int Siz=1;
37     for(int i:mp[now])if(!vis[i]&&root!=i)Siz+=calSiz(i,now);
38     return Siz;
39 }
40 void div(int now)
41 {
42     now=getRoot(now,-1,calSiz(now,-1)),vis[now]=1;
43     add(1,1);
44     for(int i:mp[now])if(!vis[i])cal(i,now,1,1);
45     ans[now]+=range_ask(1,k+1);
46     for(int i:mp[now])
47     {
48         if(!vis[i])
49         {
50             cal(i,now,1,-1);
51             dfs(i,now,1);
52             cal(i,now,1,1);
53         }
54     }
55     for(int i:mp[now])if(!vis[i])cal(i,now,1,-1);
56     add(1,-1);
57     for(int i:mp[now])if(!vis[i])div(i);
58 }
59 int main()
60 {
61     scanf("%d %d",&n,&k);
62     for(i=1;i<n;i++)
63     {
64         int x,y;
65         scanf("%d %d",&x,&y);
66         mp[x].push_back(y),mp[y].push_back(x);
67     }
68     div(1);
69     for(i=1;i<=n;i++)printf("%d%c",ans[i],i==n?'\\n':' ');
70     return 0;
71 }

```

### 3.17 DsuOnTree

```

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

```

```

2  using namespace std;
3  const int MAXN=1e5+5;
4  int n,col[MAXN],a[MAXN],sz[MAXN],son[MAXN],cnt[MAXN],bigson,max1;
5  long long ans[MAXN],sum;
6  vector<int>G[MAXN];
7  void init() {
8      for(int i=1; i<=n; i++)G[i].clear(),cnt[i]=0,max1=0;
9  }
10 void initdfs(int now,int fa) {
11     sz[now]=1;
12     for(auto to:G[now]) {
13         if(to==fa)continue;
14         initdfs(to,now);
15         sz[now]+=sz[to];
16         if(sz[to]>sz[son[now]])son[now]=to;//big son or not
17     }
18 }
19 void add(int now,int fa,int val) {
20     //operation
21     cnt[col[now]]+=val;
22     if(cnt[col[now]]>max1)max1=cnt[col[now]],sum=col[now];
23     else if(cnt[col[now]]==max1)sum+=col[now];
24
25
26     for(auto &to:G[now]) {
27         if(to==fa||to==bigson)continue;
28         add(to,now,val);
29     }
30 }
31 void dfs(int now,int fa,int op) {
32     for(auto &to:G[now]) {
33         if(to==fa)continue;
34         if(to!=son[now])dfs(to,now,0);//if(not big son) clear the influence
35     }
36     if(son[now]) { //as to big son
37         dfs(son[now],now,1);//t clear influence
38         bigson=son[now];//get big son position
39     }
40     add(now,fa,1),bigson=0;//count all small son ans
41     ans[now]=sum;//update
42     if(op==0) { //delete influence from small son
43         add(now,fa,-1);
44         sum=0;
45         max1=0;
46     }
47 }
48 int main() {
49     cin>>n;
50     for(int i=1; i<=n; i++)cin>>col[i];
51     for(int i=1; i<n; i++) {
52         int x,y;
53         cin>>x>>y;
54         G[x].push_back(y);

```

```

55     G[y].push_back(x);
56 }
57 initdfs(1,-1);
58 dfs(1,-1,0);
59 for(int i=1; i<=n; i++)printf("%lld%c",ans[i],i==n?'\n':' ');
60 return 0;
61 }

```

### 3.18 虚树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define mid (l+r)/2
4  const int maxn=100005;
5  int i,i0,n,m,dep[maxn],siz[maxn],fa[maxn],son[maxn],idx[maxn],cnt,top[maxn];
6  int dp[maxn][2];
7  vector<int>mp[maxn],mp0[maxn];
8  bool v[maxn];
9  void dfs(int now,int root,int deep)
10 {
11     dep[now]=deep,fa[now]=root,siz[now]=1,son[now]=-1;
12     for(auto i:mp[now])
13     {
14         if(i==root)continue;
15         dfs(i,now,deep+1);
16         siz[now]+=siz[i];
17         if(son[now]==-1||siz[son[now]]<siz[i])son[now]=i;
18     }
19 }
20 void dfs0(int now,int topf)
21 {
22     idx[now]=++cnt,top[now]=topf;
23     if(son[now]!=-1)dfs0(son[now],topf);
24     for(auto i:mp[now])
25     {
26         if(i==fa[now]||i==son[now])continue;
27         dfs0(i,i);
28     }
29 }
30 int lca(int x,int y)
31 {
32     while(top[x]!=top[y])
33     {
34         if(dep[top[x]]<dep[top[y]])swap(x,y);
35         x=fa[top[x]];
36     }
37     if(dep[x]>dep[y])swap(x,y);
38     return x;
39 }
40 vector<int>q;
41 bool cmp(int x,int y){return idx[x]<idx[y];}
42 stack<int>stk;

```



```
43 void link(int x,int y){mp[x].push_back(y),mp[y].push_back(x);}
44 void link0(int x,int y){mp0[x].push_back(y),mp0[y].push_back(x);}
45 void dfs(int now,int root)
46 {
47     dp[now][0]=dp[now][1]=0;
48     for(int i:mp0[now])
49     {
50         if(i!=root)
51         {
52             dfs(i,now);
53             dp[now][1]+=dp[i][1],dp[now][0]+=dp[i][0];
54             if(v[now])
55             {
56                 if(v[i]&&fa[i]==now)dp[now][0]+=1000000;
57                 if(dp[i][1])dp[now][0]++;
58             }
59             v[i]=0;
60         }
61     }
62     if(v[now])dp[now][1]=1;
63     else
64     {
65         if(dp[now][1]>1)dp[now][1]=0,dp[now][0]++;
66     }
67     mp0[now].clear();
68 }
69 int main()
70 {
71     int r=1;
72     scanf("%d",&n);
73     for(i=1;i<n;i++)
74     {
75         int x,y;
76         scanf("%d %d",&x,&y);
77         link(x,y);
78     }
79     dfs(r,-1,1),cnt=0,dfs0(r,r);
80     scanf("%d",&m);
81     while(m--)
82     {
83         int k,x;
84         scanf("%d",&k);
85         q.clear(),v[r]=0;
86         while(k--)
87         {
88             scanf("%d",&x);
89             if(x!=r)q.push_back(x);
90             v[x]=1;
91         }
92         sort(q.begin(),q.end(),cmp);
93         stk.push(r);
94         for(auto i:q)
95         {
```

```

96         if(stk.empty())
97         {
98             stk.push(i);
99             continue;
100         }
101         int p=lca(i,stk.top()),tmp=stk.top();
102         stk.pop();
103         while(!stk.empty()&&dep[stk.top()]>=dep[p])
104         {
105             link0(tmp,stk.top());
106             tmp=stk.top(),stk.pop();
107         }
108         if(p!=tmp)link0(p,tmp),stk.push(p);
109         else stk.push(tmp);
110         stk.push(i);
111     }
112     x=stk.top(),stk.pop();
113     while(!stk.empty())link0(x,stk.top()),x=stk.top(),stk.pop();
114     dfs(r,-1);
115     if(dp[r][0]>=1000000)dp[r][0]=-1;
116     printf("%d\n",dp[r][0]);
117 }
118 return 0;
119 }

```

## 4 数据结构

### 4.1 莫队分块

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int i,n,m,T,a[50005],ans0[200005],bsize,cnt[1000005],p,p0,p1,pr,tim,ans,timp;
4  #define bel(x) ((x - 1) / bsize + 1)
5  struct node{int num,pos,i;}timer[50005];
6  struct query
7  {
8      int l,r,i,p;
9      bool operator < (const query &b) const
10     {
11         if(bel(l) != bel(b.l)) return l < b.l;
12         if(bel(r) != bel(b.r)) return r < b.r;
13         return i<b.i;
14     }
15 }q[200005];
16 void change_time(int ntim)
17 {
18     if(tim>ntim)
19     {
20         timp--;
21         while(timp>=0&&timer[timp].i>ntim)
22         {
23             if(timer[timp].pos<=pr&&timer[timp].pos>=p1)

```

```

24     {
25         cnt[a[timer[timp].pos]]--;
26         if(!cnt[a[timer[timp].pos]])ans--;
27         swap(a[timer[timp].pos],timer[timp].num);
28         if(!cnt[a[timer[timp].pos]])ans++;
29         cnt[a[timer[timp].pos]]++;
30     }
31     else swap(a[timer[timp].pos],timer[timp].num);
32     timp--;
33 }
34 timp++;
35 }
36 if(tim<ntim)
37 {
38     while(timp<p0&&timer[timp].i<ntim)
39     {
40         if(timer[timp].pos<=pr&&timer[timp].pos>=pl)
41         {
42             cnt[a[timer[timp].pos]]--;
43             if(!cnt[a[timer[timp].pos]])ans--;
44             swap(a[timer[timp].pos],timer[timp].num);
45             if(!cnt[a[timer[timp].pos]])ans++;
46             cnt[a[timer[timp].pos]]++;
47         }
48         else swap(a[timer[timp].pos],timer[timp].num);
49         timp++;
50     }
51 }
52 tim=ntim;
53 }
54 int main()
55 {
56
57     while(scanf("%d",&n)!=EOF)
58     {
59         memset(cnt,0,sizeof(cnt));
60         bsize=max(1,(int)pow(n,2.0/3));
61         for(i=1;i<=n;i++)scanf("%d",&a[i]);
62         scanf("%d",&m);
63         for(i=1,p0=p=0;i<=m;i++)
64         {
65             int op;
66             scanf("%d",&op);
67             if(op==1)
68             {
69                 scanf("%d %d",&q[p].l,&q[p].r);
70                 q[p].i=i;
71                 q[p].p=p;
72                 p++;
73             }
74             if(op==2)
75             {
76                 scanf("%d %d",&timer[p0].pos,&timer[p0].num);

```

```

77         timer[p0].i=i;
78         p0++;
79     }
80 }
81 sort(q,q+p);
82 pl=1,pr=1,ans=1,tim=1,timp=0;
83 cnt[a[1]]++;
84 for(i=0;i<p;i++)
85 {
86     change_time(q[i].i);
87     while(pl<q[i].l)
88     {
89         cnt[a[pl]]--;
90         if(!cnt[a[pl]])ans--;
91         pl++;
92     }
93     while(pl>q[i].l)
94     {
95         pl--;
96         if(!cnt[a[pl]])ans++;
97         cnt[a[pl]]++;
98     }
99     while(pr<q[i].r)
100    {
101        pr++;
102        if(!cnt[a[pr]])ans++;
103        cnt[a[pr]]++;
104    }
105    while(pr>q[i].r)
106    {
107        cnt[a[pr]]--;
108        if(!cnt[a[pr]])ans--;
109        pr--;
110    }
111    ans0[q[i].p]=ans;
112 }
113 for(i=0;i<p;i++)printf("%d\n",ans0[i]);
114 }
115 return 0;
116 }

```

## 4.2 带修改莫队分块

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int i,n,m,T,a[50005],ans0[200005],bsize,cnt[1000005],p,p0,pl,pr,tim,ans,timp;
4 #define bel(x) ((x - 1) / bsize + 1)
5 struct node{int num,pos,i;}timer[50005];
6 struct query
7 {
8     int l,r,i,p;
9     bool operator < (const query &b) const

```

```

10     {
11         if(bel(l) != bel(b.l)) return l < b.l;
12         if(bel(r) != bel(b.r)) return r < b.r;
13         return i < b.i;
14     }
15 }q[200005];
16 void change_time(int ntim)
17 {
18     if(tim>ntim)
19     {
20         timp--;
21         while(timp>=0&&timer[timp].i>ntim)
22         {
23             if(timer[timp].pos<=pr&&timer[timp].pos>=pl)
24             {
25                 cnt[a[timer[timp].pos]]--;
26                 if(!cnt[a[timer[timp].pos]])ans--;
27                 swap(a[timer[timp].pos],timer[timp].num);
28                 if(!cnt[a[timer[timp].pos]])ans++;
29                 cnt[a[timer[timp].pos]]++;
30             }
31             else swap(a[timer[timp].pos],timer[timp].num);
32             timp--;
33         }
34         timp++;
35     }
36     if(tim<ntim)
37     {
38         while(timp<p0&&timer[timp].i<ntim)
39         {
40             if(timer[timp].pos<=pr&&timer[timp].pos>=pl)
41             {
42                 cnt[a[timer[timp].pos]]--;
43                 if(!cnt[a[timer[timp].pos]])ans--;
44                 swap(a[timer[timp].pos],timer[timp].num);
45                 if(!cnt[a[timer[timp].pos]])ans++;
46                 cnt[a[timer[timp].pos]]++;
47             }
48             else swap(a[timer[timp].pos],timer[timp].num);
49             timp++;
50         }
51     }
52     tim=ntim;
53 }
54 int main()
55 {
56
57     while(scanf("%d",&n)!=EOF)
58     {
59         memset(cnt,0,sizeof(cnt));
60         bsize=max(1,(int)pow(n,2.0/3));
61         for(i=1;i<=n;i++)scanf("%d",&a[i]);
62         scanf("%d",&m);

```

```

63     for(i=1,p0=p=0;i<=m;i++)
64     {
65         int op;
66         scanf("%d",&op);
67         if(op==1)
68         {
69             scanf("%d %d",&q[p].l,&q[p].r);
70             q[p].i=i;
71             q[p].p=p;
72             p++;
73         }
74         if(op==2)
75         {
76             scanf("%d %d",&timer[p0].pos,&timer[p0].num);
77             timer[p0].i=i;
78             p0++;
79         }
80     }
81     sort(q,q+p);
82     pl=1,pr=1,ans=1,tim=1,timp=0;
83     cnt[a[1]]++;
84     for(i=0;i<p;i++)
85     {
86         change_time(q[i].i);
87         while(pl<q[i].l)
88         {
89             cnt[a[pl]]--;
90             if(!cnt[a[pl]])ans--;
91             pl++;
92         }
93         while(pl>q[i].l)
94         {
95             pl--;
96             if(!cnt[a[pl]])ans++;
97             cnt[a[pl]]++;
98         }
99         while(pr<q[i].r)
100        {
101            pr++;
102            if(!cnt[a[pr]])ans++;
103            cnt[a[pr]]++;
104        }
105        while(pr>q[i].r)
106        {
107            cnt[a[pr]]--;
108            if(!cnt[a[pr]])ans--;
109            pr--;
110        }
111        ans0[q[i].p]=ans;
112    }
113    for(i=0;i<p;i++)printf("%d\n",ans0[i]);
114 }
115 return 0;

```

116 }

### 4.3 树状数组

```

1 //仅可用区间修改区间查询
2 int sum1[n],sum2[n],n;
3 void add(int p,int x){for(int i=p;i<=n;i+=i&-i)sum1[i]+=x,sum2[i]+=x*p;}
4 int ask(int p)
5 {
6     int res=0;
7     for(int i=p;i-=i&-i)res+=(p+1)*sum1[i]-sum2[i];
8     return res;
9 }
10 void range_add(int l,int r,int x){add(l,x),add(r+1,-x);}
11 int range_ask(int l,int r){return ask(r)-ask(l-1);}
12
13 //单点修改 区间查询
14 int sum[n],n;
15 void add(int p, int x){while(p<=n)sum[p]+=x,p+=p&-p;}
16 int ask(int p)
17 {
18     int res=0;
19     while(p)res+=sum[p],p-=p&-p;
20     return res;
21 }
22 int range_ask(int l, int r){return ask(r)-ask(l-1);}
23
24 void add(int x, int y, int z){
25     int memo_y = y;
26     while(x <= n){
27         y = memo_y;
28         while(y <= m)
29             tree[x][y] += z,tree[x][y]%=mod, y += y & -y;
30         x += x & -x;
31     }
32 }
33 //区间修改 单点查询
34 void range_add(int xa, int ya, int xb, int yb, int z){//左上点和右下点
35     add(xa, ya, z);
36     add(xa, yb + 1, -z);
37     add(xb + 1, ya, -z);
38     add(xb + 1, yb + 1, z);
39 }
40 long long ask(int x, int y){
41     int res = 0, memo_y = y;
42     while(x){
43         y = memo_y;
44         while(y)
45             res += tree[x][y],res%=mod, y -= y & -y;
46         x -= x & -x;
47     }
48     res+=mod,res%=mod;

```

```

49     return res;
50 }

```

#### 4.4 线段树 (重载 + 最大子列和)

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define mid ((l+r)/2)
4  const int maxn=300005;
5  int i,i0,n,m,a[maxn];
6  struct node
7  {
8      long long sum,lm,rm,mx;
9      friend node operator+(node a,node b)
10     {
11         node c;
12         c.sum=a.sum+b.sum;
13         c.lm=max(a.lm,a.sum+b.lm),c.rm=max(b.rm,b.sum+a.rm);
14         c.mx=max(a.mx,b.mx),c.mx=max(c.mx,a.rm+b.lm);
15         return c;
16     }
17 }tree[maxn*4];
18 void b_tree(int l,int r,int p)
19 {
20     if(l==r)tree[p].lm=tree[p].rm=tree[p].mx=max(0,a[l]),tree[p].sum=a[l];
21     else
22     {
23         b_tree(l,mid,p*2),b_tree(mid+1,r,p*2+1);
24         tree[p]=tree[p*2]+tree[p*2+1];
25     }
26 }
27 void c_tree(int l,int r,int p,int x,int y)
28 {
29     if(l==r)tree[p].lm=tree[p].rm=tree[p].mx=max(0,y),tree[p].sum=y;
30     else
31     {
32         if(x<=mid)c_tree(l,mid,p*2,x,y);
33         else c_tree(mid+1,r,p*2+1,x,y);
34         tree[p]=tree[p*2]+tree[p*2+1];
35     }
36 }
37 node q_tree(int l,int r,int p,int a,int b)
38 {
39     if(l==a&&r==b)return tree[p];
40     else
41     {
42         if(b<=mid)return q_tree(l,mid,p*2,a,b);
43         else if(a>=mid+1)return q_tree(mid+1,r,p*2+1,a,b);
44         else return q_tree(l,mid,p*2,a,mid)+q_tree(mid+1,r,p*2+1,mid+1,b);
45     }
46 }
47 int main()

```



```

48 {
49     scanf("%d %d",&n,&m);
50     for(i=1;i<=n;i++)scanf("%d",&a[i]);
51     b_tree(1,n,1);
52     while(m--)
53     {
54         int l,r;
55         scanf("%d %d",&l,&r);
56         printf("%lld\n",q_tree(1,n,1,l,r).mx);
57     }
58     return 0;
59 }

```

#### 4.5 可持久化线段树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int i,i0,n,m,cnt,root[100005],head[100005];
4  struct node
5  {
6      int l,r,sum;
7  }tree[2100005];
8  #define mid (l+r)/2
9  void build_tree(int l,int r,int &p)
10 {
11     p=cnt++;
12     if(l!=r)build_tree(l,mid,tree[p].l),build_tree(mid+1,r,tree[p].r);
13     tree[p].sum=0;
14 }
15 void c_tree(int l,int r,int &p,int old,int a,int v)
16 {
17     p=cnt++,tree[p]=tree[old];
18     if(l==r)
19     {
20         tree[p].sum+=v;
21         return;
22     }
23     if(a<=mid)c_tree(l,mid,tree[p].l,tree[old].l,a,v);
24     else c_tree(mid+1,r,tree[p].r,tree[old].r,a,v);
25     tree[p].sum=tree[tree[p].l].sum+tree[tree[p].r].sum;
26 }
27 int q_tree(int l,int r,int p,int a,int b)
28 {
29     if(l==a&&r==b)return tree[p].sum;
30     if(a>=mid+1)return q_tree(mid+1,r,tree[p].r,a,b);
31     else if(b<=mid) return q_tree(l,mid,tree[p].l,a,b);
32     else return q_tree(mid+1,r,tree[p].r,mid+1,b)+q_tree(l,mid,tree[p].l,a,mid);
33 }
34 struct point
35 {
36     int num,i;
37 }a[100005];

```

```

38 bool cmp(point a,point b){return a.num<b.num;}
39 vector<int>v;
40 int main()
41 {
42     int T;
43     scanf("%d",&T);
44     for(int cas=1;cas<=T;cas++)
45     {
46         printf("Case %d:\n",cas);
47         cnt=0,v.clear();
48         scanf("%d %d",&n,&m);
49         for(i=1;i<=n;i++)scanf("%d",&a[i].num),a[i].i=i,v.push_back(a[i].num);
50         sort(v.begin(),v.end()),v.erase(unique(v.begin(),v.end()),v.end());
51         sort(a+1,a+1+n,cmp);
52         build_tree(1,n,root[0]);
53         for(i=1;i<=n;i++)
54         {
55             a[i].num=lower_bound(v.begin(),v.end(),a[i].num)-v.begin()+1;
56             c_tree(1,n,root[i],root[i-1],a[i].i,1);
57             head[a[i].num]=i;
58         }
59         while(m--)
60         {
61             int l,r,num;
62             scanf("%d %d %d",&l,&r,&num);
63             num=upper_bound(v.begin(),v.end(),num)-v.begin();
64             printf("%d\n",q_tree(1,n,root[head[num]],l+1,r+1));
65         }
66     }
67     return 0;
68 }

```

## 4.6 动态开点线段树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int maxn=5000000;
4  #define mid (l+r)/2
5  long long cnt=0;
6  struct node
7  {
8      long long lson,rson,v,sum;
9  }tree[maxn+5];
10 long long new_node()
11 {
12     long long p=++cnt;
13     tree[p].lson=tree[p].rson=0;
14     tree[p].v=0,tree[p].sum=0;
15     return p;
16 }
17 void p_tree(long long p)
18 {

```

```

19     if(tree[tree[p].lson].v==tree[tree[p].rson].v)tree[p].v=tree[tree[p].lson].v;
20     else tree[p].v=-1;
21     tree[p].sum=tree[tree[p].lson].sum+tree[tree[p].rson].sum;
22 }
23 void new_son(int l,int r,int p){tree[p].lson=new_node(),tree[p].rson=new_node();}
24 void c_tree(long long l,long long r,long long p,long long L,long long R,long long x)
25 {
26     if(l==L&&r==R)tree[p].v=x,tree[p].sum=(R-L+1)*x;
27     else
28     {
29         if(!tree[p].lson)new_son(l,r,p);
30         if(L<=mid)c_tree(l,mid,tree[p].lson,L,min(R,mid),x);
31         if(R>=mid+1)c_tree(mid+1,r,tree[p].rson,max(mid+1,L),R,x);
32         p_tree(p);
33     }
34 }
35 void c0_tree(long long l,long long r,long long p,long long L,long long R,long long x
36 )
37 {
38     if(tree[p].sum==0)return;
39     if(l==L&&r==R)
40     {
41         if(tree[p].v!=-1)
42         {
43             tree[p].v/=x,tree[p].sum=(R-L+1)*tree[p].v;
44         }
45         else
46         {
47             c0_tree(l,mid,tree[p].lson,l,mid,x),c0_tree(mid+1,r,tree[p].rson,mid+1,r,x);
48             p_tree(p);
49         }
50     }
51     else
52     {
53         if(!tree[p].lson)new_son(l,r,p);
54         if(tree[p].v!=-1)
55         {
56             tree[tree[p].lson].v=tree[p].v,tree[tree[p].rson].v=tree[p].v;
57             tree[tree[p].lson].sum=(mid-l+1)*tree[p].v,tree[tree[p].rson].sum=(r-mid)*
58                 tree[p].v;
59         }
60         if(L<=mid)c0_tree(l,mid,tree[p].lson,L,min(R,mid),x);
61         if(R>=mid+1)c0_tree(mid+1,r,tree[p].rson,max(mid+1,L),R,x);
62         p_tree(p);
63     }
64 }
65 long long q_tree(long long l,long long r,long long p,long long L,long long R)
66 {
67     if(tree[p].sum==0)return 0;
68     if(tree[p].v!=-1)return (R-L+1)*tree[p].v;
69     if(l==L&&r==R)return tree[p].sum;
70     else

```

