ICPC TEMPLATE

indiewar

2019 年 10 月 24 日

目录

1 一切的开始

1.1 宏定义

by 杜教

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)//注意范围[a.n)
#define per(i,a,n) for (int i=n-1;i>=a;i--)//注意范围[a.n-1]
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
mt19937 mrand(random_device{}());
const ll mod=1000000007;
int rnd(int x) { return mrand() % x;}
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
    for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
ll gcd(ll a,ll b) { return b?gcd(b,a%b):a;}
```

```
// --------------------------------------------------------------------------
```

- CMakeLists.txt (for CLion)

```
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -O2 -Dzerol -Wall")
```

- HDU Assert Patch

```
#ifdef ONLINE_JUDGE
#define assert(condition) if (!(condition)) { int x = 1, y = 0; cout << x / y <<
    endl; }
#endif
```

1.2 快速读

```
inline char nc() {
    static char buf[100000], *p1 = buf, *p2 = buf;
    return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ?
        EOF : *p1++;
}
template <typename T>
bool rn(T& v) {
    static char ch;
    while (ch != EOF && !isdigit(ch)) ch = nc();
    if (ch == EOF) return false;
    for (v = 0; isdigit(ch); ch = nc())
        v = v * 10 + ch - '0';
    return true;
}

template <typename T>
void o(T p) {
    static int stk[70], tp;
    if (p == 0) { putchar('0'); return; }
    if (p < 0) { p = -p; putchar('-'); }
```

```
    while (p) stk[++tp] = p % 10, p /= 10;
    while (tp) putchar(stk[tp--] + '0');
}
```

- 需要初始化
- 需要一次读入
- 不支持负数

```
const int MAXS = 100 * 1024 * 1024;
char buf[MAXS];
template<typename T>
inline bool read(T& x) {
    static char* p = buf;
    x = 0;
    while (*p && !isdigit(*p)) ++p;
    if (!*p) return false;
    while (isdigit(*p)) x = x * 10 + *p++ - 48;
    return true;
}

fread(buf, 1, MAXS, stdin);
```

1.3 对拍

```
#!/usr/bin/env bash
g++ -o r main.cpp -O2 -std=c++11
g++ -o std std.cpp -O2 -std=c++11
while true; do
    python gen.py > in
    ./std < in > stdout
    ./r < in > out
    if test $? -ne 0; then
        exit 0
    fi
    if diff stdout out; then
        printf "AC\n"
```

```
    else
        printf "GG\n"
        exit 0
    fi
done
```

- 快速编译运行

```
#!/bin/bash
g++ $1.cpp -o $1 -O2 -std=c++14 -Wall -Dzerol -g
if $? -eq 0; then
    ./$1
fi
```

2 数据结构

2.1 BIT

```
struct Bit
{
    vector<int> a;
    int sz;
    void init(int n)
    {
        sz=n+5;
        for(int i=1;i<=n+5;i++)
            a.push_back(0);
    }
    int lowbit(int x)
    {
        return x&(-x);
    }
    int query(int x)
    {
        int ans = 0;
        for(;x;x-=lowbit(x))ans+=a[x];
```

```
        return ans;
    }
    void update(int x,int v)
    {
        for(;x<sz;x+=lowbit(x))
            a[x]+=v;
    }
}bit;
```

3 图论

3.1 最短路

3.1.1 floyd

```
for (int k = 1; k <= n; k++) {
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
      f[i][j] = min(f[i][j], f[i][k] + f[k][j]);
    }
  }
}
```

- 找最小环

```
int val[maxn + 1][maxn + 1]; // 原图的邻接矩阵
int floyd(const int &n) {
  static int dis[maxn + 1][maxn + 1]; // 最短路矩阵
  for (int i = 1; i <= n; ++i)
    for (int j = 1; j <= n; ++j) dis[i][j] = val[i][j]; // 初始化最短路矩阵
  int ans = inf;
  for (int k = 1; k <= n; ++k) {
    for (int i = 1; i < k; ++i)
      for (int j = 1; j < i; ++j)
        ans = std::min(ans, dis[i][j] + val[i][k] + val[k][j]); // 更新答案
    for (int i = 1; i <= n; ++i)
```

```
      for (int j = 1; j <= n; ++j)
        dis[i][j] = std::min(
            dis[i][j], dis[i][k] + dis[k][j]); // 正常的 floyd 更新最短路矩阵
  }
  return ans;
}
```

- 利用 floyd 的 dp 思路求解

```
int dp[maxn][maxn][maxn];
int w[maxn];
int s[maxn];
bool cmp(int a,int b)
{
    return w[a] < w[b];
}
rep(i,1,n+1)
{
    rep(j,1,n+1)
    {
       scanf("%d",&dp[i][j][0]);
        rep(k,1,n+1)
        {
            dp[i][j][k] = 1e9;
        }
     }
    s[i] = i;
}
sort(s+1,s+n+1,cmp);
rep(k,1,n+1)
{
    rep(i,1,n+1)
    {
        rep(j,1,n+1)
        {
            dp[i][j][k] = min(dp[i][j][k-1],dp[i][s[k]][k-1]+dp[s[k]][j][k-1]);
        }
```

```
    }
}
```

- 传递闭包已知一个有向图中任意两点之间是否有连边，要求判断任意两点是否连通。

```
for (int k = 1; k <= n; k++)
  for (int i = 1; i <= n; i++)
    if (f[i][k]) f[i] = f[i] & f[k];
```

3.2 网络流

- dinic

```
const int maxn = 4e3+100;
const int maxm = 1e5+100;
const int inf = 0x7f7f7f7f;

typedef struct Dinic
{
    typedef struct Edge
    {
        int u,v,w,nxt;
    } Edge;
    int head[maxn],hcnt;
    int dep[maxn];
    int cur[maxn];
    Edge e[maxm];
    int S,T,N;
    void init()
    {
        memset(head,-1,sizeof head);
        hcnt = 0;
        S = T = N = 0;
    }
    void adde(int u,int v,int w)
    {
```

```
24          e[hcnt].u = u,e[hcnt].v = v,e[hcnt].w = w;
25          e[hcnt].nxt = head[u];head[u] = hcnt++;
26          e[hcnt].u = v,e[hcnt].v = u,e[hcnt].w = 0;
27          e[hcnt].nxt = head[v];head[v] = hcnt++;
28      }
29      int bfs()
30      {
31          rep(i,0,