```

69     {
70         long long res=0;
71         if(L<=mid&&tree[p].lson)res+=q_tree(1,mid,tree[p].lson,L,min(R,mid));
72         if(R>=mid+1&&tree[p].rson)res+=q_tree(mid+1,r,tree[p].rson,max(mid+1,L),R);
73         return res;
74     }
75 }
76 int main()
77 {
78     long long root=new_node(),n,m;
79     scanf("%lld %lld",&n,&m);
80     for(long long l=1,r=1;l<=n;l=r+1,r=(l<=n)?n/(n/l):0)c_tree(1,n,root,l,r,n/l);
81     while(m--)
82     {
83         long long op,l,r,x;
84         scanf("%lld %lld %lld",&op,&l,&r);
85         if(op==1)
86         {
87             scanf("%lld",&x);
88             if(x!=1)c0_tree(1,n,root,l,r,x);
89         }
90         else printf("%lld\n",q_tree(1,n,root,l,r));
91     }
92     return 0;
93 }

```

#### 4.7 李超树

```

1  #define mid (l+r)/2
2  struct line
3  {
4      long long k,b;
5      bool flag;
6  }tree[400005];
7  long long cal(line a,long long pos){return a.k*pos+a.b;}
8  long long cross(line a,line b){return (a.b-b.b)/(b.k-a.k);}
9  //max
10 void c_tree(long long l,long long r,long long p,line k)
11 {
12     if(!tree[p].flag)
13     {
14         tree[p]=k;
15         if(l!=r)tree[p*2].flag=tree[p*2+1].flag=0;
16     }
17     else if(cal(k,l)>cal(tree[p],l)&&cal(k,r)>cal(tree[p],r))tree[p]=k;
18     else if(cal(k,l)>cal(tree[p],l)||cal(k,r)>cal(tree[p],r))
19     {
20         if((cross(k,tree[p])<=mid)==(k.k>=tree[p].k))swap(k,tree[p]);
21         if(cross(k,tree[p])<=mid)c_tree(l,mid,p*2,k);
22         else c_tree(mid+1,r,p*2+1,k);
23     }
24 }

```

```

25 long long q_tree(long long l,long long r,long long p,long long x)
26 {
27     if(!tree[p].flag)return INT64_MIN;
28     if(l==r)return cal(tree[p],x);
29     else
30     {
31         long long ans=cal(tree[p],x);
32         if(x<=mid)return max(ans,q_tree(l,mid,p*2,x));
33         else return max(ans,q_tree(mid+1,r,p*2+1,x));
34     }
35 }
36 //min and segment
37 void c_tree(int l,int r,int p,line k)
38 {
39     if(k.l<=l&&r<=k.r)
40     {
41         if(!tree[p].flag)
42         {
43             tree[p]=k;
44             if(l==r)tree[p*2].flag=tree[p*2+1].flag=0;
45         }
46         else if(cal(k,l)<=cal(tree[p],l)&&cal(k,r)<=cal(tree[p],r))tree[p]=k;
47         else if((cal(k,l)<=cal(tree[p],l))!=(cal(k,r)<=cal(tree[p],r)))
48         {
49             if((cross(k,tree[p])<=mid)!=(k.k>=tree[p].k))swap(k,tree[p]);
50             if(cross(k,tree[p])<=mid)c_tree(l,mid,p*2,k);
51             else c_tree(mid+1,r,p*2+1,k);
52         }
53     }
54     else
55     {
56         int mid=(l+r)/2;
57         if(k.l<=mid)c_tree(l,mid,p*2,k);
58         if(mid<k.r)c_tree(mid+1,r,p*2+1,k);
59     }
60 }
61 int q_tree(int l,int r,int p,int x)
62 {
63     if(!tree[p].flag)return INT_MAX/2;
64     if(l==r)return cal(tree[p],x);
65     else
66     {
67         int ans=cal(tree[p],x);
68         if(x<=mid)return min(ans,q_tree(l,mid,p*2,x));
69         else return min(ans,q_tree(mid+1,r,p*2+1,x));
70     }
71 }

```

#### 4.8 整体二分

```

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

```

```

3  int i,i0,n,m,ans[100005],bit[100005],f,bac[100005];
4  struct node
5  {
6      int num,i,l,r,tpe;
7  }tmp,tmp0;
8  int lowbit(int t){return t&(-t);}
9  int sum(int i)
10 {
11     int s=0;
12     while(i>0)s+=bit[i],i-=lowbit(i);
13     return s;
14 }
15 void add(int i,int x){while(i<=n+1)bit[i]+=x,i+=lowbit(i);}
16 queue<node>q[35][2];
17 void all_binary(int l,int r,int dep,bool f)
18 {
19     if(q[dep][f].empty())return;
20     if(l==r)
21     {
22         while(!q[dep][f].empty())
23         {
24             tmp=q[dep][f].front(),q[dep][f].pop();
25             if(tmp.tpe==1)ans[tmp.i]=max(ans[tmp.i],min(tmp.r-tmp.l+1-tmp.num,1));
26         }
27         return;
28     }
29     int mid=(l+r+1)/2,tcnt;
30     while(!q[dep][f].empty())
31     {
32         tmp=q[dep][f].front(),q[dep][f].pop();
33         if(tmp.tpe==1)
34         {
35             tcnt=sum(tmp.r)-sum(tmp.l-1);
36             if(tmp.r-tmp.l+1-tcnt-tmp.num<mid)q[dep+1][0].push(tmp);
37             else tmp.num+=tcnt,q[dep+1][1].push(tmp);
38         }
39         if(tmp.tpe==2)
40         {
41             if(tmp.num<mid)
42             {
43                 add(tmp.i,1);
44                 q[dep+1][0].push(tmp);
45             }
46             else
47             {
48                 q[dep+1][1].push(tmp);
49             }
50         }
51         if(tmp.tpe==3)
52         {
53             if(tmp.num<mid)
54             {
55                 add(tmp.i,-1);

```

```

56         q[dep+1][0].push(tmp);
57     }
58     else
59     {
60         q[dep+1][1].push(tmp);
61     }
62 }
63 }
64 all_binary(1,mid-1,dep+1,0);
65 all_binary(mid,r,dep+1,1);
66 }
67 int main()
68 {
69     while(scanf("%d %d",&n,&m)!=EOF)
70     {
71         memset(ans,-1,sizeof(ans));
72         tmp.tpe=2;
73         for(i=1;i<=n;i++)
74         {
75             scanf("%d",&bac[i]);
76             tmp.num=bac[i],tmp.i=i;
77             q[0][0].push(tmp);
78         }
79         for(i=0;i<m;i++)
80         {
81             scanf("%d",&tmp.tpe);
82             if(tmp.tpe==1)scanf("%d %d",&tmp.l,&tmp.r),tmp.num=0,tmp.i=i,q[0][0].push(
83                 tmp);
84             if(tmp.tpe==2)
85             {
86                 scanf("%d %d",&tmp.i,&tmp.num);
87                 tmp0.tpe=3,tmp0.i=tmp.i,tmp0.num=bac[tmp.i];
88                 q[0][0].push(tmp0);
89                 q[0][0].push(tmp);
90                 bac[tmp.i]=tmp.num;
91             }
92             tmp.tpe=3;
93             for(i=1;i<=n;i++)
94             {
95                 tmp.num=bac[i],tmp.i=i;
96                 q[0][0].push(tmp);
97             }
98             all_binary(1,n,0,0);
99             for(i=0;i<m;i++)if(ans[i]!=-1)printf("%d\n",ans[i]);
100         }
101         return 0;
102     }

```

## 4.9 扩展整体二分

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

```

2 using namespace std;
3 #define mid ((l+r)/2)
4 const int maxn=100000;
5 int ans[3*maxn+5],tree[maxn*4+5];
6 struct node
7 {
8     int tpe,id;//tpe为1表示插入,2表示查询,3表示删除。id表示编号,来确保输出答案时有序。
9     int x,y,num;//num表示权值,x、y表示坐标。
10    int lx,rx,ly,ry;//表示查询范围。
11 };
12 void c_tree(int l,int r,int p,int x,int num)
13 {
14     if(l==r)
15     {
16         tree[p]=max(tree[p],num);
17         if(!num)tree[p]=0;
18     }
19     else
20     {
21         if(x<=mid)c_tree(l,mid,p*2,x,num);
22         else c_tree(mid+1,r,p*2+1,x,num);
23         tree[p]=max(tree[p*2],tree[p*2+1]);
24     }
25 }
26 int q_tree(int l,int r,int p,int L,int R)
27 {
28     if(l==L&&r==R)return tree[p];
29     else
30     {
31         if(R<=mid)return q_tree(l,mid,p*2,L,R);
32         else if(L>mid)return q_tree(mid+1,r,p*2+1,L,R);
33         else return max(q_tree(l,mid,p*2,L,mid),q_tree(mid+1,r,p*2+1,mid+1,R));
34     }
35 }
36 queue<node>q[25][2];
37 void all_binary(int l,int r,int dep,bool f)
38 {
39     if(q[dep][f].empty())return;
40     while(!q[dep][f].empty())
41     {
42         node tmp=q[dep][f].front();
43         q[dep][f].pop();
44         if(tmp.tpe==1)
45         {
46             c_tree(1,maxn,1,tmp.x,tmp.num);
47             if(l!=r)
48             {
49                 if(tmp.y<=mid)q[dep+1][0].push(tmp);
50                 else q[dep+1][1].push(tmp);
51             }
52         }
53         if(tmp.tpe==2)
54         {

```



```

55         if(tmp.ly==l&&tmp.ry==r)ans[tmp.id]=max(ans[tmp.id],q_tree(1,maxn,1,tmp.lx
           ,tmp.rx));
56     else
57     {
58         if(l!=r)
59         {
60             if(tmp.ly<=mid)
61             {
62                 node tmp0=tmp;
63                 tmp0.ry=min(tmp0.ry,mid);
64                 q[dep+1][0].push(tmp0);
65             }
66             if(tmp.ry>mid)
67             {
68                 node tmp0=tmp;
69                 tmp0.ly=max(tmp0.ly,mid+1);
70                 q[dep+1][1].push(tmp0);
71             }
72         }
73     }
74 }
75 if(tmp.tpe==3)
76 {
77     c_tree(1,maxn,1,tmp.x,0);
78     if(l!=r)
79     {
80         if(tmp.y<=mid)q[dep+1][0].push(tmp);
81         else q[dep+1][1].push(tmp);
82     }
83 }
84 }
85 all_binary(l,mid,dep+1,0),all_binary(mid+1,r,dep+1,1);
86 }
87 vector<node>v;
88 int main()
89 {
90     int n;
91     scanf("%d",&n);
92     for(int i=1;i<=n;i++)
93     {
94         node tmp;
95         ans[i]=-1;
96         tmp.id=i;
97         scanf("%d",&tmp.tpe);
98         if(tmp.tpe==1)scanf("%d %d %d",&tmp.x,&tmp.y,&tmp.num),v.push_back(tmp);
99         if(tmp.tpe==2)scanf("%d %d %d %d",&tmp.lx,&tmp.rx,&tmp.ly,&tmp.ry);
100         q[0][0].push(tmp);
101     }
102     for(node tmp:v)
103     {
104         tmp.tpe=3;
105         q[0][0].push(tmp);
106     }

```

```

107     all_binary(1,maxn,0,0);
108     for(int i=1;i<=n;i++)if(ans[i]!=-1)printf("%d\n",ans[i]);
109     return 0;
110 }

```

## 4.10 字典树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int i,i0,n,m,tree[1000005][26],num,sum[1000005],cnt;
4  char s[11];
5  void c_tree()
6  {
7      int len=strlen(s),root=0;
8      for(int i=0;i<len;i++)
9      {
10         if(!tree[root][s[i]-'a'])tree[root][s[i]-'a']=++cnt;
11         sum[tree[root][s[i]-'a']]++;
12         root=tree[root][s[i]-'a'];
13     }
14 }
15 int q_tree()
16 {
17     int len=strlen(s),root=0;
18     for(int i=0;i<len;i++)
19     {
20         root=tree[root][s[i]-'a'];
21         if(!root)return 0;
22     }
23     return sum[root];
24 }
25 int main()
26 {
27     while(gets(s)&&s[0]!='\0')c_tree();
28     while(gets(s)!=NULL) printf("%d\n",q_tree());
29     return 0;
30 }

```

## 4.11 块状链表

```

1  #include <cstdio>
2  #include <cstring>
3  #include<algorithm>
4  using namespace std;
5  const int maxn = 1500;
6  int pos, Size[maxn], List[maxn], Next[maxn];
7  char s[2000000], c, data[maxn][maxn], cmd[20];
8  int New_Node(void)
9  {
10     return List[pos++];
11 }

```

```
12
13 void Del_Node(int t)
14 {
15     List[--pos] = t;
16 }
17
18 void Find(int& p, int& b)
19 {
20     for (b = 0; b != -1 && p > Size[b]; b = Next[b])
21     {
22         p -= Size[b];
23     }
24 }
25
26 void Fill_Block(int b, int n, char* str, int e)
27 {
28     if (b == -1)
29     {
30         return;
31     }
32     Next[b] = e;
33     Size[b] = n;
34     memcpy(data[b], str, n);
35 }
36
37 void Split(int b, int p)
38 {
39     if (b == -1 || p == Size[b])
40     {
41         return;
42     }
43     int t = New_Node();
44     Fill_Block(t, Size[b]-p, data[b]+p, Next[b]);
45     Next[b] = t;
46     Size[b] = p;
47 }
48
49 void Maintain(int b)
50 {
51     for (; b != -1; b = Next[b])
52     {
53         for (int t = Next[b]; t != -1 && Size[b]+Size[t] <= maxn; t = Next[b])
54         {
55             memcpy(data[b]+ Size[b], data[t], Size[t]);
56             Size[b] += Size[t];
57             Next[b] = Next[t];
58             Del_Node(t);
59         }
60     }
61 }
62
63 void Insert(int p, int n, char* str)
64 {
```

```
65     int b, t, i;
66     Find(p, b);
67     Split(b, p);
68     for (i = 0; i+maxn <= n; i += maxn)
69     {
70         t = New_Node();
71         Fill_Block(t, maxn, str+i, Next[b]);
72         Next[b] = t;
73         b = t;
74     }
75     if (n-i)
76     {
77         t = New_Node();
78         Fill_Block(t, n-i, str+i, Next[b]);
79         Next[b] = t;
80     }
81     Maintain(b);
82 }
83
84 void Erase(int p, int n)
85 {
86     int b, e;
87     Find(p, b);
88     Split(b, p);
89     for (e = Next[b]; e != -1 && n > Size[e]; e = Next[e])
90     {
91         n -= Size[e];
92     }
93     Split(e, n);
94     e = Next[e];
95     for (int t = Next[b]; t != e; t = Next[t])
96     {
97         Next[b] = Next[t];
98         Del_Node(t);
99     }
100     Maintain(b);
101 }
102
103 void Copy(int p, int n, char* str)
104 {
105     int b, t, i;
106     Find(p, b);
107     i = min(n, Size[b]-p);
108     memcpy(str, data[b]+p, i);
109     for (t = Next[b]; t != -1 && i+Size[t] <= n; i += Size[t], t = Next[t])
110     {
111         memcpy(str+i, data[t], Size[t]);
112     }
113     if (n-i && t != -1)
114     {
115         memcpy(str+i, data[t], n-i);
116     }
117 }
```

```
118
119 void Init(void)
120 {
121     for (int i = 1; i < maxn; ++i)
122     {
123         List[i] = i;
124     }
125     pos = 1;
126     Next[0] = -1;
127     Size[0] = 0;
128 }
129
130 int main()
131 {
132
133     int t, cur = 0, n;
134     Init();
135     scanf("%d", &t);
136     while (t--)
137     {
138         scanf("%s", cmd);
139         if (cmd[0] == 'M')
140         {
141             scanf("%d", &cur);
142         }
143         else if (cmd[0] == 'I')
144         {
145             scanf("%d", &n);
146             for (int i = 0; i < n; i++)
147             {
148                 c = getchar();
149                 if (32 <= c && c <= 126)
150                 {
151                     s[i++] = c;
152                 }
153             }
154             s[n] = '\0';
155             Insert(cur, n, s);
156         }
157         else if (cmd[0] == 'D')
158         {
159             scanf("%d", &n);
160             Erase(cur, n);
161         }
162         else if (cmd[0] == 'G')
163         {
164             scanf("%d", &n);
165             Copy(cur, n, s);
166             s[n] = '\0';
167             printf("%s\n", s);
168         }
169         else if (cmd[0] == 'P')
170         {
```

```

171         --cur;
172     }
173     else
174     {
175         ++cur;
176     }
177 }
178 return 0;
179 }

```

## 5 数论

### 5.1 GCD、LCM、EXGCD

```

1 long long lcm(long long a,long long b){return a/__gcd(a,b)*b;}
2 int exgcd(int a,int b,int &x,int &y)
3 {
4     if(b==0)
5     {
6         x=1,y=0;
7         return a;
8     }
9     int r=exgcd(b,a%b,x,y),t=x;
10    x=y,y=t-a/b*y;
11    return r;
12 }

```

### 5.2 乘法逆元

```

1 void extgcd(long long a,long long b,long long& d,long long& x,long long& y)
2 {
3     if(!b){d=a;x=1;y=0;}
4     else{extgcd(b,a%b,d,y,x);y-=x*(a/b);}
5 }
6 long long inv(long long a,long long n)//要掰成正数来求
7 {
8     if(a<0)a+=mod;
9     long long d,x,y;
10    extgcd(a,n,d,x,y);
11    return d==1?(x+n)%n:-1;
12 }
13
14
15 inv[1]=1;
16 for(i=2;i<=7000;i++)inv[i]=(MOD-MOD/i)*inv[MOD%i]%MOD;
17
18 static int mod_inv(int a, int m = MOD) {
19     // https://en.wikipedia.org/wiki/Extended\_Euclidean\_algorithm#Example
20     int g = m, r = a, x = 0, y = 1;
21
22     while (r != 0) {

```

```

23     int q = g / r;
24     g %= r; swap(g, r);
25     x -= q * y; swap(x, y);
26 }
27
28 return x < 0 ? x + m : x;
29 }

```

### 5.3 MillerRobin、PollardRho

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int bace[5]={2,3,7,61,24251};
4  long long qmul(long long x,long long y,long long mod){return (x*y-(long long)((long
   double)x/mod*y)*mod+mod)%mod;}
5  long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
   (b&1)r=qmul(r,t,mod);b>>=1;t=qmul(t,t,mod);}return r;}
6  bool millerrabin(long long x)
7  {
8      if(x==4685624825598111||x<2)return false;
9      if(x==2||x==3||x==7||x==61||x==24251)return true;
10     long long ba=x-1,r;
11     int ti=0,j;
12     while(!(ba&1))ba>>=1,++ti;
13     for(int i=0;i<=1;i++)
14     {
15         r=qpow(bace[i],ba,x);
16         if(r==1||r==x-1)continue;
17         for(j=1;j<=ti;++j)
18         {
19             r=qmul(r,r,x);
20             if(r==x-1)break;
21         }
22         if(j>ti)return false;
23     }
24     return true;
25 }
26 mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
27 long long pollardpho(long long n,int c)
28 {
29     long long x,y,d;
30     int i=1,k=2;
31     y=x=rng()(n-1)+1;
32     while(++i)
33     {
34         x=(qmul(x,x,n)+c)%n;
35         d=__gcd(y-x,n);
36         if(d>1&&d<n)return d;
37         if(x==y)return n;
38         if(i==k)y=x,k<<=1;
39     }
40 }

```

```

41 vector<long long>v0;
42 void fin(long long x,int cnt)
43 {
44     if(x==1)return;
45     if(millerrabin(x))
46     {
47         v0.push_back(x);
48         return;
49     }
50     long long p=x;
51     while(p==x)p=pollardpho(x,cnt--);
52     fin(p,cnt),fin(x/p,cnt);
53 }
54 int main()
55 {
56     long long n,T;
57     scanf("%lld",&T);
58     while(T--)
59     {
60         scanf("%lld",&n);
61         v0.clear();
62         if(n==1)v0.push_back(1);
63         fin(n,120);
64         sort(v0.begin(),v0.end());
65         for(int i=0;i<v0.size();i++)printf("%d%c",v0[i],i==v0.size()-1?'\\n':' ');
66     }
67     return 0;
68 }

```

## 5.4 快速幂乘

```

1 long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
  (b&1)r=(r*t)%mod;b>>=1;t=(t*t)%mod;}return r;}
2
3
4 long long qmul(long long x,long long y,long long mod){return (x*y-(long long)((long
  double)x/mod*y)*mod+mod)%mod;}
5 long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
  (b&1)r=qmul(r,t,mod);b>>=1;t=qmul(t,t,mod);}return r;}

```

## 5.5 矩阵快速幂

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int MAXN=4,MOD=10007;
4 struct MAT
5 {
6     int mat[MAXN][MAXN];
7     MAT operator*(const MAT &a)const
8     {
9         MAT b;

```



```

10     memset(b.mat,0,sizeof(b.mat));
11     for(int i=0;i<MAXN;i++)
12     {
13         for(int j=0;j<MAXN;j++)
14         {
15             for(int k=0;k<MAXN;k++)b.mat[i][j]=(b.mat[i][j]+mat[i][k]*a.mat[k][j]);
16             b.mat[i][j]%=MOD;
17         }
18     }
19     return b;
20 }
21 };
22 MAT Mqpow(MAT base,int b)
23 {
24     MAT r;
25     for(int i=0;i<MAXN;i++)for(int j=0;j<MAXN;j++)r.mat[i][j]=i==j;
26     while(b)
27     {
28         if(b&1)r=base*r;
29         base=base*base;
30         b>>=1;
31     }
32     return r;
33 }
34 int main()
35 {
36     int n;
37     while(scanf("%d",&n)!=EOF)
38     {
39         MAT start,r;
40         //转移矩阵 横着为f[i][0]=2*f[i-1][0]+f[i-1][1]+f[i-1][2]....
41         start.mat[0][0]=2,start.mat[0][1]=1,start.mat[0][2]=1,start.mat[0][3]=0;
42         start.mat[1][0]=1,start.mat[1][1]=2,start.mat[1][2]=0,start.mat[1][3]=1;
43         start.mat[2][0]=1,start.mat[2][1]=0,start.mat[2][2]=2,start.mat[2][3]=1;
44         start.mat[3][0]=0,start.mat[3][1]=1,start.mat[3][2]=1,start.mat[3][3]=2;
45         //r第一列为初始值 f[0][3]=1...;
46         r.mat[0][0]=0,r.mat[0][1]=0,r.mat[0][2]=0,r.mat[0][3]=0;
47         r.mat[1][0]=0,r.mat[1][1]=0,r.mat[1][2]=0,r.mat[1][3]=0;
48         r.mat[2][0]=0,r.mat[2][1]=0,r.mat[2][2]=0,r.mat[2][3]=0;
49         r.mat[3][0]=1,r.mat[3][1]=0,r.mat[3][2]=0,r.mat[3][3]=0;
50
51         printf("%d\n", (Mqpow(start,n)*r).mat[3][0]);
52     }
53     return 0;
54 }

```

## 5.6 快速阶乘

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define R register
4 #define ll long long

```

```

5 #define fp(i,a,b) for(R int i=(a),I=(b)+1;i<I;++i)
6 #define fd(i,a,b) for(R int i=(a),I=(b)-1;i>I;--i)
7 #define go(u) for(int i=head[u],v=e[i].v;i;i=e[i].nx,v=e[i].v)
8 const int N=(1<<17)+5;int P;
9 inline int add(R int x,R int y){return 011+x+y>=P?011+x+y-P:x+y;}
10 inline int dec(R int x,R int y){return x-y<0?x-y+P:x-y;}
11 inline int mul(R int x,R int y){return 111*x*y-111*x*y/P*P;}
12 int ksm(R int x,R int y){
13     R int res=1;
14     for(;y;y>>=1,x=mul(x,x))(y&1)?res=mul(res,x):0;
15     return res;
16 }
17 const double Pi=acos(-1.0);
18 struct cp{
19     double x,y;
20     inline cp(){}
21     inline cp(R double xx,R double yy):x(xx),y(yy){}
22     inline cp operator +(const cp &b)const{return cp(x+b.x,y+b.y);}
23     inline cp operator -(const cp &b)const{return cp(x-b.x,y-b.y);}
24     inline cp operator *(const cp &b)const{return cp(x*b.x-y*b.y,x*b.y+y*b.x);}
25     inline cp operator *(const double &b)const{return cp(x*b,y*b);}
26     inline cp operator ~()const{return cp(x,-y);}
27 }w[2][N];
28 int r[21][N],ifac[N],lg[N],inv[N];double iv[21];
29 void Pre(){
30     iv[0]=1;
31     fp(d,1,17){
32         fp(i,0,(1<<d)-1)r[d][i]=(r[d][i>>1]>>1)|((i&1)<<(d-1));
33         lg[1<<d]=d,iv[d]=iv[d-1]*0.5;
34     }
35     inv[0]=inv[1]=ifac[0]=ifac[1]=1;
36     fp(i,2,131072)inv[i]=mul(P-P/i,inv[P%i]),ifac[i]=mul(ifac[i-1],inv[i]);
37     for(R int i=1,d=0;i<131072;i<=1,++d)fp(k,0,i-1)
38         w[1][i+k]=cp(cos(Pi*k*iv[d]),sin(Pi*k*iv[d])),
39         w[0][i+k]=cp(cos(Pi*k*iv[d]),-sin(Pi*k*iv[d]));
40 }
41 int lim,d;
42 void FFT(cp *A,int ty){
43     fp(i,0,lim-1)if(i<r[d][i])swap(A[i],A[r[d][i]]);
44     cp t;
45     for(R int mid=1;mid<lim;mid<=1)
46         for(R int j=0;j<lim;j+=(mid<<1))
47             fp(k,0,mid-1)
48                 A[j+k+mid]=A[j+k]-(t=w[ty][mid+k]*A[j+k+mid]),
49                 A[j+k]=A[j+k]+t;
50     if(!ty)fp(i,0,lim-1)A[i]=A[i]*iv[d];
51 }
52 void MTT(int *a,int *b,int len,int *c){
53     static cp f[N],g[N],p[N],q[N];
54     lim=len,d=lg[lim];
55     fp(i,0,len-1)f[i]=cp(a[i]>>16,a[i]&65535),g[i]=cp(b[i]>>16,b[i]&65535);
56     fp(i,len,lim-1)f[i]=g[i]=cp(0,0);
57     FFT(f,1),FFT(g,1);

```

```

58     fp(i,0,lim-1){
59         cp t,f0,f1,g0,g1;
60         t=~f[i?lim-i:0],f0=(f[i]-t)*cp(0,-0.5),f1=(f[i]+t)*0.5;
61         t=~g[i?lim-i:0],g0=(g[i]-t)*cp(0,-0.5),g1=(g[i]+t)*0.5;
62         p[i]=f1*g1,q[i]=f1*g0+f0*g1+f0*g0*cp(0,1);
63     }
64     FFT(p,0),FFT(q,0);
65     fp(i,0,lim-1)c[i]=((((11)(p[i].x+0.5)%P<<16)%P<<16)+((11)(q[i].x+0.5)<<16)+((11)
        (q[i].y+0.5)))%P;
66 }
67 void calc(int *a,int *b,int n,int k){
68     static int f[N],g[N],h[N],sum[N],isum[N];
69     int len=1;while(len<=n+n)len<<=1;
70     fp(i,0,n)f[i]=mul(a[i],mul(ifac[i],ifac[n-i]));
71     for(R int i=n-1;i>=0;i-=2)f[i]=P-f[i];
72     int t=dec(k,n);
73     fp(i,0,n+n)g[i]=add(i,t);
74     sum[0]=g[0];fp(i,1,n+n)sum[i]=mul(sum[i-1],g[i]);
75     isum[n+n]=ksm(sum[n+n],P-2);
76     fd(i,n+n,1)isum[i-1]=mul(isum[i],g[i]);
77     fp(i,1,n+n)g[i]=mul(isum[i],sum[i-1]);g[0]=isum[0];
78     fp(i,n+1,len-1)f[i]=0;fp(i,n+n+1,len-1)g[i]=0;
79
80     MTT(f,g,len,h);
81     int res=1,p1=k-n,p2=k;
82     fp(i,p1,p2)res=111*res*i%P;
83     res=add(res,0);
84
85     fp(i,0,n)g[i]=(011+P+p1+i)%P;
86     sum[0]=g[0];fp(i,1,n)sum[i]=mul(sum[i-1],g[i]);
87     isum[n]=ksm(sum[n],P-2);
88     fd(i,n,1)isum[i-1]=mul(isum[i],g[i]);
89     fp(i,1,n)g[i]=mul(isum[i],sum[i-1]);g[0]=isum[0];
90
91     for(R int i=0;i<=n;p2=add(p2,1),++i)
92         b[i]=mul(h[i+n],res),res=mul(res,mul(g[i],p2+1));
93 }
94 int solve(int b1){
95     static int a[N],b[N],c[N];
96     int s=0;for(int p=b1;p>=1)++s;a[0]=1,--s;
97     int qwq=ksm(b1,P-2);
98     for(int p=0;s>=0;--s){
99         if(p){
100             calc(a,b,p,p+1);
101             fp(i,0,p)a[p+i+1]=b[i];a[p<<1|1]=0;
102             calc(a,b,p<<1,mul(p,qwq));
103             p<<=1;fp(i,0,p)a[i]=mul(a[i],b[i]);
104         }
105         if(b1>>s&1){
106             fp(i,0,p)a[i]=mul(a[i],(111*b1*i+p+1)%P);
107             p|=1,a[p]=1;
108             fp(i,1,p)a[p]=mul(a[p],(111*b1*p+i)%P);
109         }

```

```

110     }
111     int res=1;
112     fp(i,0,b1-1)res=mul(res,a[i]);
113     return res;
114 }
115 int GetFac(int n){
116     int s=sqrt(n),res=solve(s);
117     fp(i,s*s+1,n)res=mul(res,i);
118     return res;
119 }
120 int Fac(int n){
121     if(n>P-1-n){
122         int res=ksm(GetFac(P-1-n),P-2);
123         return n&1?res:P-res;
124     }
125     return GetFac(n);
126 }
127 int n;
128 int main(){
129     scanf("%d",&n,&P),Pre();
130     printf("%d\n",Fac(n));
131     return 0;
132 }

```

## 5.7 欧拉函数

```

1  int euler(int n)
2  {
3      int ret=n,t=(int)sqrt(n*1.0);
4      for(int i=2;i<=t;i++)
5      {
6          if(n%i==0)
7          {
8              ret=ret/i*(i-1);
9              while(n%i==0)n/=i;
10         }
11     }
12     if(n>1) ret=ret/n*(n-1);
13     return ret;
14 }
15
16 for(i=1;i<=3000000;i++)p[i]=i;
17 for(i=2;i<=3000000;i+=2)p[i]/=2;
18 for(i=3;i<=3000000;i+=2)
19 if(p[i]==i)
20 {
21     for(j=i;j<=3000000;j+=i)
22         p[j]=p[j]/i*(i-1);
23 }

```

## 5.8 欧拉降幂

```

1 //a^b mod p
2 long long Dphi(int a,int b,int p)
3 {
4     int mod=phi(p);
5     return qpow(a,b<mod?b:b%mod+mod,p);
6 }

```

## 5.9 线性基

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int n,m,i;
4 struct Linebasis
5 {
6     typedef unsigned int bint;
7     const static int sz=32;
8     bint p[sz];
9     void init(){memset(p, 0, sizeof(p));}
10    bool insert(bint x,bool f=1)
11    {
12        for(int i=sz-1;i>=0;i--)
13        {
14            if((x>>i)&1)
15            {
16                if(!p[i])
17                {
18                    if(f)p[i]=x;
19                    break;
20                }
21                x^=p[i];
22            }
23        }
24        return x;
25    }
26    Linebasis operator |(const Linebasis &r)const
27    {
28        Linebasis res=r;
29        for(int i=0;i<sz;i++)if(p[i])res.insert(p[i]);
30        return res;
31    }
32    Linebasis operator &(const Linebasis &r)const
33    {
34        Linebasis All,C,D;
35        All.init(),C.init(),D.init();
36        for (int i =sz-1;i>=0;i--)All.p[i]=this->p[i],D.p[i]=1ll<<i;
37        for (int i =sz-1;i>=0;i--)
38        {
39            if(r.p[i])
40            {
41                bint v=r.p[i],k=0;
42                bool f=1;

```

```

43         for (int j=sz-1;j>=0;j--)
44         {
45             if(v&(1ll<<j))
46             {
47                 if (All.p[j])v^=All.p[j],k^=D.p[j];
48                 else
49                 {
50                     f=0,All.p[j]=v,D.p[j]=k;
51                     break;
52                 }
53             }
54         }
55         if(f)
56         {
57             bint v=0;
58             for (int j=sz-1;j>=0;j--)if(k&(1ll<<j))v^=this->p[j];
59             C.insert(v);
60         }
61     }
62 }
63 return C;
64 }
65 bint get_max(bint x)
66 {
67     for(int i=sz-1;i>=0;i--)x=max(x,x^p[i]);
68     return x;
69 }
70 bint get_min(bint x)
71 {
72     for(int i=sz-1;i>=0;i--)x=min(x,x^p[i]);
73     return x;
74 }
75 }tree[50005*4];
76 vector<unsigned int>v[50005];
77 #define mid (l+r)/2
78 void b_tree(int l,int r,int p)
79 {
80     if(l==r)for(unsigned int x:v[l])tree[p].insert(x);
81     else
82     {
83         b_tree(l,mid,p*2),b_tree(mid+1,r,p*2+1);
84         tree[p]=tree[p*2]&tree[p*2+1];
85     }
86 }
87 bool q_tree(int l,int r,int p,int a,int b,int x)
88 {
89     if(l==a&&r==b)return !tree[p].insert(x,0);
90     else
91     {
92         if(b<=mid)return q_tree(l,mid,p*2,a,b,x);
93         else if(a>=mid+1)return q_tree(mid+1,r,p*2+1,a,b,x);
94         else return q_tree(l,mid,p*2,a,mid,x)&&q_tree(mid+1,r,p*2+1,mid+1,b,x);
95     }

```

```

96 }
97 int main()
98 {
99     scanf("%d %d",&n,&m);
100     for(i=1;i<=n;i++)
101     {
102         int k;
103         scanf("%d",&k);
104         while(k--)
105         {
106             unsigned int x;
107             scanf("%u",&x);
108             v[i].push_back(x);
109         }
110     }
111     b_tree(1,n,1);
112     while(m--)
113     {
114         int l,r;
115         unsigned int x;
116         scanf("%d %d %u",&l,&r,&x);
117         if(q_tree(1,n,1,l,r,x))printf("YES\n");
118         else printf("NO\n");
119     }
120     return 0;
121 }

```

## 5.10 线性筛

```

1  const int MAXN=10000000;
2  vector<int>prime;
3  int check[MAXN+5],phi[MAXN+5],mob[MAXN+5],fac[MAXN+5],tim[MAXN+5],sd[MAXN+5],sp[MAXN
   +5];
4  void sieve()
5  {
6      phi[1]=1; //欧拉筛
7      mob[1]=1; //莫比乌斯函数筛
8      fac[1]=1; //约数个数筛
9      tim[1]=0; //最小质因子次幂
10     sd[1]=1; //约数和筛
11     sp[1]=1; //最小质因子的等比数列和 (1+p+p^2+p^3+...+p^k)
12     for(int i=2;i<=MAXN;i++)
13     {
14         if(check[i]==0)
15         {
16             prime.push_back(i);
17             phi[i]=i-1;
18             mob[i]=-1;
19             fac[i]=2;
20             tim[i]=1;
21             sd[i]=i+1;
22             sp[i]=i+1;

```

```

23     }
24     for(int j=0;j<prime.size()&&i*prime[j]<=MAXN;j++)
25     {
26         check[i*prime[j]]=1;
27         if(i%prime[j]==0)
28         {
29             phi[i*prime[j]]=phi[i]*prime[j];
30             mob[i*prime[j]]=0;
31             fac[i*prime[j]]=fac[i]/(tim[i]+1)*(tim[i]+2);
32             tim[i*prime[j]]=tim[i]+1;
33             sd[i*prime[j]]=sd[i]/sp[i]*(sp[i]*prime[j]+1);
34             sp[i*prime[j]]=sp[i]*prime[j]+1;
35             break;
36         }
37         phi[i*prime[j]]=phi[i]*(prime[j]-1);
38         mob[i*prime[j]]=-mob[i];
39         fac[i*prime[j]]=fac[i]*2;
40         tim[i*prime[j]]=1;
41         sd[i*prime[j]]=sd[i]*(prime[j]+1);
42         sp[i*prime[j]]=prime[j]+1;
43     }
44 }
45 }

```

## 5.11 杜教筛

```

1  #include<bits/stdc++.h>
2  #include<ext/pb_ds/assoc_container.hpp>
3  using namespace __gnu_pbds;
4  using namespace std;
5  int i,i0,n,m,T,ans;
6  const int MAXN=7000000;
7  vector<int>prime;
8  bool check[MAXN+5];
9  short mob[MAXN+5],premob[MAXN+5];
10 int phi[MAXN+5];
11 long long prephi[MAXN+5];
12 void sieve()
13 {
14     prephi[1]=phi[1]=1; //欧拉筛
15     premob[1]=mob[1]=1; //莫比乌斯函数筛
16     for(int i=2;i<=MAXN;i++)
17     {
18         if(check[i]==0)
19         {
20             prime.push_back(i);
21             phi[i]=i-1;
22             mob[i]=-1;
23         }
24         for(int j=0;j<prime.size()&&i*prime[j]<=MAXN;j++)
25         {
26             check[i*prime[j]]=1;

```



```

27         if(i%prime[j]==0)
28         {
29             phi[i*prime[j]]=phi[i]*prime[j];
30             mob[i*prime[j]]=0;
31             break;
32         }
33         phi[i*prime[j]]=phi[i]*(prime[j]-1);
34         mob[i*prime[j]]=-mob[i];
35     }
36     prephi[i]=prephi[i-1]+phi[i];
37     premob[i]=premob[i-1]+mob[i];
38 }
39 }
40 namespace MU
41 {
42     long long get_preFcovG(int n)
43     {
44         return 1;
45     }
46     long long get_preG(int n)
47     {
48         return n;
49     }
50     gp_hash_table<int,long long>F;
51     long long get_preF(int n)
52     {
53         if(n<=MAXN)return premob[n];
54         if(F.find(n)!=F.end())return F[n];
55         long long ans=get_preFcovG(n);
56         for(long long l=2,r;l<=n;l=r+1)
57         {
58             r=n/(n/l);
59             ans-=(get_preG(r)-get_preG(l-1))*get_preF(n/l);
60         }
61         return F[n]=ans;
62     }
63 }
64 namespace PHI
65 {
66     unsigned long long get_preFcovG(int n)
67     {
68         return (1llu+n)*n/2;
69     }
70     long long get_preG(int n)
71     {
72         return n;
73     }
74     gp_hash_table<int,long long>F;
75     long long get_preF(int n)
76     {
77         if(n<=MAXN)return prephi[n];
78         if(F.find(n)!=F.end())return F[n];
79         long long ans=get_preFcovG(n);

```

```

80     for(long long l=2,r;l<=n;l=r+1)
81     {
82         r=n/(n/l);
83         ans+=(get_preG(r)-get_preG(l-1))*get_preF(n/l);
84     }
85     return F[n]=ans;
86 }
87 }
88 int main()
89 {
90     sieve();
91     scanf("%d",&T);
92     while(T--)
93     {
94         scanf("%d",&n);
95         printf("%lld %lld\n",PHI::get_preF(n),MU::get_preF(n));
96     }
97     return 0;
98 }

```

## 5.12 组合数

```

1  long long C(int n,int m)
2  {
3      if(m<n-m) m=n-m;
4      long long ans=1;
5      for(int i=m+1;i<=n;i++)ans*=i;
6      for(int i=1;i<=n-m;i++)ans/=i;
7      return ans;
8  }
9  //lucas定理 组合数求模
10 long long F[100010];
11 void init(long long p) //初始化
12 {
13     F[0]=1;
14     for(int i=1;i<=p;i++)F[i]=F[i-1]*i%(1000000007);
15 }
16 long long inv(long long a,long long m)
17 {
18     if(a==1)return 1;
19     return inv(m%a,m)*(m-m/a)%m;
20 }
21 long long Lucas(long long n,long long m,long long p) //n中取m个 m、p中有一个小于1e6
22 {
23     long long ans=1;
24     while(n&& m)
25     {
26         long long a=n%p,b=m%p;
27         if(a<b)return 0;
28         ans=ans*F[a]%p*inv(F[b]*F[a-b]%p,p)%p;
29         n/=p;
30         m/=p;

```

```

31     }
32     return ans;
33 }
34 //lucas2
35 long long qpow(long long x,long long y,long long MOD)
36 {
37     long long ans=1;
38     while(y)
39     {
40         if(y&1)ans=(x*ans)%MOD;
41         x=(x*x)%MOD,y>>=1;
42     }
43     return ans;
44 }
45 long long C(long long n, long long m,long long p)
46 {
47     if(m>n) return 0;
48     long long ans=1;
49     for(int i=1;i<=m;i++)
50     {
51         long long a=(n+i-m)%p,b=i%p;
52         ans=ans*(a*qpow(b,p-2,p)%p)%p;
53     }
54     return ans;
55 }
56 long long Lucas(long long n,long long m,long long p)
57 {
58     if(m==0)return 1;
59     return C(n%p,m%p,p)*Lucas(n/p,m/p,p)%p;
60 }
61
62 //组合数逆元打表
63 //2
64 long long fac[maxn+5],inv[maxn+5];
65 long long qpow(long long a,long long b){long long r=1,t=a; while(b){if(b&1)r=(r*t)%mod;b>>=1;t=(t*t)%mod;}return r;}
66 long long C(long long n,long long m){return fac[n]*inv[m]%mod*inv[n-m]%mod;}
67 void init()
68 {
69     fac[0]=fac[1]=1;
70     for (int i=2;i<=maxn;i++)fac[i]=fac[i-1]*i%mod;
71     inv[maxn]=qpow(fac[maxn],mod-2);
72     for(int i=maxn-1;i>=0;i--)inv[i]=inv[i+1]*(i+1)%mod;
73 }

```

### 5.13 模系解码、分数间最小分子

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 void euclid(long long pa,long long pb,long long qa,long long qb,long long &x,long
   long &y)//pa/pb<x/y<qa/qb min(x)
4 {

```

```

5   long long z=(pa+pb-1)/pb;
6   if(z<=qa/qb)
7   {
8       x=z,y=1;
9       return;
10  }
11  pa-=(z-1)*pb,qa-=(z-1)*qb;
12  euclid(qb,qa,pb,pa,y,x);
13  x+=(z-1)*y;
14  }
15  void solve(long long x,long long p)
16  {
17      long long a,b,y;
18      euclid(p,x,p,x-1,b,y);
19      a=b*x-p*y;
20      printf("%lld/%lld\n",a,b);
21  }
22  int main()
23  {
24      long long x,mod;
25      while(scanf("%lld %lld",&x,&mod)!=EOF)solve(x,mod);
26  }

```

## 5.14 EXCRT

```

1  #include <stdio>
2  const int MAXN = 100010;
3  typedef long long ll;
4  int n;
5  ll a[MAXN], b[MAXN], ans, M, x, y;
6  ll exgcd(ll a, ll b, ll &x, ll &y){
7      if(!b){ x = 1; y = 0; return a; }
8      ll d = exgcd(b, a % b, x, y);
9      ll z = x; x = y; y = z - (a / b) * y;
10     return d;
11 }
12 ll Slow_Mul(ll n, ll k, ll mod){
13     ll ans = 0;
14     while(k){
15         if(k & 1) ans = (ans + n) % mod;
16         k >>= 1;
17         n = (n + n) % mod;
18     }
19     return ans;
20 }
21 //min x=a mod b
22 int main(){
23     scanf("%d", &n);
24     for(int i = 1; i <= n; ++i)
25         scanf("%lld%lld", &b[i], &a[i]);
26     ans = a[1];
27     M = b[1];

```

```

28     for(int i = 2; i <= n; ++i){
29         ll B = ((a[i] - ans) % b[i] + b[i]) % b[i];
30         ll GCD = exgcd(M, b[i], x, y);
31         x = Slow_Mul(x, B / GCD, b[i]);
32         ans += M * x;
33         M *= b[i] / GCD;
34         ans = (ans + M) % M;
35     }
36     printf("%lld\n", ans);
37     return 0;
38 }

```

### 5.15 n 次同余

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  long long quick_mod(long long a,long long b,long long mod)
4  {
5      long long ans=1;
6      while(b)
7      {
8          if(b&1)ans=ans*a%mod;
9          b>>=1;
10         a=a*a%mod;
11     }
12     return ans;
13 }
14 //快速幂
15 long long ex_gcd(long long a, long long b, long long &x, long long &y)
16 {
17     if (b == 0)
18     {
19         x = 1, y = 0;
20         return a;
21     }
22     else
23     {
24         long long r = ex_gcd(b, a % b, y, x);
25         y -= x * (a / b);
26         return r;
27     }
28 }
29 //扩展欧几里得算法
30 vector<long long>v;
31 bool g_text(long long g,long long p)
32 {
33     for(long long i=0;i<v.size();i++)if(quick_mod(g,(p-1)/v[i],p)==1)return 0;
34     return 1;
35 }
36 long long primitive_root(long long p)
37 {
38     long long tmp=p-1;

```

```

39     for(long long i=2;i<=tmp/i;i++)
40     {
41         if(tmp%i==0)
42         {
43             v.push_back(i);
44             while(tmp%i==0)tmp/=i;
45         }
46     }
47     if(tmp!=1)v.push_back(tmp);
48     long long g=1;
49     while(1)
50     {
51         if(g_text(g,p))return g;
52         ++g;
53     }
54 }
55 //求解原根
56 struct sa
57 {
58     long long x;
59     int id;
60     bool operator<(const sa &b)const
61     {
62         if (x == b.x) return id < b.id;
63         return x<b.x;
64     }
65 }rec[100500];
66 //用rec存离散对数
67 long long discerte_log(long long x,long long n,long long m)
68 {
69     int s=(int)(sqrt((double)m+0.5));
70     while((long long)s*s<=m)s++;
71     long long cur=1;
72     sa tmp;
73     for(int i=0;i<s;i++)
74     {
75         tmp.x=cur,tmp.id=i;
76         rec[i]=tmp;
77         cur=cur*x%m;
78     }
79     sort(rec,rec+s);
80     //这里不能用map查找比较慢,采用排序二分就快了
81     long long mul= quick_mod(cur, m - 2, m) % m;
82     //这里有的方法是在下面的循环里求解快速幂,但本题是不行的 要在循环外面弄,保证时间
83     cur=1;
84     for(long long i=0;i<s;i++)
85     {
86         long long more=n*cur%m;
87         tmp.x=more,tmp.id=-1;
88         int j=lower_bound(rec,rec+s,tmp)-rec;
89         if(rec[j].x==more)return i*s+rec[j].id;
90         cur=cur*mul%m;
91     }

```

```

92     return -1;
93 }
94 //求解离散对数
95 vector<long long>residue(long long p,long long n,long long a)
96 {
97     vector<long long>ret;
98     if(a==0)
99     {
100         ret.push_back(0);
101         return ret;
102     }
103     long long g=primitive_root(p),m=discerte_log(g,a,p);
104     if(m==-1)return ret;
105     long long A=n,B=p-1,C=m,x,y,G=ex_gcd(A,B,x,y);
106     if(C%G!=0)return ret;
107     x=x*(C/G)%B;
108     long long delta=B/G;
109     for(int i=0;i<G;i++)
110     {
111         x=((x+delta)%B+B)%B;
112         ret.push_back(quick_mod(g,x,p));
113     }
114     sort(ret.begin(),ret.end());
115     ret.erase(unique(ret.begin(),ret.end()),ret.end());
116     return ret;
117 }
118 //求解n次剩余  $X^A \bmod P = B$ 
119 int main()
120 {
121     int t;
122     scanf("%d",&t);
123     while(t--){
124         long long p,A,b;
125         v.clear();
126         scanf("%I64d%I64d%I64d",&p,&A,&b);
127         vector<long long>ans;
128         ans=residue(p,A,b);
129         if(ans.empty()){
130             puts("No Solution");
131         }
132         else {
133             for(unsigned int i=0;i<ans.size();i++){
134                 printf("%I64d ",ans[i]);
135             }
136             puts("");
137         }
138     }
139     return 0;
140 }

```

## 5.16 离散对数

```

1  //a^x=b mod p
2  #include<bits/stdc++.h>
3  using namespace std;
4  unordered_map<int,int>Hash;
5  #define mul(a,b,p) (1ll*(a)*(b)%p)
6  int gcd(int a,int b){return b?gcd(b,a%b):a;}
7  int exBSGS(int a,int b,int p)
8  {
9      a%=p,b%=p;
10     if(b==1) return 0;
11     if(!b&&!a) return 1;
12     if(!a) return -1;
13     if(!b)
14     {
15         int ret=0,d;
16         while((d=gcd(a,p))!=1)
17         {
18             ++ret,p/=d;
19             if(p==1) return ret;
20         }
21         return -1;
22     }
23     int ret=0,A=a,B=b,P=p,C=1,d;
24     while((d=gcd(A,P))!=1)
25     {
26         if(B%d) return -1;
27         P/=d,B/=d;
28         C=mul(C,A/d,P);
29         ++ret;
30         if(C==B) return ret;
31     }
32     Hash.clear();
33     int f=1,t=sqrt(P)+1;
34     for(int i=0;i<t;i++)
35     {
36         Hash[mul(f,B,P)]=i;
37         f=mul(f,A,P);
38     }
39     int tf=f;
40     f=mul(f,C,P);
41     for(int i=1;i<=t;i++)
42     {
43         if(Hash.find(f)!=Hash.end()) return ret+i*t-Hash[f];
44         f=mul(f,tf,P);
45     }
46     return -1;
47 }
48 int main()
49 {
50     int a,p,b;
51     while(scanf("%d %d %d",&a,&p,&b)&&(a||p||b))
52     {
53         int ans=exBSGS(a,b,p);

```



```

54     if(~ans) printf("%d\n",ans);
55     else puts("No Solution");
56 }
57 return 0;
58 }
59
60 //p-1的质因子只有2, 3
61 #include<bits/stdc++.h>
62 using namespace std;
63 int i,i0,n,m,T,ans,cnt2,cnt3;
64 long long qmul(long long x,long long y,long long mod){return (x*y-(long long)((long
double)x/mod*y)*mod+mod)%mod;}
65 long long qpow(long long a,long long b,long long mod){long long r=1,t=a; while(b){if
(b&1)r=qmul(r,t,mod);b>>=1;t=qmul(t,t,mod);}return r;}
66 bool check_root(int x,long long p)
67 {
68     if(cnt2&&qpow(x,(p-1)/2,p)==1)return 0;
69     if(cnt3&&qpow(x,(p-1)/3,p)==1)return 0;
70     return 1;
71 }
72 int find_root(long long p){for(int i=2;1;i++)if(check_root(i,p))return i;}
73 long long cal(long long x,long long r,long long p)
74 {
75     long long t=p-1,res=p-1,i,i0,i1;
76     for(i=1,i0=1,i1=r;i<=cnt2;i++,i0=i0*2%p,i1=qmul(i1,i1,p))
77     {
78         t/=2;
79         if(qpow(x,t,p)!=1)res=(res-i0)%p,x=qmul(x,i1,p);
80     }
81     for(i=1;i<=cnt3;i++,i0=i0*3%p,i1=qmul(i1,qmul(i1,i1,p),p))
82     {
83         t/=3;
84         if(qpow(x,t,p)!=1)res=(res-i0)%p,x=qmul(x,i1,p);
85         if(qpow(x,t,p)!=1)res=(res-i0)%p,x=qmul(x,i1,p);
86     }
87     return res;
88 }
89 long long exgcd(long long a,long long b,long long &x,long long &y)
90 {
91     if(b==0)
92     {
93         x=1,y=0;
94         return a;
95     }
96     long long r=exgcd(b,a%b,x,y),t=x;
97     x=y,y=t-a/b*y;
98     return r;
99 }
100 int main()
101 {
102     scanf("%d",&T);
103     while(T-->0)
104     {

```

```

105     long long a,b,p,r,A,B,x,y;
106     scanf("%lld %lld %lld",&p,&a,&b);
107     cnt2=cnt3=0;
108     long long tmp=p-1;
109     while(tmp%2==0)cnt2++,tmp/=2;
110     while(tmp%3==0)cnt3++,tmp/=3;
111     r=find_root(p);
112     A=cal(a,r,p);
113     B=cal(b,r,p);
114     long long g=exgcd(A,p-1,x,y);
115     if(B%g)printf("-1\n");
116     else
117     {
118         y=(p-1)/g,x=qmul(x,B/g,y);
119         printf("%lld\n",x);
120     }
121 }
122 return 0;
123 }

```

## 5.17 BerlekampMassey、ReedsSloane

```

1  #include <bits/stdc++.h>
2  using namespace std ;
3  typedef long long ll;
4
5  ///BM: 解决递推式.请保证模数的平方不会爆long long!!!
6  ///不用全抄,需要那一部分,就抄哪一部分.
7  using VI=vector<ll>;
8  class Linear_Seq{
9  public:
10     static const int N = 50010;///多项式系数最大值
11     ll res[N],c[N],md[N],COEF[N]**COEF是多项式系数*/,Mod;
12     vector<int> Md;
13     inline static ll gcdEx(ll a, ll b, ll&x, ll&y)
14     {
15         if(!b) {x=1;y=0;return a;}
16         ll d = gcdEx(b,a%b,y,x);
17         y -= (a/b)*x;
18         return d;
19     }
20     static ll Inv(ll a, ll Mod) {
21         ll x, y;
22         return gcdEx(a, Mod, x, y)==1?(x%Mod+Mod)%Mod:-1;
23     };
24
25     inline void mul(ll *a,ll *b,int k) {///下边的线性齐次递推用的.
26         fill(c,c+2*k,0) ;
27         for(int i(0);i<k;++i)if(a[i])for(int j(0);j<k;++j)
28             c[i+j]=(c[i+j]+a[i]*b[j])%Mod;
29         for (int i(2*k-1);i>=k;--i) if (c[i])for(size_t j(0);j<Md.size();++j)
30             c[i-k+Md[j]]=(c[i-k+Md[j]]-c[i]*md[Md[j]])%Mod;

```

```

31     copy(c,c+k,a) ;
32 }
33
34 int solve(ll n,VI A,VI B) { //线性齐次递推:A系数,B初值B[n]=A[0]*B[n-1]+...
35     ///这里可以单独用,给出递推系数和前几项代替矩阵快速幂求递推式第n项.
36     ll ans(0),cnt(0);
37     int k(A.size());
38     for(int i(0);i<k;++i) md[k-i-1]=-A[i];
39     md[k]=1 ; Md.clear() ;
40     for(int i(0);i<k;++i) {
41         res[i] = 0 ;
42         if (md[i]) Md.push_back(i);
43     }
44     res[0]=1;
45     while ((1LL<<cnt)<=n) ++ cnt;
46     for (int p(cnt);~p;-- p) {
47         mul(res,res,k);
48         if ((n>>p)&1) {
49             copy(res,res+k,res+1) ; res[0]=0;
50             for(size_t j(0);j<Md.size();++j)
51                 res[Md[j]]=(res[Md[j]]-res[k]*md[Md[j]])%Mod;
52         }
53     }
54     for(int i(0);i<k;++i) ans=(ans+res[i]*B[i])%Mod;
55     return ans+(ans<0?Mod:0);
56 }
57
58 ///1-st*****模数是质数用这里*****/
59 VI BM(VI s) {///BM算法求模数是质数的递推式子的通项公式,可以单独用
60     VI C(1,1),B(1,1);
61     int L(0),m(1),b(1);
62     for(size_t n(0);n<s.size();++n) {
63         ll d(0);
64         for(int i(0);i<=L;++i) d=(d+(ll)C[i]*s[n-i])%Mod;
65         if (!d) ++m;
66         else {
67             VI T(C);
68             ll c(Mod-d*Inv(b,Mod)%Mod);
69             while (C.size()<B.size()+m) C.push_back(0);
70             for (size_t i(0);i<B.size();++i)
71                 C[i+m]=(C[i+m]+c*B[i])%Mod;
72             if (2*L<=(int)n) {L=n+1-L; B=T; b=d; m=1;}
73             else ++m ;
74         }
75     }
76     /** //下边这样写能够输出递推式的系数.
77     printf("F[n] = ") ;
78     for(size_t i(0);i<C.size();++i) {
79         COEF[i+1] = min(C[i],Mod-C[i]) ;
80         if(i>0) {
81             if(i != 1) printf(" + ") ;
82             printf("%lld*F[n-%d]",COEF[i+1],i+1) ;
83             putchar(i+1==C.size()?'\n':' ' ) ;

```

```

84     }
85 }
86 */
87 return C;
88 }
89 ///1-ed*****模数是质数用这里*****
90
91
92
93
94
95 ///2-st*****模数非质数用这里*****
96 inline static void extend(VI &a, size_t d, ll value = 0) {
97     if (d <= a.size()) return; a.resize(d, value);
98 }
99 static ll CRT(const VI &c, const VI &m) {///中国剩余定理合并
100     int n(c.size());
101     ll M(1), ans(0);
102     for (int i = 0; i < n; ++i) M *= m[i];
103     for (int i = 0; i < n; ++i) {
104         ll x,y,tM(M / m[i]);
105         gcdEx(tM, m[i], x, y);
106         ans = (ans + tM * x * c[i] % M) % M;
107     }
108     return (ans + M) % M;
109 }
110
111 static VI ReedsSloane(const VI &s, ll Mod) {///求模数不是质数的递推式系数
112     auto L = [(const VI &a, const VI &b) {
113         int da = (a.size()>1||(a.size()== 1&&a[0]))?a.size()-1:-1000;
114         int db = (b.size()>1||(b.size()== 1&&b[0]))?b.size()-1:-1000;
115         return max(da, db + 1);
116     }];
117     auto prime_power = [&](const VI &s, ll Mod, ll p, ll e) {
118         vector<VI> a(e), b(e), an(e), bn(e), ao(e), bo(e);
119         VI t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
120         pw[0] = 1;
121         for (int i(pw[0] = 1); i <= e; ++i) pw[i] = pw[i - 1] * p;
122         for (ll i(0); i < e; ++i) {
123             a[i] = {pw[i]}; an[i] = {pw[i]};
124             b[i] = {0}; bn[i] = {s[0] * pw[i] % Mod};
125             t[i] = s[0] * pw[i] % Mod;
126             if (!t[i]) {t[i] = 1; u[i] = e;}
127             else for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i]);
128         }
129         for (size_t k(1); k < s.size(); ++k) {
130             for (int g(0); g < e; ++g) {
131                 if (L(an[g], bn[g]) > L(a[g], b[g])) {
132                     int id (e-1-u[g]);
133                     ao[g] = a[id]; bo[g] = b[id];
134                     to[g] = t[id]; uo[g] = u[id];
135                     r[g] = k - 1;
136                 }

```

```

137     }
138     a = an; b = bn;
139     for (int o(0); o < e; ++o) {
140         ll d(0);
141         for (size_t i(0); i < a[o].size() && i <= k; ++i)
142             d = (d + a[o][i] * s[k - i]) % Mod;
143         if (d == 0) {t[o] = 1; u[o] = e;}
144         else {
145             for (u[o]=0, t[o]=d; !(t[o]%p); t[o]/=p, ++u[o]);
146             int g (e-1-u[o]);
147             if (!L(a[g], b[g])) {
148                 extend(bn[o], k + 1);
149                 bn[o][k] = (bn[o][k] + d) % Mod;
150             } else {
151                 ll coef = t[o]*Inv(to[g], Mod)%Mod*pw[u[o]-uo[g]]%Mod;
152                 int m(k-r[g]);
153                 extend(an[o], ao[g].size()+m); extend(bn[o], bo[g].size()+m);
154                 auto fun = [&](vector<VI> &vn, vector<VI> &vo, bool f) {
155                     for (size_t i(0); i < vo[g].size(); ++i) {
156                         vn[o][i+m] -= coef*vo[g][i]%Mod;
157                         if (vn[o][i+m] < 0) vn[o][i+m] += Mod*(f?-1:1);
158                     }
159                     while (vn[o].size() && !vn[o].back()) vn[o].pop_back();
160                 };
161                 fun(an, ao, 1); fun(bn, bo, -1);
162             }
163         }
164     }
165 }
166 return make_pair(an[0], bn[0]);
167 };
168 vector<tuple<ll, ll, int> > fac;
169 for (ll i(2); i*i <= Mod; ++i)
170     if (!(Mod % i)) {
171         ll cnt(0), pw(1);
172         while (!(Mod % i)) {Mod /= i; ++cnt; pw *= i;}
173         fac.emplace_back(pw, i, cnt);
174     }
175 if (Mod > 1) fac.emplace_back(Mod, Mod, 1);
176 vector<VI> as;
177 size_t n = 0;
178 for (auto &&x: fac) {
179     ll Mod, p, e;
180     VI a, b;
181     std::tie(Mod, p, e) = x;
182     auto ss = s;
183     for (auto &&x: ss) x %= Mod;
184     std::tie(a, b) = prime_power(ss, Mod, p, e);
185     as.emplace_back(a);
186     n = max(n, a.size());
187 }
188 VI a(n), c(as.size()), m(as.size());
189 for (size_t i(0); i < n; ++i) {

```

```

190         for (size_t j(0); j < as.size(); ++j) {
191             m[j] = std::get<0>(fac[j]);
192             c[j] = i < as[j].size() ? as[j][i] : 0;
193         }
194         a[i] = CRT(c, m);
195     }
196     return a;
197 }
198 ///2-ed*****模数非质数用这里*****
199 ll solve(VI a,ll n,ll Mod,bool prime=true) {
200     VI c; this->Mod = Mod ;
201     if(prime) c = BM(a);///如果已经知道系数了,直接输入到c就行了,不用调用BM().
202     else c = ReedsSloane(a,Mod);
203     c.erase(c.begin()) ;
204     for(size_t i(0);i<c.size();++i) c[i] = (Mod-c[i])%Mod;
205     return solve(n,c,VI(a.begin(),a.begin()+c.size()));
206 }
207 }BMEX;
208 ///BMEX.slove(初始值vector[从0开始],要得到的项数,模数,模数是不是质数)
209 ///质数为1, 非质数为0
210 int f[2025],sum[2025];
211 ll quickpow(ll a,ll b,ll mod){
212     ll ans=1;
213     a%=mod;
214     while(b){
215         if(b&1)ans=ans*a%mod;
216         a=a*a%mod;
217         b>>=1;
218     }
219     return ans%mod;
220 }
221 int main(){
222     ll mod=1e9,n,m;
223     scanf("%lld%lld",&n,&m);
224     VI G;
225     f[0]=sum[0]=0;
226     f[1]=sum[1]=1;
227     for(int i=2;i<2020;i++){
228         f[i]=(f[i-1]+f[i-2])%mod;
229         sum[i]=(sum[i-1]+quickpow(f[i],m,mod))%mod;
230     }
231     for(int i=0;i<2020;i++)G.push_back(sum[i]);
232     printf("%lld\n",BMEX.solve(G,n,mod,0));
233     return 0;
234 }

```

### 5.18 拉格朗日插值法

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int mod=1e9+7;
4 long long qpow(long long a,long long b){long long r=1,t=a; while(b){if(b&1)r=(r*t)%

```

```

    mod;b>>=1;t=(t*t)%mod;}return r;}
5 namespace polysum
6 {
7     //先init前M项,然后计算
8     const int D=1000005;
9     long long a[D],f[D],g[D],p[D],p1[D],p2[D],b[D],h[D][2],C[D];
10    long long calcn(int d,long long *a,long long n)
11    {
12        if (n<=d) return a[n];
13        p1[0]=p2[0]=1;
14        long long ans=0;
15        for(int i=0;i<=d;i++)p1[i+1]=p1[i]*(n-i+mod)%mod,p2[i+1]=p2[i]*(n-d+i+mod)%
            mod;
16        for(int i=0;i<=d;i++)
17        {
18            long long t=g[i]*g[d-i]%mod*p1[i]%mod*p2[d-i]%mod*a[i]%mod;
19            if ((d-i)&1)ans=(ans-t+mod)%mod;
20            else ans=(ans+t)%mod;
21        }
22        return ans;
23    }
24    void init(int M)
25    {
26        f[0]=f[1]=g[0]=g[1]=1;
27        for(int i=2;i<=M+4;i++)f[i]=f[i-1]*i%mod;
28        g[M+4]=qpow(f[M+4],mod-2);
29        for(int i=M+3;i>=2;i--)g[i]=g[i+1]*(i+1)%mod;
30    }
31    long long polysum(long long n,long long *a,long long m)
32    { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]
33        a[m+1]=calcn(m,a,m+1);
34        for(int i=1;i<=m+1;i++)a[i]=(a[i-1]+a[i])%mod;
35        return calcn(m+1,a,n-1);
36    }
37    long long qpolysum(long long R,long long n,long long *a,long long m)
38    { // a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
39        if(R==1)return polysum(n,a,m);
40        a[m+1]=calcn(m,a,m+1);
41        long long r=qpow(R,mod-2),p3=0,p4=0,c,ans;
42        h[0][0]=0,h[0][1]=1;
43        for(int i=1;i<=m+1;i++)h[i][0]=(h[i-1][0]+a[i-1])*r%mod,h[i][1]=h[i-1][1]*r%
            mod;
44        for(int i=0;i<=m+1;i++)
45        {
46            long long t=g[i]*g[m+1-i]%mod;
47            if (i&1) p3=((p3-h[i][0]*t)%mod+mod)%mod,p4=((p4-h[i][1]*t)%mod+mod)%mod;
48            else p3=(p3+h[i][0]*t)%mod,p4=(p4+h[i][1]*t)%mod;
49        }
50        c=qpow(p4,mod-2)*(mod-p3)%mod;
51        for(int i=0;i<=m+1;i++)h[i][0]=(h[i][0]+h[i][1]*c)%mod,C[i]=h[i][0];
52        ans=(calcn(m,C,n)*qpow(R,n)-c)%mod;
53        if(ans<0)ans+=mod;
54        return ans;

```

```

55     }
56 }
57 long long a[1000005];
58 int main()
59 {
60     int n,k;
61     scanf("%d %d",&n,&k);
62     for(int i=0;i<=k;i++)a[i]=qpow(i,k);
63     polysum::init(k);
64     if(k==0)printf("%d\n",n);
65     else printf("%lld\n",polysum::polysum(n+1,a,k));
66     return 0;
67 }

```

### 5.19 高斯消元

```

1  bool gauss(int n)
2  {
3      long long del;
4      for(int i=1;i<=n;i++)
5      {
6          int k=i;
7          for(int j=i+1;j<=n;j++)if(a[j][i])k=j;
8          if((del=a[k][i])==0)return 0;
9          long long invdel=inv(del);
10         for(int j=i;j<=n+1;j++)swap(a[i][j],a[k][j]);
11         for(int j=i;j<=n+1;j++)a[i][j]=a[i][j]*invdel%mod;
12         for(k=1;k<=n;k++)if(k!=i)
13         {
14             del=a[k][i];
15             for(int j=i;j<=n+1;j++)a[k][j]=(a[k][j]-a[i][j]*del%mod)%mod;
16         }
17     }
18     return 1;
19 }
20
21 const int maxn = 100 + 5;
22 int n;
23 double a[maxn][maxn]; //用二维数组存系数矩阵
24
25 inline void Gauss(){ //高斯消元的主函数
26     for(int i = 1; i <= n; i++){ //选取一行作为消元系数的对象
27         for(int j = i; j <= n; j++){ //挨个系数化一
28             for(int k = n + 1; k >= i; k--)
29                 a[j][k] /= a[j][i];
30             for(int j = i + 1; j <= n; j++){ //挨个选取方程，减去系数（可能不太好理解，请自行手推
                // 弄懂高斯消元）
31                 for(int k = i; k <= n + 1; k++)
32                     a[j][k] -= a[i][k];
33             }
34             for(int i = n; i >= 1; i--) //直接在系数的那个位置乘上未知数，然后移到方程的等号右边
35                 for(int j = n; j >= i + 1; j--)

```



```

36         a[i][j] *= a[j][n + 1], a[i][n + 1] -= a[i][j];
37     }
38
39     inline bool judge(){//判断函数
40         for(int i = 1; i <= n; i++)
41             if(a[i][n + 1] != a[i][n + 1])//挨个寻找解, 如果有某个解自己不等于自己, 说明没有唯一解
42                 return false;
43         return true;
44     }

```

## 5.20 Dirichlet

```

1 void Dirichlet(long long *a, long long *b)//a*b
2 {
3     memset(tmp, 0, sizeof tmp);
4     for(int i=1; i*i<=n; ++i)
5     {
6         tmp[i*i] += a[i]*b[i]%mod, Mod(tmp[i*i]);
7         for(int j=i+1; i*j<=n; ++j)//下边加上a[i]*b[j]和a[j]*b[i], 所以j从i+1开始即可
8             (tmp[i*j] += a[i]*b[j]%mod + a[j]*b[i]%mod)%=mod;//注意这加两个数不能一步用Mod取模。。
9     }
10    memcpy(a, tmp, sizeof tmp);
11 }

```

## 5.21 类欧几里德

```

1 constexpr int mod = 998244353;
2 constexpr int inv2 = 499122177;
3 constexpr int inv6 = 166374059;
4 //f: ai+b/c 向下取整 1-n 求和
5 long long f(long long a, long long b, long long c, long long n) {
6     if (a == 0)
7         return (n + 1) * (b / c) % mod;
8     if (n == 0) return (b / c);
9     if (a >= c || b >= c)
10         return (f(a % c, b % c, c, n) + (a / c) * n % mod * (n + 1) % mod * inv2 %
11             mod + (b / c) * (n + 1) % mod) % mod;
12     long long m = (a * n + b) / c;
13     return (n * m % mod - f(c, c - b - 1, a, m - 1)) % mod;
14 }
15 //g: ai+b/c 向下取整的平方 1-n 求和
16 long long g(long long a, long long b, long long c, long long n) {
17     if (a == 0) return (b / c) * n % mod * (n + 1) % mod * inv2 % mod;
18     if (n == 0) return 0;
19     if (a >= c || b >= c) return (g(a % c, b % c, c, n) + (a / c) * n % mod * (n + 1) %
20         mod * (2 * n + 1) % mod * inv6 % mod + (b / c) * n % mod * (n + 1) %
21         mod * inv2 % mod) % mod;
22     long long m = (a * n + b) / c;

```

```

20     return (n * (n + 1) % mod * m % mod - f(c, c - b - 1, a, m - 1) - h(c, c - b -
21         1, a, m - 1)) % mod * inv2 % mod;
22 }
23 //f:i*(ai+b/c向下取整)1-n求和
24 long long h(long long a, long long b, long long c, long long n) {
25     if (a == 0) return (n + 1) * (b / c) % mod * (b / c) % mod;
26     if (n == 0) return (b / c) * (b / c) % mod;
27     if (a >= c || b >= c)
28         return ((a / c) * (a / c) % mod * n % mod * (n + 1) % mod * (2 * n + 1) % mod
29             * inv6 % mod +
30             (b / c) * (b / c) % mod * (n + 1) % mod + (a / c) * (b / c) % mod * n %
31             mod * (n + 1) % mod +
32             h(a % c, b % c, c, n) + 2 * (a / c) % mod * g(a % c, b % c, c, n) % mod
33             +
34             2 * (b / c) % mod * f(a % c, b % c, c, n) % mod) % mod;
35     long long m = (a * n + b) / c;
36     return (n * m % mod * (m + 1) % mod - 2 * g(c, c - b - 1, a, m - 1) - 2 * f(c, c
37         - b - 1, a, m - 1) - f(a, b, c, n)) % mod;
38 }

```

## 6 其他

### 6.1 01 分数规划

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define fi first
5  #define se second
6  #define mem(a, b) memset(a, b, sizeof(a))
7  #define INF 0X3f3f3f3f
8  const ll MAXN = 5000 + 7;
9  const ll MOD = 1e9 + 7;
10 //-----//
11 int n, m;
12 struct pop
13 {
14     int a, b;
15     double v;
16     bool operator<(const pop &a) const
17     {
18         return a.v > v; //a.v越小越好，所以从大到小排序
19     }
20 } num[MAXN];
21 double check(double ans)
22 {
23     double va = 0, vb = 0;
24     for (int i = 1; i <= n; i++)
25     {
26         num[i].v = num[i].a - num[i].b * ans; //num[i].v是偏移量。
27     }
28     sort(num + 1, num + 1 + n);

```

```

29     for (int i = m + 1; i <= n; i++)
30     {
31         va += num[i].a;
32         vb += num[i].b;
33     }
34     return va / vb;
35 }
36 int main()
37 {
38     while (scanf("%d %d", &n, &m) != EOF && n && m)
39     {
40         for (int i = 1; i <= n; i++)
41         {
42             scanf("%d", &num[i].a);
43         }
44         for (int i = 1; i <= n; i++)
45         {
46             scanf("%d", &num[i].b);
47         }
48         double temp = 0, ans = 1;
49         while (fabs(temp - ans) > 1e-4) //精度
50         {
51             ans = temp;
52             temp = check(temp);
53         }
54         printf("%.01f\n", ans*100);
55     }
56 }

```

## 6.2 单纯形法

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn = 500; // 变量数目上限
4  const int maxm = 1500; // 约束数目上限
5  const double INF = 1e10;
6  const double eps = 1e-6;
7  struct Simplex
8  {
9      int n; // 变量个数
10     int m; // 约束个数
11     double a[maxn+5][maxn+5]; // 输入矩阵
12     double x[maxn+5]; // 方案
13     int B[maxn+5], N[maxn+5]; // 算法辅助变量
14     bool fn[maxn+5], fm[maxn+5];
15     void pivot(int r, int c)
16     {
17         swap(N[c], B[r]);
18         a[r][c] = 1 / a[r][c];
19         if(fm[r])
20         {
21             fm[r]=0;

```

```

22     for (int j = 0; j <= n; j++)
23     {
24         if (j != c) a[r][j] *= a[r][c];
25         if(abs(a[r][j])>eps)fm[r]=1;
26     }
27     if(fn[c])
28     {
29         fn[c]=0;
30         for (int i = 0; i <= m; i++)
31         {
32             if (i != r&&abs(a[i][c])>eps)
33             {
34                 for (int j = 0; j <= n; j++)if (j != c) a[i][j] -= a[i][c] * a[r
                    ][j];
35                 a[i][c] = -a[i][c] * a[r][c];
36             }
37             if(abs(a[i][c])>eps)fn[c]=1;
38         }
39     }
40 }
41 }
42 bool feasible()
43 {
44     while(1)
45     {
46         int r, c;
47         double p = INF;
48         for (int i = 0; i < m; i++)if (a[i][n] < p) p = a[r = i][n];
49         if (p > -eps) return true;
50         p = 0;
51         for (int i = 0; i < n; i++)if (a[r][i] < p) p = a[r][c = i];
52         if (p > -eps) return false;
53         p = a[r][n] / a[r][c];
54         for (int i = r + 1; i < m; i++)
55         {
56             if (a[i][c] > eps)
57             {
58                 double v = a[i][n] / a[i][c];
59                 if (v < p) r = i, p = v;
60             }
61         }
62         pivot(r, c);
63     }
64 }
65 // 解有界返回1, 无解返回0, 无界返回-1。b[i]为x[i]的值, ret为目标函数的值
66 int simplex()
67 {
68     for (int i = 0; i < n; i++) N[i] = i,fn[i]=1;
69     for (int i = 0; i < m; i++) B[i] = n + i,fm[i]=1;
70     if (!feasible()) return 0;
71     while(1)
72     {
73         int r, c;

```

```

74     double p = 0;
75     for (int i = 0; i < n; i++) if (a[m][i] > p) p = a[m][i];
76     if (p < eps)
77     {
78         //输出方案
79         for (int i = 0; i < n; i++) if (N[i] < n) x[N[i]] = 0;
80         for (int i = 0; i < m; i++) if (B[i] < n) x[B[i]] = a[i][n];
81         //a[m][n]就是最大值 但是好像有精度误差
82         return 1;
83     }
84     p = INF;
85     for (int i = 0; i < m; i++)
86         if (a[i][c] > eps)
87         {
88             double v = a[i][n] / a[i][c];
89             if (v < p) r = i, p = v;
90         }
91     if (p == INF) return -1;
92     pivot(r, c);
93 }
94 }
95 }sp;
96 double a[maxn];
97 int main()
98 {
99     int T;
100    scanf("%d",&T);
101    while(T--)
102    {
103        int n;
104        scanf("%d",&n);
105        for(int i=0;i<n;i++)scanf("%lf",&a[i]);
106        int k,L,R;
107        scanf("%d %d %d",&k,&L,&R);
108        sp.n=n,sp.m=0;
109        for(int i=0;i<n;i++)
110        {
111            for(int i0=0;i0<n;i0++)sp.a[sp.m][i0]=(i0==i);
112            sp.a[sp.m++][n]=1;
113        }
114        for(int i=0;i+k-1<n;i++)
115        {
116            int l=i,r=i+k-1;
117            for(int i0=0;i0<n;i0++)sp.a[sp.m][i0]=(i0>=l&&i0<=r);
118            sp.a[sp.m++][n]=R;
119            for(int i0=0;i0<n;i0++)sp.a[sp.m][i0]=-(i0>=l&&i0<=r);
120            sp.a[sp.m++][n]=-L;
121        }
122        for(int i=0;i<n;i++)sp.a[sp.m][i]=a[i];
123        sp.simplex();
124        double ans=0;
125        for(int i=0;i<n;i++)if(sp.x[i]>eps)ans+=a[i];
126        printf("%.0f\n",ans);

```

```

127     for(int i=0;i<n;i++)printf("%d", (sp.x[i]>eps));
128     printf("\n");
129 }
130 return 0;
131 }

```

### 6.3 java 大数

```

1  1、新建一个值为123的大整数对象
2  BigInteger a=new BigInteger( "123" ); //第一种，参数是字符串
3  BigInteger a=BigInteger.valueOf(123); //第二种，参数可以是int、long
4  2、大整数的四则运算
5  a.add(b); //a,b均为BigInteger类型，加法
6  a.subtract(b); //减法
7  a.divide(b); //除法
8  a.multiply(b); //乘法
9
10 3、大整数比较大小
11 a.equals(b); //如果a、b相等返回true否则返回false
12 a.compareTo(); //a小于b返回-1，等于返回0，大于返回1
13
14 4、常用方法
15 a.mod(b); //求余
16 a.gcd(b); //求最大公约数
17 a.max(b); //求最大值
18 a.min(b); //求最小值
19
20 5、BigInteger中的常数
21 BigInteger.ZERO //大整数0
22 BigInteger.ONE //大整数1
23 BigInteger.TEN //大整数10
24
25 Scanner cin = new Scanner(System.in); //读入
26 while(cin.hasNext()) { //等同于!=EOF
27     BigInteger a;
28     a = cin.BigInteger(); //读入一个BigInteger;
29     System.out.println(a); //输出a并换行
30 }
31
32
33 1、A == B ? //高精度小数运算
34 题目描述：输入两个非常大的数A和B，判断A是否等于B，如果相等输出YES，否则输出NO
35 分析：这个题在hdu上实际上并没有给出范围，WA了之后才知道这是道大数题，因为仅仅涉及到输入、比
    较和输出，所以非常适合用作大数的入门题
36 注意：这里只是说给出两个数A和B，并没有说是两个整数，所以应该采用BigDecimal
37 import java.math.BigDecimal;
38 import java.util.Scanner;
39 public class Main {
40     public static void main(String[] args) { // TODO Auto-generated method stub
        BigDecimal a, b;
41         Scanner cin = new Scanner(System.in);
42         while (cin.hasNext()) {

```

```
43         a = cin.nextBigDecimal();
44         b = cin.nextBigDecimal();
45         if (a.compareTo(b) == 0) System.out.println("YES");
46         else System.out.println("NO");
47     }
48 }
49 }
50
51 BigDecimal的常用方法:
52 加: add (BigDecimal)
53 减: subtract (BigDecimal)
54 乘: multiply (BigDecimal)
55 除: divide (BigDecimal)
56 乘方: pow (int)
57 取绝对值: abs ()
58 取反: negate ()
59 对比: compareTo (BigDecimal)
60 设置小数点精确度: setScale (int)
61 设置保留小数点精确度并添加保留方式 (直接加1或者四舍五入) : setScale (int, int)
62 BigDecimal a=new BigDecimal("0.1000");
63 System.out.println(a.stripTrailingZeros().toPlainString());
64 String a = "1";//去掉后面无用的0;
65 String b = "4.56";
66 BigDecimal aBD = new BigDecimal(a);
67 BigDecimal bBD = new BigDecimal(b);
68 BigDecimal resultBD = aBD.divide(bBD).setScale(3, java.math.BigDecimal.ROUND_HALF_UP
    );
69
70 例子:
71 import java.util.*;
72 import java.math.*;
73 import java.math.BigInteger;
74 public class Main {
75     public static void main(String[] args) {
76
77         Scanner in = new Scanner(System.in);
78         while (in.hasNext()) {
79             int n=in.nextInt();
80             BigInteger ans=new BigInteger("1");
81             while(n-->0)
82             {
83                 BigInteger a = in.nextBigInteger();
84                 ans=ans.multiply(a);
85             }
86             System.out.println(ans);
87         }
88         in.close();
89     }
90 }
91
92 import java.math.BigInteger;
93 import java.util.Scanner;
94 import java.math.*;
```

```

95 public class Main {
96     public static void main(String[] args) {
97         Scanner in =new Scanner (System.in);
98         while(in.hasNext()) {
99             int t=in.nextInt();
100             while(t-->0) {
101                 BigInteger m=in.nextBigInteger();
102                 BigInteger n=in.nextBigInteger();
103                 BigInteger n2=new BigInteger("1");
104                 BigInteger ans=new BigInteger("1");
105                 for(BigInteger i=n2;; i=i.add(n2)) {
106                     if(i.compareTo(n)==0)
107                         break;
108                     ans=ans.multiply(m);
109                 }
110                 BigInteger temp=ans.gcd(n);
111                 BigInteger up=n.divide(temp);
112                 BigInteger down=ans.divide(temp);
113                 System.out.println(up+"/"+down);
114             }
115         }
116     }
117 }

```

## 6.4 mini 读入挂

```

1 namespace FastIO {
2     inline int read() {
3         char ch = getchar(); int r = 0, w = 1;
4         while(!isdigit(ch)) {if(ch == '-') w = -1; ch = getchar();}
5         while(isdigit(ch)) {r = r * 10 + ch - '0', ch = getchar();}
6         return r * w;
7     }
8     void _write(int x) {
9         if(x < 0) putchar('-'), x = -x;
10        if(x > 9) _write(x / 10);
11        putchar(x % 10 + '0');
12    }
13    inline void write(int x) {
14        _write(x);
15        puts("");
16    }
17 }

```

## 6.5 FFT 快速傅里叶变换

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int N=1<<18;
4 struct E
5 {

```



```

6   double a,b;
7   E(double A=0,double B=0){a=A;b=B;}
8   E operator + (E y){return E(a+y.a,b+y.b);}
9   E operator - (E y){return E(a-y.a,b-y.b);}
10  E operator * (E y){return E(a*y.a-b*y.b,a*y.b+b*y.a);}
11 }w[N],a[N];
12 void fft(E *a,int n,int tp)
13 {
14     for(int i=1,j=0; i<n; i++)
15     {
16         for(int k=(n>>1);!((j^=k)&k); k>>=1);
17         if(i<j)swap(a[i],a[j]);
18     }
19     for(int j=2;j<=n;j<<=1)
20     {
21         w[0]=1,w[1]=E(cos(2*acos(-1)/j),tp*sin(2*acos(-1)/j));
22         int m=(j>>1)-1;
23         for(int i=2;i<=m;i++)w[i]=w[i-1]*w[1];
24         for(int i=0; i<n; i+=j)
25         {
26             for(int k=0;k<=m;k++)
27             {
28                 E x=a[i+k+(j>>1)]*w[k];
29                 a[i+k+(j>>1)]=a[i+k]-x;
30                 a[i+k]=a[i+k]+x;
31             }
32         }
33     }
34 }
35 int x,n,m,i;
36 int main()
37 {
38     scanf("%d %d",&n,&m);
39     for(i=0;i<=n;i++)scanf("%d",&x),a[i].a=x;
40     for(i=0;i<=m;i++)scanf("%d",&x),a[i].b=x;
41     int r=1;
42     while(r<=n+m)r*=2;
43     fft(a,r,1);
44     for(i=0;i<=r;i++)a[i]=a[i]*a[i];
45     fft(a,r,-1);
46     for(i=0;i<=n+m;i++)printf("%d ",int(a[i].b/(r*2)+0.5));
47     return 0;
48 }

```

## 6.6 FWT 快速沃尔什变换

```

1  const ll mod = 1e9+7;
2  const int maxn = 6e5+10;
3  int a[maxn],b[maxn];
4  int sum;
5  ll rev = 5e8+4;
6  void FWT(int a[],int n)

```

```

7 {
8     for(int d=1;d<n;d<=1)
9         for(int m=d<<1,i=0;i<n;i+=m)
10            for(int j=0;j<d;j++)
11                {
12                    int x=a[i+j],y=a[i+j+d];
13                    a[i+j]=(x+y)%mod,a[i+j+d]=(x-y+mod)%mod;
14                    //xor:a[i+j]=x+y,a[i+j+d]=x-y;
15                    //and:a[i+j]=x+y;
16                    //or:a[i+j+d]=x+y;
17                }
18 }
19
20 void UFWT(int a[],int n)
21 {
22     for(int d=1;d<n;d<=1)
23         for(int m=d<<1,i=0;i<n;i+=m)
24            for(int j=0;j<d;j++)
25                {
26                    int x=a[i+j],y=a[i+j+d];
27                    a[i+j]=1LL*(x+y)*rev%mod,a[i+j+d]=(1LL*(x-y)*rev%mod+mod)%mod;
28                    //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
29                    //and:a[i+j]=x-y;
30                    //or:a[i+j+d]=y-x;
31                }
32 }

```

## 6.7 NTT 快速数论变换

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 set<int>s1,s2;
4 const int MOD = 998244353;
5 struct mod_int
6 {
7     int val;
8
9     mod_int(long long v = 0)
10    {
11        if (v < 0)
12            v = v % MOD + MOD;
13
14        if (v >= MOD)
15            v %= MOD;
16
17        val = v;
18    }
19
20    static int mod_inv(int a, int m = MOD)
21    {
22        // https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm#Example
23        int g = m, r = a, x = 0, y = 1;

```

```

24
25     while (r != 0)
26     {
27         int q = g / r;
28         g %= r;
29         swap(g, r);
30         x -= q * y;
31         swap(x, y);
32     }
33
34     return x < 0 ? x + m : x;
35 }
36
37 explicit operator int() const
38 {
39     return val;
40 }
41
42 mod_int &operator+=(const mod_int &other)
43 {
44     val += other.val;
45     if (val >= MOD)
46         val -= MOD;
47     return *this;
48 }
49
50 mod_int &operator--(const mod_int &other)
51 {
52     val -= other.val;
53     if (val < 0)
54         val += MOD;
55     return *this;
56 }
57
58 static unsigned fast_mod(uint64_t x, unsigned m = MOD)
59 {
60     return x % m;
61     // Optimized mod for Codeforces 32-bit machines.
62     // x must be less than 2^32 * m for this to work, so that x / m fits in a 32-
63     // bit integer.
64     unsigned x_high = x >> 32, x_low = (unsigned)x;
65     unsigned quot, rem;
66     asm("divl %4\n"
67         : "=a"(quot), "=d"(rem)
68         : "d"(x_high), "a"(x_low), "r"(m));
69     return rem;
70 }
71
72 mod_int &operator*=(const mod_int &other)
73 {
74     val = fast_mod((uint64_t)val * other.val);
75     return *this;
76 }

```

```

76
77     mod_int &operator/=(const mod_int &other)
78     {
79         return *this *= other.inv();
80     }
81
82     friend mod_int operator+(const mod_int &a, const mod_int &b) { return mod_int(a)
      += b; }
83     friend mod_int operator-(const mod_int &a, const mod_int &b) { return mod_int(a)
      -= b; }
84     friend mod_int operator*(const mod_int &a, const mod_int &b) { return mod_int(a)
      *= b; }
85     friend mod_int operator/(const mod_int &a, const mod_int &b) { return mod_int(a)
      /= b; }
86
87     mod_int &operator++()
88     {
89         val = val == MOD - 1 ? 0 : val + 1;
90         return *this;
91     }
92
93     mod_int &operator--()
94     {
95         val = val == 0 ? MOD - 1 : val - 1;
96         return *this;
97     }
98
99     mod_int operator++(int)
100    {
101        mod_int before = *this;
102        ++*this;
103        return before;
104    }
105    mod_int operator--(int)
106    {
107        mod_int before = *this;
108        --*this;
109        return before;
110    }
111
112    mod_int operator-() const
113    {
114        return val == 0 ? 0 : MOD - val;
115    }
116
117    bool operator==(const mod_int &other) const { return val == other.val; }
118    bool operator!=(const mod_int &other) const { return val != other.val; }
119
120    mod_int inv() const
121    {
122        return mod_inv(val);
123    }
124

```

```

125     mod_int pow(long long p) const
126     {
127         assert(p >= 0);
128         mod_int a = *this, result = 1;
129
130         while (p > 0)
131         {
132             if (p & 1)
133                 result *= a;
134
135             a *= a;
136             p >>= 1;
137         }
138
139         return result;
140     }
141
142     friend ostream &operator<<(ostream &stream, const mod_int &m)
143     {
144         return stream << m.val;
145     }
146 };
147
148 namespace NTT
149 {
150     vector<mod_int> roots = {0, 1};
151     vector<int> bit_reverse;
152     int max_size = -1;
153     mod_int root;
154
155     bool is_power_of_two(int n)
156     {
157         return (n & (n - 1)) == 0;
158     }
159
160     int round_up_power_two(int n)
161     {
162         assert(n > 0);
163
164         while (n & (n - 1))
165             n = (n | (n - 1)) + 1;
166
167         return n;
168     }
169
170     // Given n (a power of two), finds k such that n == 1 << k.
171     int get_length(int n)
172     {
173         assert(is_power_of_two(n));
174         return __builtin_ctz(n);
175     }
176
177     // Rearranges the indices to be sorted by lowest bit first, then second lowest,

```

```

    etc., rather than highest bit first.
178 // This makes even-odd div-conquer much easier.
179 void bit_reorder(int n, vector<mod_int> &values)
180 {
181     if ((int)bit_reverse.size() != n)
182     {
183         bit_reverse.assign(n, 0);
184         int length = get_length(n);
185
186         for (int i = 0; i < n; i++)
187             bit_reverse[i] = (bit_reverse[i >> 1] >> 1) + ((i & 1) << (length - 1))
188             ;
189
190         for (int i = 0; i < n; i++)
191             if (i < bit_reverse[i])
192                 swap(values[i], values[bit_reverse[i]]);
193     }
194
195 void find_root()
196 {
197     int order = MOD - 1;
198     max_size = 1;
199
200     while (order % 2 == 0)
201     {
202         order /= 2;
203         max_size *= 2;
204     }
205
206     root = 2;
207
208     // Find a max_size-th primitive root of MOD.
209     while (!(root.pow(max_size) == 1 && root.pow(max_size / 2) != 1))
210         root++;
211 }
212
213 void prepare_roots(int n)
214 {
215     if (max_size < 0)
216         find_root();
217
218     assert(n <= max_size);
219
220     if ((int)roots.size() >= n)
221         return;
222
223     int length = get_length(roots.size());
224     roots.resize(n);
225
226     // The roots array is set up such that for a given power of two n >= 2, roots
227     // [n / 2] through roots[n - 1] are
228     // the first half of the n-th primitive roots of MOD.

```

```

228     while (1 << length < n)
229     {
230         // z is a 2^(length + 1)-th primitive root of MOD.
231         mod_int z = root.pow(max_size >> (length + 1));
232
233         for (int i = 1 << (length - 1); i < 1 << length; i++)
234         {
235             roots[2 * i] = roots[i];
236             roots[2 * i + 1] = roots[i] * z;
237         }
238
239         length++;
240     }
241 }
242
243 void fft_iterative(int N, vector<mod_int> &values)
244 {
245     assert(is_power_of_two(N));
246     prepare_roots(N);
247     bit_reorder(N, values);
248
249     for (int n = 1; n < N; n *= 2)
250         for (int start = 0; start < N; start += 2 * n)
251             for (int i = 0; i < n; i++)
252             {
253                 mod_int even = values[start + i];
254                 mod_int odd = values[start + n + i] * roots[n + i];
255                 values[start + n + i] = even - odd;
256                 values[start + i] = even + odd;
257             }
258 }
259
260 const int FFT_CUTOFF = 150;
261
262 vector<mod_int> mod_multiply(vector<mod_int> left, vector<mod_int> right)
263 {
264     int n = left.size();
265     int m = right.size();
266
267     // Brute force when either n or m is small enough.
268     if (min(n, m) < FFT_CUTOFF)
269     {
270         const uint64_t ULL_BOUND = numeric_limits<uint64_t>::max() - (uint64_t)MOD
                * MOD;
271         vector<uint64_t> result(n + m - 1);
272
273         for (int i = 0; i < n; i++)
274             for (int j = 0; j < m; j++)
275             {
276                 result[i + j] += (uint64_t)((int)left[i]) * ((int)right[j]);
277
278                 if (result[i + j] > ULL_BOUND)
279                     result[i + j] %= MOD;

```

```

280         }
281
282         for (uint64_t &x : result)
283             if (x >= MOD)
284                 x %= MOD;
285
286         return vector<mod_int>(result.begin(), result.end());
287     }
288
289     int N = round_up_power_two(n + m - 1);
290     left.resize(N);
291     right.resize(N);
292
293     bool equal = left == right;
294     fft_iterative(N, left);
295
296     if (equal)
297         right = left;
298     else
299         fft_iterative(N, right);
300
301     mod_int inv_N = mod_int(N).inv();
302
303     for (int i = 0; i < N; i++)
304         left[i] *= right[i] * inv_N;
305
306     reverse(left.begin() + 1, left.end());
307     fft_iterative(N, left);
308     left.resize(n + m - 1);
309     return left;
310 }
311
312 vector<mod_int> mod_power(const vector<mod_int> &v, int exponent)
313 {
314     assert(exponent >= 0);
315     vector<mod_int> result = {1};
316
317     if (exponent == 0)
318         return result;
319
320     for (int k = 31 - __builtin_clz(exponent); k >= 0; k--)
321     {
322         result = mod_multiply(result, result);
323
324         if (exponent >> k & 1)
325             result = mod_multiply(result, v);
326     }
327
328     return result;
329 }
330 } // namespace NTT
331 int n, m, a[300005];
332 vector<mod_int> v;

```



```

333 map<int, long long> mp;
334 int main()
335 {
336     int n, k;
337     scanf("%d %d", &n, &k);
338     for(int i=1; i<=n; i++)
339     {
340         scanf("%d", &a[i]);
341     }
342     while(k--)
343     {
344         int x;
345         scanf("%d", &x);
346
347         s1.clear(), s2.clear();
348         for(int i=1; i<=n; i++)
349         {
350             if(a[i]<x)
351             {
352                 if(s1.count(a[i])) s2.insert(a[i]);
353                 else s1.insert(a[i]);
354             }
355         }
356         vector<mod_int> v = NTT::mod_multiply(NTT::mod_power({1, 2}, s1.size()-s2.size()),
357             NTT::mod_power({1, 2, 1}, s2.size()));
358         for(int i=0; i<v.size(); i++)
359         {
360             mp[(x+1+i)*2] += v[i].val;
361             mp[(x+1+i)*2] %= MOD;
362         }
363     }
364     int q;
365     scanf("%d", &q);
366     while(q--)
367     {
368         int x;
369         scanf("%d", &x);
370         printf("%11d\n", mp[x]);
371     }
372     return 0;
373 }

```

## 6.8 MTT 快速数论变换 (任意模数)

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <cstring>
4 int mod;
5 namespace Math {
6     inline int pw(int base, int p, const int mod) {
7         static int res;

```

```

8      for (res = 1; p; p >>= 1, base = static_cast<long long> (base) * base % mod)
9          if (p & 1) res = static_cast<long long> (res) * base % mod;
10     return res;
11 }
12 inline int inv(int x, const int mod) { return pw(x, mod - 2, mod); }
13 }
14 const int mod1 = 998244353, mod2 = 1004535809, mod3 = 469762049, G = 3;
15 const long long mod_1_2 = static_cast<long long> (mod1) * mod2;
16 const int inv_1 = Math::inv(mod1, mod2), inv_2 = Math::inv(mod_1_2 % mod3, mod3);
17 struct Int {
18     int A, B, C;
19     explicit inline Int() { }
20     explicit inline Int(int __num) : A(__num), B(__num), C(__num) { }
21     explicit inline Int(int __A, int __B, int __C) : A(__A), B(__B), C(__C) { }
22     static inline Int reduce(const Int &x) {
23         return Int(x.A + (x.A >> 31 & mod1), x.B + (x.B >> 31 & mod2), x.C + (x.C >>
24             31 & mod3));
25     }
26     inline friend Int operator + (const Int &lhs, const Int &rhs) {
27         return reduce(Int(lhs.A + rhs.A - mod1, lhs.B + rhs.B - mod2, lhs.C + rhs.C -
28             mod3));
29     }
30     inline friend Int operator - (const Int &lhs, const Int &rhs) {
31         return reduce(Int(lhs.A - rhs.A, lhs.B - rhs.B, lhs.C - rhs.C));
32     }
33     inline friend Int operator * (const Int &lhs, const Int &rhs) {
34         return Int(static_cast<long long> (lhs.A) * rhs.A % mod1, static_cast<long
35             long> (lhs.B) * rhs.B % mod2, static_cast<long long> (lhs.C) * rhs.C %
36             mod3);
37     }
38     inline int get() {
39         long long x = static_cast<long long> (B - A + mod2) % mod2 * inv_1 % mod2 *
40             mod1 + A;
41         return (static_cast<long long> (C - x % mod3 + mod3) % mod3 * inv_2 % mod3 *
42             (mod_1_2 % mod) % mod + x) % mod;
43     }
44 } ;
45 #define maxn 131072
46 namespace Poly {
47 #define N (maxn << 1)
48     int lim, s, rev[N];
49     Int Wn[N | 1];
50     inline void init(int n) {
51         s = -1, lim = 1; while (lim < n) lim <= 1, ++s;
52         for (register int i = 1; i < lim; ++i) rev[i] = rev[i >> 1] >> 1 | (i & 1) <<
53             s;
54         const Int t(Math::pw(G, (mod1 - 1) / lim, mod1), Math::pw(G, (mod2 - 1) / lim
55             , mod2), Math::pw(G, (mod3 - 1) / lim, mod3));
56         *Wn = Int(1); for (register Int *i = Wn; i != Wn + lim; ++i) *(i + 1) = *i *
57             t;

```

```

51     }
52     inline void NTT(Int *A, const int op = 1) {
53         for (register int i = 1; i < lim; ++i) if (i < rev[i]) std::swap(A[i], A[rev[
54             i]]);
55         for (register int mid = 1; mid < lim; mid <= 1) {
56             const int t = lim / mid >> 1;
57             for (register int i = 0; i < lim; i += mid << 1) {
58                 for (register int j = 0; j < mid; ++j) {
59                     const Int W = op ? Wn[t * j] : Wn[lim - t * j];
60                     const Int X = A[i + j], Y = A[i + j + mid] * W;
61                     A[i + j] = X + Y, A[i + j + mid] = X - Y;
62                 }
63             }
64             if (!op) {
65                 const Int ilim(Math::inv(lim, mod1), Math::inv(lim, mod2), Math::inv(lim,
66                     mod3));
67                 for (register Int *i = A; i != A + lim; ++i) *i = (*i) * ilim;
68             }
69         }
70     }
71     #undef N
72     int n, m;
73     Int A[maxn << 1], B[maxn << 1];
74     int main() {
75         scanf("%d%d", &n, &m, &mod); ++n, ++m;
76         for (int i = 0; i < n; ++i) scanf("%d", &x), A[i] = Int(x % mod);
77         for (int i = 0; i < m; ++i) scanf("%d", &x), B[i] = Int(x % mod);
78         Poly::init(n + m);
79         Poly::NTT(A), Poly::NTT(B);
80         for (int i = 0; i < Poly::lim; ++i) A[i] = A[i]^a * B[i]^b;
81         Poly::NTT(A, 0);
82         for (int i = 0; i < n + m - 1; ++i) {
83             printf("%d", A[i].get());
84             putchar(i == n + m - 2 ? '\n' : ' ');
85         }
86         return 0;
87     }

```

## 6.9 SG 函数

```

1  const int MN= 10000;
2  int data[MN],SG[MN],mid[MN];
3  void SG(int n)
4  {
5      int i,j;
6      for(i=1;i<=n;i++)
7      {
8          memset(mid,0,sizeof(mid));
9          for(j=1;data[j]<=i;j++) mid[SG[i-data[j]]]=1;
10         for(j=0;j<=n;j++)

```

```

11         if(mid[j]==0)
12         {
13             SG[i]=j;
14             break;
15         }
16     }
17 }
18
19 SJ定理
20 (1)游戏的SG函数不为0且游戏中某个单一游戏的SG函数大于 1; (2)游戏的SG函数为0且游戏中没有单一
    游戏的SG函数大于1。

```

## 6.10 第二类斯特林数

```

1  #include<bits/stdc++.h>
2  #define LL long long
3  using namespace std;
4  const int N=4e5+10;
5  const LL P=998244353,yg=3;
6  LL n,fac[N],inv[N],f[N],g[N],S2[N];
7  LL bin[N];
8
9  LL power(LL x,LL p) {
10     LL ret=1;
11     for (;p>=1) {
12         if (p&1) ret=(ret*x)%P;
13         x=(x*x)%P;
14     }
15     return ret;
16 }
17
18 void NTT(LL *a,LL n,LL op) { //NTT:系数a数组,长度为n, op=1求值op=-1插值
19     for(LL i=0;i<n;i++) bin[i]=(bin[i]>>1)|((i&1)*(n>>1));
20     for(LL i=0;i<n;i++) if(i<bin[i]) swap(a[i],a[bin[i]]);
21     for(LL i=1;i<n;i<=1) {
22         LL wn=power(yg,op==1?(P-1)/(2*i):(P-1)-(P-1)/(2*i)),w,t;
23         for(LL j=0;j<n;j+=i<=1) {
24             w=1;
25             for(LL k=0;k<i;k++) {
26                 t=a[i+j+k]*w%P;w=w*wn%P;
27                 a[i+j+k]=(a[j+k]-t+P)%P;a[j+k]=(a[j+k]+t)%P;
28             }
29         }
30     }
31     if(op==-1) {
32         LL Inv=power(n,P-2);
33         for(LL i=0;i<n;i++) a[i]=a[i]*Inv%P;
34     }
35 }
36 long long C(long long n,long long m){return fac[n]*inv[m]%P*inv[n-m]%P;}
37 int main()
38 {

```

```

39     cin>>n;
40     if(n==1)
41     {
42         printf("1\n");
43         return 0;
44     }
45     if(n==2)
46     {
47         printf("2\n");
48         return 0;
49     }
50     fac[0]=inv[0]=1;
51     for (int i=1;i<=n;i++) fac[i]=fac[i-1]*i%P,inv[i]=power(fac[i],P-2);
52     for (int i=0;i<=n;i++) f[i]=(power(-1,i)+P)%P*inv[i]%P;
53     for (int i=0;i<=n;i++) g[i]=power(i,n-2)*inv[i]%P;
54
55     LL N=n-1;
56     LL len=1;while(len<(n+1)<<1) len<<=1;
57
58     NTT(f,len,1); NTT(g,len,1);
59     for (int i=0;i<len;i++) S2[i]=(f[i]*g[i])%P; //求f.g的卷积为S2
60     NTT(S2,len,-1);
61
62     long long ans=0;
63     for(int i=1;i<=n-2;i++)
64     {
65         long long d=fac[i]*S2[i];
66         d%=P;
67         //printf("%d %lld %lld\n",n-i,d,S2[i]);
68         ans+=d*C(n,i)%P*(n-i)%P;
69         ans%=P;
70     }
71     ans%=P,ans+=P,ans%=P;
72     printf("%lld\n",ans);
73     return 0;
74 }

```

## 6.11 离散化

```

1 struct Discretization
2 {
3     vector<int>v;
4     void init(){v.clear();}
5     void insert(int x){v.push_back(x);}
6     void sortV(){sort(v.begin(),v.end()),v.erase(unique(v.begin(),v.end()),v.end())
7         ;}
8     int getPos(int x){return lower_bound(v.begin(),v.end(),x)-v.begin()+1;}
9     int getVal(int pos){return v[pos-1];}
10    int getSize(){return v.size();}
11 }D;

```

## 6.12 STL

```

1  int __builtin_ffs (unsigned int x)
2  返回x的最后一位1的是从后向前第几位，比如7368 (1110011001000) 返回4。
3  int __builtin_clz (unsigned int x)
4  返回前导的0的个数。
5  int __builtin_ctz (unsigned int x)
6  返回后面的0的个数，和__builtin_clz相对。
7  int __builtin_popcount (unsigned int x)
8  返回二进制表示中1的个数。
9  int __builtin_parity (unsigned int x)
10 返回x的奇偶校验位，也就是x的1的个数模2的结果。
11
12 此外，这些函数都有相应的unsigned long和unsigned long long版本，只需要在函数名后面加上l或ll
    就可以了，比如int __builtin_clzll。
13
14  mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());//随机数
15  shuffle(a, a + n, rng);
16
17  struct cmp
18  {
19      bool operator()(int a,int b)
20      {
21          return dvv[a]<dvv[b];
22      }
23  }
24  priority_queue<int,vector<int>,cmp>q;
25
26  multiset<int,greater<int>>sa;
27
28  ios::sync_with_stdio(false);
29
30  //读入一行
31  getline(cin,s);
32
33  #include<ext/pb_ds/assoc_container.hpp>
34  #include<ext/pb_ds/hash_policy.hpp>//用hash
35  using namespace __gnu_pbds;
36  cc_hash_table<long long,int>dp;
37  gp_hash_table<long long,int>dp;
38
39  struct pair_hash{inline size_t operator()(const pair<int,int> & p) const {return p.
    first*1007 + p.second;}};
40
41  #include<ext/pb_ds/assoc_container.hpp>
42  #include<ext/pb_ds/tree_policy.hpp>
43  using namespace __gnu_pbds;
44  tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update>s;
45  s.order_of_key(n)
46
47  struct custom_hash {
48      static uint64_t splitmix64(uint64_t x) {
49          // http://xorshift.di.unimi.it/splitmix64.c

```

```

50     x += 0x9e3779b97f4a7c15;
51     x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
52     x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
53     return x ^ (x >> 31);
54 }
55
56 size_t operator()(uint64_t x) const {
57     static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().
        time_since_epoch().count();
58     return splitmix64(x + FIXED_RANDOM);
59 }
60 };
61
62 unordered_map<long long, int, custom_hash> safe_map;
63 gp_hash_table<long long, int, custom_hash> safe_hash_table;

```

### 6.13 星期

```

1  const int md
    [2][13]={ {0,31,28,31,30,31,30,31,31,30,31,30,31}, {0,31,29,31,30,31,30,31,31,30,31,31,30,31}};
2  struct Calendar
3  {
4      int y,m,d;
5      bool isLeap(){return (y%100!=0&& y%4==0 || y%400==0);}
6      bool isCalendar(){return !(y<1600 || y>9999 || m>12 || m<1 || d<1 || d>md[isLeap()][m]);}
7      int getWeek(){return (d+1+2*(m+(m<3)*12)+3*(m+(m<3)*12+1)/5+y-(m<3)+(y-(m<3))/4-(y-(m<3))/100+(y-(m<3))/400)%7;}//0为周日
8      int getDist(Calendar x)
9      {
10         int res=(y-1)*365+(y-1)/4-(y-1)/100+y/400+d-((x.y-1)*365+(x.y-1)/4-(x.y-1)/100+(x.y-1)/400+x.d);
11         for(int i=1;i<m;i++)res+=md[isLeap()][i];
12         for(int i=1;i<x.m;i++)res-=md[x.isLeap()][i];
13         return res;
14     }
15 };

```

### 6.14 博弈

```

1  一、巴什博弈
2
3      1、问题模型：只有一堆n个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取m
        个，最后取光者得胜。
4
5      2、解决思路：当n=m+1时，由于一次最多只能取m个，所以无论先取者拿走多少个，后取者都能够一
        次拿走剩余的物品，后者取胜，所以当一方面面对的局势是n%(m+1)=0时，其面临的是必败的局
        势。所以当n= (m+1)*r+s, (r为任意自然数, s≤m)时,如果先取者要拿走s个物品，如果后取
        者拿走x (≤m)个，那么先取者再拿走m+1-k个，结果剩下 (m+1) (r-1) 个，以后保持这样的取
        法，那么先取者肯定获胜。总之，要保持给对手留下 (m+1) 的倍数，就能最后获胜。
6

```

- 7 3、变形：条件不变，改为最后取光的人输。
- 8
- 9 结论：当  $(n-1) \% (m+1) == 0$  时后手胜利。
- 10
- 11 二、威佐夫博弈
- 12
- 13 1、问题模型：有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。
- 14
- 15 2、解决思路：A：设  $(a_i, b_i)$  ( $a_i \leq b_i, i=0, 1, 2, \dots, n$ ) 表示两堆物品的数量并称其为局势，如果甲面对  $(0, 0)$ ，那么甲已经输了，这种局势我们称为奇异局势。前几个奇异局势是： $(0, 0)$ 、 $(1, 2)$ 、 $(3, 5)$ 、 $(4, 7)$ 、 $(6, 10)$ 、 $(8, 13)$ 、 $(9, 15)$ 、 $(11, 18)$ 、 $(12, 20)$ 。任给一个局势  $(a, b)$ ，如下公式判断它是不是奇异局势： $a_k = [k(1+\sqrt{5})/2]$ ,  $b_k = a_k + k$  ( $k=0, 1, 2, \dots, n$  方括号表示取整函数)。(证明见百度百科)
- 16 三、Fibonacci 博弈
- 17
- 18 1、问题模型：
- 19
- 20 有一堆个数为  $n$  的石子，游戏双方轮流取石子，满足：
- 21
- 22 (1) 先手不能在第一次把所有的石子取完；
- 23
- 24 (2) 之后每次可以取的石子数介于 1 到对手刚取的石子数的 2 倍之间（包含 1 和对手刚取的石子数的 2 倍）。约定取走最后一个石子的人为赢家。
- 25
- 26 2、解决思路：
- 27
- 28 当  $n$  为 Fibonacci 数时，先手必败。即存在先手的必败态当且仅当石头个数为 Fibonacci 数。
- 29 四、尼姆博弈
- 30
- 31 1、问题模型：有三堆各若干个物品，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限，最后取光者得胜。
- 32
- 33 2、解决思路：用  $(a, b, c)$  表示某种局势，显证  $(0, 0, 0)$  是第一种奇异局势，无论谁面对奇异局势，都必然失败。第二种奇异局势是  $(0, n, n)$ ，只要与对手拿走一样多的物品，最后都将导致  $(0, 0, 0)$ 。
- 34
- 35 搞定这个问题需要把必败态的规律找出： $(a, b, c)$  是必败态等价于  $a \oplus b \oplus c = 0$  ( $\oplus$  表示异或运算)。
- 36 3、推广一：如果我们面对的是一个非奇异局势  $(a, b, c)$ ，要如何变为奇异局势呢？假设  $a < b < c$ ，我们只要将  $c$  变为  $a \oplus b$ ，即可，因为有如下的运算结果： $a \oplus b \oplus (a \oplus b) = (a \oplus a) \oplus (b \oplus b) = 0 \oplus 0 = 0$ 。要将  $c$  变为  $a \oplus b$ ，只从  $c$  中减去  $c - (a \oplus b)$
- 37
- 38 4、推广二：当石子堆数为  $n$  堆时，则推广为当对每堆的数目进行亦或之后值为零是必败态。

## 7 几何

### 7.1 二维几何

```
1 // `计算几何模板`
2 const double eps = 1e-8;
3 const double inf = 1e20;
4 const double pi = acos(-1.0);
```



```
5  const int maxp = 1010;
6  //判断小数和0是否相等`
7  int sgn(double x){
8      if(fabs(x) < eps)return 0;
9      if(x < 0)return -1;
10     else return 1;
11 }
12 //浮点数的平方
13 inline double sqr(double x){return x*x;}
14 struct Point{
15     double x,y;
16     Point(){ }
17     Point(double _x,double _y){
18         x = _x;
19         y = _y;
20     }
21     void input(){
22         scanf("%lf%lf",&x,&y);
23     }
24     void output(){
25         printf("%.2f %.2f\n",x,y);
26     }
27     bool operator == (Point b)const{
28         return sgn(x-b.x) == 0 && sgn(y-b.y) == 0;
29     }
30     bool operator < (Point b)const{
31         return sgn(x-b.x)== 0?sgn(y-b.y)<0:x<b.x;
32     }
33     Point operator -(const Point &b)const{
34         return Point(x-b.x,y-b.y);
35     }
36     //叉积
37     double operator ^(const Point &b)const{
38         return x*b.y - y*b.x;
39     }
40     //点积
41     double operator *(const Point &b)const{
42         return x*b.x + y*b.y;
43     }
44     //返回长度
45     double len(){
46         return hypot(x,y); //库函数
47     }
48     //返回长度的平方
49     double len2(){
50         return x*x + y*y;
51     }
52     //返回两点的距离
53     double distance(Point p){
54         return hypot(x-p.x,y-p.y);
55     }
56     Point operator +(const Point &b)const{
57         return Point(x+b.x,y+b.y);
```

```

58     }
59     Point operator *(const double &k)const{
60         return Point(x*k,y*k);
61     }
62     Point operator /(const double &k)const{
63         return Point(x/k,y/k);
64     }
65     ///计算pa 和 pb 的夹角`
66     ///就是求这个点看a,b 所成的夹角`
67     ///测试 LightOJ1203`
68     double rad(Point a,Point b){
69         Point p = *this;
70         return fabs(atan2( fabs((a-p)^(b-p)),(a-p)*(b-p) ));
71     }
72     ///化为长度为r的向量`
73     Point trunc(double r){
74         double l = len();
75         if(!sgn(l))return *this;
76         r /= l;
77         return Point(x*r,y*r);
78     }
79     ///逆时针旋转90度`
80     Point rotleft(){
81         return Point(-y,x);
82     }
83     ///顺时针旋转90度`
84     Point rotright(){
85         return Point(y,-x);
86     }
87     ///绕着p点逆时针旋转angle`
88     Point rotate(Point p,double angle){
89         Point v = (*this) - p;
90         double c = cos(angle), s = sin(angle);
91         return Point(p.x + v.x*c - v.y*s,p.y + v.x*s + v.y*c);
92     }
93 };
94 struct Line{
95     Point s,e;
96     Line(){}
97     Line(Point _s,Point _e){
98         s = _s;
99         e = _e;
100     }
101     bool operator ==(Line v){
102         return (s == v.s)&&(e == v.e);
103     }
104     ///根据一个点和倾斜角angle确定直线,0<=angle<pi`
105     Line(Point p,double angle){
106         s = p;
107         if(sgn(angle-pi/2) == 0){
108             e = (s + Point(0,1));
109         }
110         else{

```

```

111         e = (s + Point(1,tan(angle)));
112     }
113 }
114 //ax+by+c=0
115 Line(double a,double b,double c){
116     if(sgn(a) == 0){
117         s = Point(0,-c/b);
118         e = Point(1,-c/b);
119     }
120     else if(sgn(b) == 0){
121         s = Point(-c/a,0);
122         e = Point(-c/a,1);
123     }
124     else{
125         s = Point(0,-c/b);
126         e = Point(1,(-c-a)/b);
127     }
128 }
129 void input(){
130     s.input();
131     e.input();
132 }
133 void adjust(){
134     if(e < s)swap(s,e);
135 }
136 //求线段长度
137 double length(){
138     return s.distance(e);
139 }
140 //返回直线倾斜角  $0 \leq \text{angle} < \pi$ 
141 double angle(){
142     double k = atan2(e.y-s.y,e.x-s.x);
143     if(sgn(k) < 0)k += pi;
144     if(sgn(k-pi) == 0)k -= pi;
145     return k;
146 }
147 //点和直线关系`
148 //1 在左侧`
149 //2 在右侧`
150 //3 在直线上`
151 int relation(Point p){
152     int c = sgn((p-s)^(e-s));
153     if(c < 0)return 1;
154     else if(c > 0)return 2;
155     else return 3;
156 }
157 //点在线段上的判断
158 bool pointonseg(Point p){
159     return sgn((p-s)^(e-s)) == 0 && sgn((p-s)*(p-e)) <= 0;
160 }
161 //两向量平行(对应直线平行或重合)`
162 bool parallel(Line v){
163     return sgn((e-s)^(v.e-v.s)) == 0;

```

```

164     }
165     //`两线段相交判断`
166     //`2 规范相交`
167     //`1 非规范相交`
168     //`0 不相交`
169     int segcrossseg(Line v){
170         int d1 = sgn((e-s)^(v.s-s));
171         int d2 = sgn((e-s)^(v.e-s));
172         int d3 = sgn((v.e-v.s)^(s-v.s));
173         int d4 = sgn((v.e-v.s)^(e-v.s));
174         if( (d1^d2)==-2 && (d3^d4)==-2 )return 2;
175         return (d1==0 && sgn((v.s-s)*(v.s-e))<=0) ||
176             (d2==0 && sgn((v.e-s)*(v.e-e))<=0) ||
177             (d3==0 && sgn((s-v.s)*(s-v.e))<=0) ||
178             (d4==0 && sgn((e-v.s)*(e-v.e))<=0);
179     }
180     //`直线和线段相交判断`
181     //`-*this line -v seg`
182     //`2 规范相交`
183     //`1 非规范相交`
184     //`0 不相交`
185     int linecrossseg(Line v){
186         int d1 = sgn((e-s)^(v.s-s));
187         int d2 = sgn((e-s)^(v.e-s));
188         if((d1^d2)==-2) return 2;
189         return (d1==0 || d2==0);
190     }
191     //`两直线关系`
192     //`0 平行`
193     //`1 重合`
194     //`2 相交`
195     int linecrossline(Line v){
196         if((*this).parallel(v))
197             return v.relation(s)==3;
198         return 2;
199     }
200     //`求两直线的交点`
201     //`要保证两直线不平行或重合`
202     Point crosspoint(Line v){
203         double a1 = (v.e-v.s)^(s-v.s);
204         double a2 = (v.e-v.s)^(e-v.s);
205         return Point((s.x*a2-e.x*a1)/(a2-a1),(s.y*a2-e.y*a1)/(a2-a1));
206     }
207     //点到直线的距离
208     double dispointtoline(Point p){
209         return fabs((p-s)^(e-s))/length();
210     }
211     //点到线段的距离
212     double dispointtoseg(Point p){
213         if(sgn((p-s)*(e-s))<0 || sgn((p-e)*(s-e))<0)
214             return min(p.distance(s),p.distance(e));
215         return dispointtoline(p);
216     }

```

```

217  //`返回线段到线段的距离`
218  //`前提是两线段不相交, 相交距离就是0了`
219  double dissegtoseg(Line v){
220      return min(min(dispointtoseg(v.s),dispointtoseg(v.e)),min(v.dispointtoseg(s),
221          v.dispointtoseg(e)));
222  }
223  //`返回点p在直线上的投影`
224  Point lineprog(Point p){
225      return s + ( ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
226  }
227  //`返回点p关于直线的对称点`
228  Point symmetrpoint(Point p){
229      Point q = lineprog(p);
230      return Point(2*q.x-p.x,2*q.y-p.y);
231  };
232  //圆
233  struct circle{
234      Point p; //圆心
235      double r; //半径
236      circle(){}
237      circle(Point _p,double _r){
238          p = _p;
239          r = _r;
240      }
241      circle(double x,double y,double _r){
242          p = Point(x,y);
243          r = _r;
244      }
245      //`三角形的外接圆`
246      //`需要Point的+ / rotate() 以及Line的crosspoint()`
247      //`利用两条边的中垂线得到圆心`
248      //`测试: UVA12304`
249      circle(Point a,Point b,Point c){
250          Line u = Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
251          Line v = Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
252          p = u.crosspoint(v);
253          r = p.distance(a);
254      }
255      //`三角形的内切圆`
256      //`参数bool t没有作用, 只是为了和上面外接圆函数区别`
257      //`测试: UVA12304`
258      circle(Point a,Point b,Point c,bool t){
259          Line u,v;
260          double m = atan2(b.y-a.y,b.x-a.x), n = atan2(c.y-a.y,c.x-a.x);
261          u.s = a;
262          u.e = u.s + Point(cos((n+m)/2),sin((n+m)/2));
263          v.s = b;
264          m = atan2(a.y-b.y,a.x-b.x), n = atan2(c.y-b.y,c.x-b.x);
265          v.e = v.s + Point(cos((n+m)/2),sin((n+m)/2));
266          p = u.crosspoint(v);
267          r = Line(a,b).dispointtoseg(p);
268      }

```

```

269 //输入
270 void input(){
271     p.input();
272     scanf("%lf",&r);
273 }
274 //输出
275 void output(){
276     printf("%.2lf %.2lf %.2lf\n",p.x,p.y,r);
277 }
278 bool operator == (circle v){
279     return (p==v.p) && sgn(r-v.r)==0;
280 }
281 bool operator < (circle v)const{
282     return ((p<v.p)||((p==v.p)&&sgn(r-v.r)<0));
283 }
284 //面积
285 double area(){
286     return pi*r*r;
287 }
288 //周长
289 double circumference(){
290     return 2*pi*r;
291 }
292 //`点和圆的关系`
293 //`0 圆外`
294 //`1 圆上`
295 //`2 圆内`
296 int relation(Point b){
297     double dst = b.distance(p);
298     if(sgn(dst-r) < 0)return 2;
299     else if(sgn(dst-r)==0)return 1;
300     return 0;
301 }
302 //`线段和圆的关系`
303 //`比较的是圆心到线段的距离和半径的关系`
304 int relationseg(Line v){
305     double dst = v.dispointtoseg(p);
306     if(sgn(dst-r) < 0)return 2;
307     else if(sgn(dst-r) == 0)return 1;
308     return 0;
309 }
310 //`直线和圆的关系`
311 //`比较的是圆心到直线的距离和半径的关系`
312 int relationline(Line v){
313     double dst = v.dispointtoline(p);
314     if(sgn(dst-r) < 0)return 2;
315     else if(sgn(dst-r) == 0)return 1;
316     return 0;
317 }
318 //`两圆的关系`
319 //`5 相离`
320 //`4 外切`
321 //`3 相交`

```

```

322  //`2 内切`
323  //`1 内含`
324  //`需要Point的distance`
325  //`测试: UVA12304`
326  int relationcircle(circle v){
327      double d = p.distance(v.p);
328      if(sgn(d-r-v.r) > 0)return 5;
329      if(sgn(d-r-v.r) == 0)return 4;
330      double l = fabs(r-v.r);
331      if(sgn(d-r-v.r)<0 && sgn(d-l)>0)return 3;
332      if(sgn(d-l)==0)return 2;
333      if(sgn(d-l)<0)return 1;
334  }
335  //`求两个圆的交点, 返回0表示没有交点, 返回1是一个交点, 2是两个交点`
336  //`需要relationcircle`
337  //`测试: UVA12304`
338  int pointcrosscircle(circle v,Point &p1,Point &p2){
339      int rel = relationcircle(v);
340      if(rel == 1 || rel == 5)return 0;
341      double d = p.distance(v.p);
342      double l = (d*d+r*r-v.r*v.r)/(2*d);
343      double h = sqrt(r*r-l*l);
344      Point tmp = p + (v.p-p).trunc(l);
345      p1 = tmp + ((v.p-p).rotleft().trunc(h));
346      p2 = tmp + ((v.p-p).rotright().trunc(h));
347      if(rel == 2 || rel == 4)
348          return 1;
349      return 2;
350  }
351  //`求直线和圆的交点, 返回交点个数`
352  int pointcrossline(Line v,Point &p1,Point &p2){
353      if(!(*this).relationline(v))return 0;
354      Point a = v.lineprog(p);
355      double d = v.dispointtoline(p);
356      d = sqrt(r*r-d*d);
357      if(sgn(d) == 0){
358          p1 = a;
359          p2 = a;
360          return 1;
361      }
362      p1 = a + (v.e-v.s).trunc(d);
363      p2 = a - (v.e-v.s).trunc(d);
364      return 2;
365  }
366  //`得到过a,b两点, 半径为r1的两个圆`
367  int gercircle(Point a,Point b,double r1,circle &c1,circle &c2){
368      circle x(a,r1),y(b,r1);
369      int t = x.pointcrosscircle(y,c1.p,c2.p);
370      if(!t)return 0;
371      c1.r = c2.r = r;
372      return t;
373  }
374  //`得到与直线u相切, 过点q,半径为r1的圆`

```

```

375 //`测试: UVA12304`
376 int getcircle(Line u,Point q,double r1,circle &c1,circle &c2){
377     double dis = u.dispointtoline(q);
378     if(sgn(dis-r1*2)>0)return 0;
379     if(sgn(dis) == 0){
380         c1.p = q + ((u.e-u.s).rotleft().trunc(r1));
381         c2.p = q + ((u.e-u.s).rotright().trunc(r1));
382         c1.r = c2.r = r1;
383         return 2;
384     }
385     Line u1 = Line((u.s + (u.e-u.s).rotleft().trunc(r1)),(u.e + (u.e-u.s).rotleft
        ().trunc(r1)));
386     Line u2 = Line((u.s + (u.e-u.s).rotright().trunc(r1)),(u.e + (u.e-u.s).
        rotright().trunc(r1)));
387     circle cc = circle(q,r1);
388     Point p1,p2;
389     if(!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
390     c1 = circle(p1,r1);
391     if(p1 == p2){
392         c2 = c1;
393         return 1;
394     }
395     c2 = circle(p2,r1);
396     return 2;
397 }
398 //`同时与直线u,v相切, 半径为r1的圆`
399 //`测试: UVA12304`
400 int getcircle(Line u,Line v,double r1,circle &c1,circle &c2,circle &c3,circle &
    c4){
401     if(u.parallel(v))return 0;//两直线平行
402     Line u1 = Line(u.s + (u.e-u.s).rotleft().trunc(r1),u.e + (u.e-u.s).rotleft().
        trunc(r1));
403     Line u2 = Line(u.s + (u.e-u.s).rotright().trunc(r1),u.e + (u.e-u.s).rotright
        ().trunc(r1));
404     Line v1 = Line(v.s + (v.e-v.s).rotleft().trunc(r1),v.e + (v.e-v.s).rotleft().
        trunc(r1));
405     Line v2 = Line(v.s + (v.e-v.s).rotright().trunc(r1),v.e + (v.e-v.s).rotright
        ().trunc(r1));
406     c1.r = c2.r = c3.r = c4.r = r1;
407     c1.p = u1.crosspoint(v1);
408     c2.p = u1.crosspoint(v2);
409     c3.p = u2.crosspoint(v1);
410     c4.p = u2.crosspoint(v2);
411     return 4;
412 }
413 //`同时与不相交圆cx,cy相切, 半径为r1的圆`
414 //`测试: UVA12304`
415 int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2){
416     circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
417     int t = x.pointcrosscircle(y,c1.p,c2.p);
418     if(!t)return 0;
419     c1.r = c2.r = r1;
420     return t;

```



```

421     }
422
423     ///过一点作圆的切线(先判断点和圆的关系)
424     ///测试: UVA12304
425     int tangentline(Point q, Line &u, Line &v){
426         int x = relation(q);
427         if(x == 2) return 0;
428         if(x == 1){
429             u = Line(q, q + (q-p).rotleft());
430             v = u;
431             return 1;
432         }
433         double d = p.distance(q);
434         double l = r*r/d;
435         double h = sqrt(r*r-l*l);
436         u = Line(q, p + ((q-p).trunc(l) + (q-p).rotleft().trunc(h)));
437         v = Line(q, p + ((q-p).trunc(l) + (q-p).rotright().trunc(h)));
438         return 2;
439     }
440     ///求两圆相交的面积
441     double areacircle(circle v){
442         int rel = relationcircle(v);
443         if(rel >= 4) return 0.0;
444         if(rel <= 2) return min(area(), v.area());
445         double d = p.distance(v.p);
446         double hf = (r+v.r+d)/2.0;
447         double ss = 2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
448         double a1 = acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
449         a1 = a1*r*r;
450         double a2 = acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
451         a2 = a2*v.r*v.r;
452         return a1+a2-ss;
453     }
454     ///求圆和三角形pab的相交面积
455     ///测试: POJ3675 HDU3982 HDU2892
456     double areatriangle(Point a, Point b){
457         if(sgn((p-a)^(p-b)) == 0) return 0.0;
458         Point q[5];
459         int len = 0;
460         q[len++] = a;
461         Line l(a, b);
462         Point p1, p2;
463         if(pointcrossline(l, q[1], q[2]) == 2){
464             if(sgn((a-q[1])*(b-q[1])) < 0) q[len++] = q[1];
465             if(sgn((a-q[2])*(b-q[2])) < 0) q[len++] = q[2];
466         }
467         q[len++] = b;
468         if(len == 4 && sgn((q[0]-q[1])*(q[2]-q[1])) > 0) swap(q[1], q[2]);
469         double res = 0;
470         for(int i = 0; i < len-1; i++){
471             if(relation(q[i]) == 0 || relation(q[i+1]) == 0){
472                 double arg = p.rad(q[i], q[i+1]);
473                 res += r*r*arg/2.0;

```

```

474     }
475     else{
476         res += fabs((q[i]-p)^(q[i+1]-p))/2.0;
477     }
478 }
479 return res;
480 }
481 };
482
483 struct polygon{
484     int n;
485     Point p[maxp];
486     Line l[maxp];
487     void input(int _n){
488         n = _n;
489         for(int i = 0; i < n; i++){
490             p[i].input();
491         }
492     void add(Point q){
493         p[n++] = q;
494     }
495     void getline(){
496         for(int i = 0; i < n; i++){
497             l[i] = Line(p[i], p[(i+1)%n]);
498         }
499     }
500     struct cmp{
501         Point p;
502         cmp(const Point &p0){p = p0;}
503         bool operator()(const Point &aa, const Point &bb){
504             Point a = aa, b = bb;
505             int d = sgn((a-p)^(b-p));
506             if(d == 0){
507                 return sgn(a.distance(p)-b.distance(p)) < 0;
508             }
509             return d > 0;
510         }
511     };
512     //`进行极角排序`
513     //`首先需要找到最左下角的点`
514     //`需要重载号好Point的 < 操作符(min函数要用)`
515     void norm(){
516         Point mi = p[0];
517         for(int i = 1; i < n; i++) mi = min(mi, p[i]);
518         sort(p, p+n, cmp(mi));
519     }
520     //`得到凸包`
521     //`得到的凸包里面的点编号是0~n-1的`
522     //`两种凸包的方法`
523     //`注意如果有影响, 要特判下所有点共点, 或者共线的特殊情况`
524     //`测试 LightOJ1203 LightOJ1239`
525     void getconvex(polygon &convex){
526         sort(p, p+n);

```

```

527     convex.n = n;
528     for(int i = 0; i < min(n, 2); i++){
529         convex.p[i] = p[i];
530     }
531     if(convex.n == 2 && (convex.p[0] == convex.p[1])) convex.n--; //特判
532     if(n <= 2) return;
533     int &top = convex.n;
534     top = 1;
535     for(int i = 2; i < n; i++){
536         while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)
537             top--;
538         convex.p[++top] = p[i];
539     }
540     int temp = top;
541     convex.p[++top] = p[n-2];
542     for(int i = n-3; i >= 0; i--){
543         while(top != temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <=
544             0)
545             top--;
546         convex.p[++top] = p[i];
547     }
548     if(convex.n == 2 && (convex.p[0] == convex.p[1])) convex.n--; //特判
549     convex.norm(); // 原来得到的是顺时针的点，排序后逆时针`
550 // 得到凸包的另外一种方法`
551 // 测试 LightOJ1203 LightOJ1239`
552 void Graham(polygon &convex){
553     norm();
554     int &top = convex.n;
555     top = 0;
556     if(n == 1){
557         top = 1;
558         convex.p[0] = p[0];
559         return;
560     }
561     if(n == 2){
562         top = 2;
563         convex.p[0] = p[0];
564         convex.p[1] = p[1];
565         if(convex.p[0] == convex.p[1]) top--;
566         return;
567     }
568     convex.p[0] = p[0];
569     convex.p[1] = p[1];
570     top = 2;
571     for(int i = 2; i < n; i++){
572         while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])^(p[i]-convex.p[top-2])) <= 0 )
573             top--;
574         convex.p[top++] = p[i];
575     }
576     if(convex.n == 2 && (convex.p[0] == convex.p[1])) convex.n--; //特判
577 }

```

```

578  //判断是不是凸的`
579  bool isconvex(){
580      bool s[3];
581      memset(s,false,sizeof(s));
582      for(int i = 0;i < n;i++){
583          int j = (i+1)%n;
584          int k = (j+1)%n;
585          s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
586          if(s[0] && s[2])return false;
587      }
588      return true;
589  }
590  //判断点和任意多边形的关系`
591  // 3 点上`
592  // 2 边上`
593  // 1 内部`
594  // 0 外部`
595  int relationpoint(Point q){
596      for(int i = 0;i < n;i++){
597          if(p[i] == q)return 3;
598      }
599      getline();
600      for(int i = 0;i < n;i++){
601          if(l[i].pointonseg(q))return 2;
602      }
603      int cnt = 0;
604      for(int i = 0;i < n;i++){
605          int j = (i+1)%n;
606          int k = sgn((q-p[j])^(p[i]-p[j]));
607          int u = sgn(p[i].y-q.y);
608          int v = sgn(p[j].y-q.y);
609          if(k > 0 && u < 0 && v >= 0)cnt++;
610          if(k < 0 && v < 0 && u >= 0)cnt--;
611      }
612      return cnt != 0;
613  }
614  //直线u切割凸多边形左侧`
615  //注意直线方向`
616  //测试: HDU3982`
617  void convexcut(Line u,polygon &po){
618      int &top = po.n;//注意引用
619      top = 0;
620      for(int i = 0;i < n;i++){
621          int d1 = sgn((u.e-u.s)^(p[i]-u.s));
622          int d2 = sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
623          if(d1 >= 0)po.p[top++] = p[i];
624          if(d1*d2 < 0)po.p[top++] = u.crosspoint(Line(p[i],p[(i+1)%n]));
625      }
626  }
627  //得到周长`
628  //测试 LightOJ1239`
629  double getcircumference(){
630      double sum = 0;

```

```

631     for(int i = 0;i < n;i++){
632         sum += p[i].distance(p[(i+1)%n]);
633     }
634     return sum;
635 }
636 //`得到面积`
637 double getarea(){
638     double sum = 0;
639     for(int i = 0;i < n;i++){
640         sum += (p[i]^p[(i+1)%n]);
641     }
642     return fabs(sum)/2;
643 }
644 //`得到方向`
645 //` 1 表示逆时针, 0表示顺时针`
646 bool getdir(){
647     double sum = 0;
648     for(int i = 0;i < n;i++){
649         sum += (p[i]^p[(i+1)%n]);
650     }
651     if(sgn(sum) > 0)return 1;
652     return 0;
653 }
654 //`得到重心`
655 Point getbarycentre(){
656     Point ret(0,0);
657     double area = 0;
658     for(int i = 1;i < n-1;i++){
659         double tmp = (p[i]-p[0])^(p[i+1]-p[0]);
660         if(sgn(tmp) == 0)continue;
661         area += tmp;
662         ret.x += (p[0].x+p[i].x+p[i+1].x)/3*tmp;
663         ret.y += (p[0].y+p[i].y+p[i+1].y)/3*tmp;
664     }
665     if(sgn(area)) ret = ret/area;
666     return ret;
667 }
668 //`多边形和圆交的面积`
669 //`测试: POJ3675 HDU3982 HDU2892`
670 double areacircle(circle c){
671     double ans = 0;
672     for(int i = 0;i < n;i++){
673         int j = (i+1)%n;
674         if(sgn( (p[j]-c.p)^(p[i]-c.p) ) >= 0)
675             ans += c.areastriangle(p[i],p[j]);
676         else ans -= c.areastriangle(p[i],p[j]);
677     }
678     return fabs(ans);
679 }
680 //`多边形和圆关系`
681 //` 2 圆完全在多边形内`
682 //` 1 圆在多边形里面, 碰到了多边形边界`
683 //` 0 其它`
684 int relationcircle(circle c){

```

```

684     getline();
685     int x = 2;
686     if(relationpoint(c.p) != 1)return 0;//圆心不在内部
687     for(int i = 0;i < n;i++){
688         if(c.relationseg(l[i])==2)return 0;
689         if(c.relationseg(l[i])==1)x = 1;
690     }
691     return x;
692 }
693 };
694 //`AB X AC`
695 double cross(Point A,Point B,Point C){
696     return (B-A)^(C-A);
697 }
698 //`AB*AC`
699 double dot(Point A,Point B,Point C){
700     return (B-A)*(C-A);
701 }
702 //`最小矩形面积覆盖`
703 //`A 必须是凸包(而且是逆时针顺序)`
704 //`测试 UVA 10173`
705 double minRectangleCover(polygon A){
706     //`要特判A.n < 3的情况`
707     if(A.n < 3)return 0.0;
708     A.p[A.n] = A.p[0];
709     double ans = -1;
710     int r = 1, p = 1, q;
711     for(int i = 0;i < A.n;i++){
712         //`卡出离边A.p[i] - A.p[i+1]最远的点`
713         while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],A.p[i+1],A.p[r]) )
714             >= 0 )
715             r = (r+1)%A.n;
716         //`卡出A.p[i] - A.p[i+1]方向上正向n最远的点`
717         while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0
718             )
719             p = (p+1)%A.n;
720         //`卡出A.p[i] - A.p[i+1]方向上负向最远的点`
721         while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i+1],A.p[q])) <= 0)
722             q = (q+1)%A.n;
723         double d = (A.p[i] - A.p[i+1]).len2();
724         double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
725             (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
726         if(ans < 0 || ans > tmp)ans = tmp;
727     }
728     return ans;
729 }
730 //`直线切凸多边形`
731 //`多边形是逆时针的, 在q1q2的左侧`
732 //`测试:HDU3982`
733 vector<Point> convexCut(const vector<Point> &ps,Point q1,Point q2){
734     vector<Point>q;

```

```

735     int n = ps.size();
736     for(int i = 0; i < n; i++){
737         Point p1 = ps[i], p2 = ps[(i+1)%n];
738         int d1 = sgn((q2-q1)^(p1-q1)), d2 = sgn((q2-q1)^(p2-q1));
739         if(d1 >= 0)
740             qs.push_back(p1);
741         if(d1 * d2 < 0)
742             qs.push_back(Line(p1,p2).crosspoint(Line(q1,q2)));
743     }
744     return qs;
745 }
746 //`半平面交`
747 //`测试 POJ3335 POJ1474 POJ1279`
748 //*****
749 struct halfplane:public Line{
750     double angle;
751     halfplane(){}
752     //`表示向量s->e逆时针(左侧)的半平面`
753     halfplane(Point _s,Point _e){
754         s = _s;
755         e = _e;
756     }
757     halfplane(Line v){
758         s = v.s;
759         e = v.e;
760     }
761     void calcangle(){
762         angle = atan2(e.y-s.y,e.x-s.x);
763     }
764     bool operator <(const halfplane &b)const{
765         return angle < b.angle;
766     }
767 };
768 struct halfplanes{
769     int n;
770     halfplane hp[maxp];
771     Point p[maxp];
772     int que[maxp];
773     int st,ed;
774     void push(halfplane tmp){
775         hp[n++] = tmp;
776     }
777     //去重
778     void unique(){
779         int m = 1;
780         for(int i = 1; i < n; i++){
781             if(sgn(hp[i].angle-hp[i-1].angle) != 0)
782                 hp[m++] = hp[i];
783             else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s) ) > 0)
784                 hp[m-1] = hp[i];
785         }
786         n = m;
787     }

```

```

788 bool halfplaneinsert(){
789     for(int i = 0;i < n;i++)hp[i].calcangle();
790     sort(hp, hp+n);
791     unique();
792     que[st=0] = 0;
793     que[ed=1] = 1;
794     p[1] = hp[0].crosspoint(hp[1]);
795     for(int i = 2;i < n;i++){
796         while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)ed--;
797         while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)st++;
798         que[++ed] = i;
799         if(hp[i].parallel(hp[que[ed-1]]))return false;
800         p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
801     }
802     while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0)ed--;
803     while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-hp[que[ed]].s))<0)st++;
804     if(st+1==ed)return false;
805     return true;
806 }
807 //`得到最后半平面交得到的凸多边形`
808 //`需要先调用halfplaneinsert() 且返回true`
809 void getconvex(polygon &con){
810     p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
811     con.n = ed-st+1;
812     for(int j = st,i = 0;j <= ed;i++,j++){
813         con.p[i] = p[j];
814     }
815 };
816 //*****
817
818 const int maxn = 1010;
819 struct circles{
820     circle c[maxn];
821     double ans[maxn];//`ans[i]表示被覆盖了i次的面积`
822     double pre[maxn];
823     int n;
824     circles(){
825         void add(circle cc){
826             c[n++] = cc;
827         }
828         //`x包含在y中`
829         bool inner(circle x,circle y){
830             if(x.relationcircle(y) != 1)return 0;
831             return sgn(x.r-y.r)<=0?1:0;
832         }
833         //圆的面积并去掉内含的圆
834         void init_or(){
835             bool mark[maxn] = {0};
836             int i,j,k=0;
837             for(i = 0;i < n;i++){
838                 for(j = 0;j < n;j++){

```



```

839         if(i != j && !mark[j]){
840             if( (c[i]==c[j])||inner(c[i],c[j]) )break;
841         }
842         if(j < n)mark[i] = 1;
843     }
844     for(i = 0;i < n;i++)
845         if(!mark[i])
846             c[k++] = c[i];
847     n = k;
848 }
849 // 圆的面积交去掉内含的圆`
850 void init_add(){
851     int i,j,k;
852     bool mark[maxn] = {0};
853     for(i = 0;i < n;i++){
854         for(j = 0;j < n;j++){
855             if(i != j && !mark[j]){
856                 if( (c[i]==c[j])||inner(c[j],c[i]) )break;
857             }
858             if(j < n)mark[i] = 1;
859         }
860     }
861     for(i = 0;i < n;i++)
862         if(!mark[i])
863             c[k++] = c[i];
864     n = k;
865 }
866 // 半径为r的圆，弧度为th对应的弓形的面积`
867 double areaarc(double th,double r){
868     return 0.5*r*r*(th-sin(th));
869 }
870 // 测试SPOJVCIRCLES SPOJCIRUT`
871 // SPOJVCIRCLES求n个圆并的面积，需要加上init_or()去掉重复圆（否则WA）`
872 // SPOJCIRUT 是求被覆盖k次的面积，不能加init_or()`
873 // 对于求覆盖多少次面积的问题，不能解决相同圆，而且不能init_or()`
874 // 求多圆面积并，需要init_or,其中一个目的就是去掉相同圆`
875 void getarea(){
876     memset(ans,0,sizeof(ans));
877     vector<pair<double,int> >v;
878     for(int i = 0;i < n;i++){
879         v.clear();
880         v.push_back(make_pair(-pi,1));
881         v.push_back(make_pair(pi,-1));
882         for(int j = 0;j < n;j++){
883             if(i != j){
884                 Point q = (c[j].p - c[i].p);
885                 double ab = q.len(),ac = c[i].r, bc = c[j].r;
886                 if(sgn(ab+ac-bc)<=0){
887                     v.push_back(make_pair(-pi,1));
888                     v.push_back(make_pair(pi,-1));
889                     continue;
890                 }
891                 if(sgn(ab+bc-ac)<=0)continue;
892                 if(sgn(ab-ac-bc)>0)continue;

```

```

892         double th = atan2(q.y,q.x), fai = acos((ac*ac+ab*ab-bc*bc)/(2.0*ac*
           ab));
893         double a0 = th-fai;
894         if(sgn(a0+pi)<0)a0+=2*pi;
895         double a1 = th+fai;
896         if(sgn(a1-pi)>0)a1-=2*pi;
897         if(sgn(a0-a1)>0){
898             v.push_back(make_pair(a0,1));
899             v.push_back(make_pair(pi,-1));
900             v.push_back(make_pair(-pi,1));
901             v.push_back(make_pair(a1,-1));
902         }
903         else{
904             v.push_back(make_pair(a0,1));
905             v.push_back(make_pair(a1,-1));
906         }
907     }
908     sort(v.begin(),v.end());
909     int cur = 0;
910     for(int j = 0;j < v.size();j++){
911         if(cur && sgn(v[j].first-pre[cur])){
912             ans[cur] += areaarc(v[j].first-pre[cur],c[i].r);
913             ans[cur] += 0.5*(Point(c[i].p.x+c[i].r*cos(pre[cur]),c[i].p.y+c[i].
               r*sin(pre[cur]))^Point(c[i].p.x+c[i].r*cos(v[j].first),c[i].p.y
               +c[i].r*sin(v[j].first)));
914         }
915         cur += v[j].second;
916         pre[cur] = v[j].first;
917     }
918 }
919 for(int i = 1;i < n;i++){
920     ans[i] -= ans[i+1];
921 }
922 };

```

## 7.2 三维几何

```

1  const double eps = 1e-8;
2  int sgn(double x){
3      if(fabs(x) < eps)return 0;
4      if(x < 0)return -1;
5      else return 1;
6  }
7  struct Point3{
8      double x,y,z;
9      Point3(double _x = 0,double _y = 0,double _z = 0){
10         x = _x;
11         y = _y;
12         z = _z;
13     }
14     void input(){
15         scanf("%lf%lf%lf",&x,&y,&z);

```

```

16     }
17     void output(){
18         printf("%.21f %.21f %.21f\n",x,y,z);
19     }
20     bool operator ==(const Point3 &b)const{
21         return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 && sgn(z-b.z) == 0;
22     }
23     bool operator <(const Point3 &b)const{
24         return sgn(x-b.x)==0?(sgn(y-b.y)==0?sgn(z-b.z)<0:y<b.y):x<b.x;
25     }
26     double len(){
27         return sqrt(x*x+y*y+z*z);
28     }
29     double len2(){
30         return x*x+y*y+z*z;
31     }
32     double distance(const Point3 &b)const{
33         return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.z)*(z-b.z));
34     }
35     Point3 operator -(const Point3 &b)const{
36         return Point3(x-b.x,y-b.y,z-b.z);
37     }
38     Point3 operator +(const Point3 &b)const{
39         return Point3(x+b.x,y+b.y,z+b.z);
40     }
41     Point3 operator *(const double &k)const{
42         return Point3(x*k,y*k,z*k);
43     }
44     Point3 operator /(const double &k)const{
45         return Point3(x/k,y/k,z/k);
46     }
47     //点乘
48     double operator *(const Point3 &b)const{
49         return x*b.x+y*b.y+z*b.z;
50     }
51     //叉乘
52     Point3 operator ^(const Point3 &b)const{
53         return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
54     }
55     double rad(Point3 a,Point3 b){
56         Point3 p = (*this);
57         return acos( ( (a-p)*(b-p) )/ (a.distance(p)*b.distance(p)) );
58     }
59     //变换长度
60     Point3 trunc(double r){
61         double l = len();
62         if(!sgn(l))return *this;
63         r /= l;
64         return Point3(x*r,y*r,z*r);
65     }
66 };
67 struct Line3
68 {

```

```

69     Point3 s,e;
70     Line3(){ }
71     Line3(Point3 _s,Point3 _e)
72     {
73         s = _s;
74         e = _e;
75     }
76     bool operator ==(const Line3 v)
77     {
78         return (s==v.s)&&(e==v.e);
79     }
80     void input()
81     {
82         s.input();
83         e.input();
84     }
85     double length()
86     {
87         return s.distance(e);
88     }
89     //点到直线距离
90     double dispointtoline(Point3 p)
91     {
92         return ((e-s)^(p-s)).len()/s.distance(e);
93     }
94     //点到线段距离
95     double dispointtoseg(Point3 p)
96     {
97         if(sgn((p-s)*(e-s)) < 0 || sgn((p-e)*(s-e)) < 0)
98             return min(p.distance(s),e.distance(p));
99         return dispointtoline(p);
100    }
101    //返回点p在直线上的投影
102    Point3 lineprog(Point3 p)
103    {
104        return s + ( ((e-s)*((e-s)*(p-s)))/((e-s).len2()) );
105    }
106    //p绕此向量逆时针ang角度
107    Point3 rotate(Point3 p,double ang)
108    {
109        if(sgn(((s-p)^(e-p)).len()) == 0)return p;
110        Point3 f1 = (e-s)^(p-s);
111        Point3 f2 = (e-s)^(f1);
112        double len = ((s-p)^(e-p)).len()/s.distance(e);
113        f1 = f1.trunc(len); f2 = f2.trunc(len);
114        Point3 h = p+f2;
115        Point3 pp = h+f1;
116        return h + ((p-h)*cos(ang)) + ((pp-h)*sin(ang));
117    }
118    //点在直线上
119    bool pointonseg(Point3 p)
120    {
121        return sgn( ((s-p)^(e-p)).len() ) == 0 && sgn((s-p)*(e-p)) == 0;

```

```

122     }
123 };
124 struct Plane
125 {
126     Point3 a,b,c,o; // 平面上的三个点, 以及法向量`
127     Plane(){}
128     Plane(Point3 _a,Point3 _b,Point3 _c)
129     {
130         a = _a;
131         b = _b;
132         c = _c;
133         o = pvec();
134     }
135     Point3 pvec()
136     {
137         return (b-a)^(c-a);
138     }
139     // `ax+by+cz+d = 0`
140     Plane(double _a,double _b,double _c,double _d)
141     {
142         o = Point3(_a,_b,_c);
143         if(sgn(_a) != 0)
144             a = Point3((-_d-_c-_b)/_a,1,1);
145         else if(sgn(_b) != 0)
146             a = Point3(1,(-_d-_c-_a)/_b,1);
147         else if(sgn(_c) != 0)
148             a = Point3(1,1,(-_d-_a-_b)/_c);
149     }
150     // 点在平面上的判断`
151     bool pointonplane(Point3 p)
152     {
153         return sgn((p-a)*o) == 0;
154     }
155     // 两平面夹角`
156     double angleplane(Plane f)
157     {
158         return acos((o*f.o)/(o.len()*f.o.len()));
159     }
160     // 平面和直线的交点, 返回值是交点个数`
161     int crossline(Line3 u,Point3 &p)
162     {
163         double x = o*(u.e-a);
164         double y = o*(u.s-a);
165         double d = x-y;
166         if(sgn(d) == 0)return 0;
167         p = ((u.s*x)-(u.e*y))/d;
168         return 1;
169     }
170     // 点到平面最近点(也就是投影)`
171     Point3 pointtoplane(Point3 p)
172     {
173         Line3 u = Line3(p,p+o);
174         crossline(u,p);

```

```

175     return p;
176 }
177 //`平面和平面的交线`
178 int crossplane(Plane f,Line3 &u)
179 {
180     Point3 oo = o^f.o;
181     Point3 v = o^oo;
182     double d = fabs(f.o*v);
183     if(sgn(d) == 0)return 0;
184     Point3 q = a + (v*(f.o*(f.a-a))/d);
185     u = Line3(q,q+oo);
186     return 1;
187 }
188 };

```

### 7.3 三维凸包

```

1  const double eps = 1e-8;
2  const int MAXN = 550;
3  int sgn(double x){
4      if(fabs(x) < eps)return 0;
5      if(x < 0)return -1;
6      else return 1;
7  }
8  struct Point3{
9      double x,y,z;
10     Point3(double _x = 0, double _y = 0, double _z = 0){
11         x = _x;
12         y = _y;
13         z = _z;
14     }
15     void input(){
16         scanf("%lf%lf%lf",&x,&y,&z);
17     }
18     bool operator ==(const Point3 &b)const{
19         return sgn(x-b.x) == 0 && sgn(y-b.y) == 0 && sgn(z-b.z) == 0;
20     }
21     double len(){
22         return sqrt(x*x+y*y+z*z);
23     }
24     double len2(){
25         return x*x+y*y+z*z;
26     }
27     double distance(const Point3 &b)const{
28         return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.z)*(z-b.z));
29     }
30     Point3 operator -(const Point3 &b)const{
31         return Point3(x-b.x,y-b.y,z-b.z);
32     }
33     Point3 operator +(const Point3 &b)const{
34         return Point3(x+b.x,y+b.y,z+b.z);
35     }

```

```

36 Point3 operator *(const double &k)const{
37     return Point3(x*k,y*k,z*k);
38 }
39 Point3 operator /(const double &k)const{
40     return Point3(x/k,y/k,z/k);
41 }
42 //点乘
43 double operator *(const Point3 &b)const{
44     return x*b.x + y*b.y + z*b.z;
45 }
46 //叉乘
47 Point3 operator ^(const Point3 &b)const{
48     return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
49 }
50 };
51 struct CH3D{
52     struct face{
53         //表示凸包一个面上的三个点的编号
54         int a,b,c;
55         //表示该面是否属于最终的凸包上的面
56         bool ok;
57     };
58     //初始顶点数
59     int n;
60     Point3 P[MAXN];
61     //凸包表面的三角形数
62     int num;
63     //凸包表面的三角形
64     face F[8*MAXN];
65     int g[MAXN][MAXN];
66     //叉乘
67     Point3 cross(const Point3 &a,const Point3 &b,const Point3 &c){
68         return (b-a)^(c-a);
69     }
70     //`三角形面积*2`
71     double area(Point3 a,Point3 b,Point3 c){
72         return ((b-a)^(c-a)).len();
73     }
74     //`四面体有向面积*6`
75     double volume(Point3 a,Point3 b,Point3 c,Point3 d){
76         return ((b-a)^(c-a))*(d-a);
77     }
78     //`正: 点在面同向`
79     double dblcmp(Point3 &p,face &f){
80         Point3 p1 = P[f.b] - P[f.a];
81         Point3 p2 = P[f.c] - P[f.a];
82         Point3 p3 = p - P[f.a];
83         return (p1^p2)*p3;
84     }
85     void deal(int p,int a,int b){
86         int f = g[a][b];
87         face add;
88         if(F[f].ok){

```

```

89         if(dblcmp(P[p],F[f]) > eps)
90             dfs(p,f);
91         else {
92             add.a = b;
93             add.b = a;
94             add.c = p;
95             add.ok = true;
96             g[p][b] = g[a][p] = g[b][a] = num;
97             F[num++] = add;
98         }
99     }
100 }
101 //递归搜索所有应该从凸包内删除的面
102 void dfs(int p,int now){
103     F[now].ok = false;
104     deal(p,F[now].b,F[now].a);
105     deal(p,F[now].c,F[now].b);
106     deal(p,F[now].a,F[now].c);
107 }
108 bool same(int s,int t){
109     Point3 &a = P[F[s].a];
110     Point3 &b = P[F[s].b];
111     Point3 &c = P[F[s].c];
112     return fabs(volume(a,b,c,P[F[t].a])) < eps &&
113            fabs(volume(a,b,c,P[F[t].b])) < eps &&
114            fabs(volume(a,b,c,P[F[t].c])) < eps;
115 }
116 //构建三维凸包
117 void create(){
118     num = 0;
119     face add;
120
121     //*****
122     //此段是为了保证前四个点不共面
123     bool flag = true;
124     for(int i = 1;i < n;i++){
125         if(!(P[0] == P[i])){
126             swap(P[1],P[i]);
127             flag = false;
128             break;
129         }
130     }
131     if(flag)return;
132     flag = true;
133     for(int i = 2;i < n;i++){
134         if( ((P[1]-P[0])^(P[i]-P[0])).len() > eps ){
135             swap(P[2],P[i]);
136             flag = false;
137             break;
138         }
139     }
140     if(flag)return;
141     flag = true;

```



```

142     for(int i = 3;i < n;i++){
143         if(fabs( ((P[1]-P[0])^(P[2]-P[0]))*(P[i]-P[0]) ) > eps){
144             swap(P[3],P[i]);
145             flag = false;
146             break;
147         }
148     }
149     if(flag)return;
150     //*****
151
152     for(int i = 0;i < 4;i++){
153         add.a = (i+1)%4;
154         add.b = (i+2)%4;
155         add.c = (i+3)%4;
156         add.ok = true;
157         if(dblcmp(P[i],add) > 0)swap(add.b,add.c);
158         g[add.a][add.b] = g[add.b][add.c] = g[add.c][add.a] = num;
159         F[num++] = add;
160     }
161     for(int i = 4;i < n;i++){
162         for(int j = 0;j < num;j++){
163             if(F[j].ok && dblcmp(P[i],F[j]) > eps){
164                 dfs(i,j);
165                 break;
166             }
167         }
168         int tmp = num;
169         num = 0;
170         for(int i = 0;i < tmp;i++){
171             if(F[i].ok)
172                 F[num++] = F[i];
173         }
174         //表面积
175         //`测试: HDU3528`
176         double area(){
177             double res = 0;
178             if(n == 3){
179                 Point3 p = cross(P[0],P[1],P[2]);
180                 return p.len()/2;
181             }
182             for(int i = 0;i < num;i++){
183                 res += area(P[F[i].a],P[F[i].b],P[F[i].c]);
184             }
185             return res/2.0;
186         }
187         double volume(){
188             double res = 0;
189             Point3 tmp = Point3(0,0,0);
190             for(int i = 0;i < num;i++){
191                 res += volume(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
192             }
193             return fabs(res/6);
194         }
195     }
196     //表面三角形个数
197     int triangle(){
198         return num;
199     }

```

```

195     }
196     //表面多边形个数
197     //`测试: HDU3662`
198     int polygon(){
199         int res = 0;
200         for(int i = 0;i < num;i++){
201             bool flag = true;
202             for(int j = 0;j < i;j++){
203                 if(same(i,j)){
204                     flag = 0;
205                     break;
206                 }
207             }
208             res += flag;
209         }
210         return res;
211     }
212     //重心
213     //`测试: HDU4273`
214     Point3 barycenter(){
215         Point3 ans = Point3(0,0,0);
216         Point3 o = Point3(0,0,0);
217         double all = 0;
218         for(int i = 0;i < num;i++){
219             double vol = volume(o,P[F[i].a],P[F[i].b],P[F[i].c]);
220             ans = ans + ((o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0)*vol;
221             all += vol;
222         }
223         ans = ans/all;
224         return ans;
225     }
226     //点到面的距离
227     //`测试: HDU4273`
228     double ptoface(Point3 p,int i){
229         double tmp1 = fabs(volume(P[F[i].a],P[F[i].b],P[F[i].c],p));
230         double tmp2 = ((P[F[i].b]-P[F[i].a])^(P[F[i].c]-P[F[i].a])).len();
231         return tmp1/tmp2;
232     }
233 };
234 CH3D hull;
235 int main()
236 {
237     while(scanf("%d",&hull.n) == 1){
238         for(int i = 0;i < hull.n;i++)hull.P[i].input();
239         hull.create();
240         Point3 p = hull.barycenter();
241         double ans = 1e20;
242         for(int i = 0;i < hull.num;i++)
243             ans = min(ans,hull.ptoface(p,i));
244         printf("%.3lf\n",ans);
245     }
246     return 0;

```

## 7.4 多边形交

```

1  /*
2   * 多边形的交，多边形的边一定是要按逆时针方向给出
3   * 还要判断是凸包还是凹包，调用相应的函数
4   * 面积并，只要和面积减去交即可
5   */
6  #include <bits/stdc++.h>
7  using namespace std;
8  const int maxn = //300;
9  const double eps = 1e-8;
10 int dcmp(double x)
11 {
12     if(x > eps) return 1;
13     return x < -eps ? -1 : 0;
14 }
15 struct Point
16 {
17     double x, y;
18 };
19 double cross(Point a, Point b, Point c) ///叉积
20 {
21     return (a.x-c.x)*(b.y-c.y)-(b.x-c.x)*(a.y-c.y);
22 }
23 Point intersection(Point a, Point b, Point c, Point d)
24 {
25     Point p = a;
26     double t = ((a.x-c.x)*(c.y-d.y)-(a.y-c.y)*(c.x-d.x))/((a.x-b.x)*(c.y-d.y)-(a.y-b.y)*(c.x-d.x));
27     p.x +=(b.x-a.x)*t;
28     p.y +=(b.y-a.y)*t;
29     return p;
30 }
31 //计算多边形面积
32 double PolygonArea(Point p[], int n)
33 {
34     if(n < 3) return 0.0;
35     double s = p[0].y * (p[n-1].x - p[1].x);
36     p[n] = p[0];
37     for(int i = 1; i < n; ++i)
38         s += p[i].y * (p[i-1].x - p[i+1].x);
39     return fabs(s * 0.5);
40 }
41 double CPIA(Point a[], Point b[], int na, int nb)///ConvexPolygonIntersectArea
42 {
43     Point p[20], tmp[20];
44     int tn, sflag, eflag;
45     a[na] = a[0], b[nb] = b[0];
46     memcpy(p, b, sizeof(Point)*(nb + 1));
47     for(int i = 0; i < na && nb > 2; i++)
48     {
49         sflag = dcmp(cross(a[i+1], p[0], a[i]));
50         for(int j = tn = 0; j < nb; j++, sflag = eflag)

```

```

51     {
52         if(sflag>=0) tmp[tn++] = p[j];
53         eflag = dcmp(cross(a[i + 1], p[j + 1],a[i]));
54         if((sflag ^ eflag) == -2)
55             tmp[tn++] = intersection(a[i], a[i + 1], p[j], p[j + 1]); ///求交点
56     }
57     memcpy(p, tmp, sizeof(Point) * tn);
58     nb = tn, p[nb] = p[0];
59 }
60 if(nb < 3) return 0.0;
61 return PolygonArea(p, nb);
62 }
63 double SPIA(Point a[], Point b[], int na, int nb)///SimplePolygonIntersectArea 调用
    此函数
64 {
65     int i, j;
66     Point t1[4], t2[4];
67     double res = 0, num1, num2;
68     a[na] = t1[0] = a[0], b[nb] = t2[0] = b[0];
69     for(i = 2; i < na; i++)
70     {
71         t1[1] = a[i-1], t1[2] = a[i];
72         num1 = dcmp(cross(t1[1], t1[2],t1[0]));
73         if(num1 < 0) swap(t1[1], t1[2]);
74         for(j = 2; j < nb; j++)
75         {
76             t2[1] = b[j - 1], t2[2] = b[j];
77             num2 = dcmp(cross(t2[1], t2[2],t2[0]));
78             if(num2 < 0) swap(t2[1], t2[2]);
79             res += CPIA(t1, t2, 3, 3) * num1 * num2;
80         }
81     }
82     return res;
83 }
84 Point p1[maxn], p2[maxn];
85 int n1, n2;
86 int main()
87 {
88     while(cin>>n1>>n2)
89     {
90         for(int i = 0; i < n1; i++) scanf("%lf%lf", &p1[i].x, &p1[i].y);
91         for(int i = 0; i < n2; i++) scanf("%lf%lf", &p2[i].x, &p2[i].y);
92         double Area = SPIA(p1, p2, n1, n2);
93     }
94     return 0;
95 }

```

## 8 工具

### 8.1 对拍

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int main()
4  {
5      int s,t,s0,cnt=0;
6      while(1)
7      {
8          system("cls");
9          do
10         {
11             system("data.exe > data.txt"); //data是数据生成程序，自己写
12             s=clock();
13             system("TmpCode0.exe < data.txt > try0.out"); //a是要测试的程序
14             t=clock();
15             system("TmpCode1.exe < data.txt > try1.out"); //b是正确的程序
16             s0=clock();
17             printf("Test:%d\n",++cnt);
18             if(system("fc try0.out try1.out > nul"))break;
19             else printf("AC time0: %ldms time1:%ldms\n",t-s,s0-t); //并输出运行时间
20         }while(1);
21         printf("WA time0: %ldms time1:%ldms\n",t-s,s0-t); //运行时间 wa了会停止对拍，wa
           的那一组会在data.txt里，所以说为了方便查找一次不要生成太大量的数据
22         system("fc try0.out try1.out");
23         system("pause>nul");
24     }
25     return 0;
26 }

```

## 8.2 数据生成器

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count()); //É?ú?ý
4  long long rd(long long l,long long r)
5  {
6      unsigned long long ans=rng();
7      return (ans%(r-l+1))+l;
8  }
9  int main()
10 {
11     int T=1;
12     //printf("%d\n",T);
13     while(T--)
14     {
15         int n=rd(1,10000);
16         printf("%d\n",n);
17     }
18     return 0;
19 }

```

## 8.3 cb 环境配置

```
1 settings-->environment-->generalsettings-->Terminal to launch console programs:  
2 把方框里默认的终端改成 gnome-terminal -t $TITLE -x  
3  
4 cb闪退:setting-editor-symbolbrowser 关闭
```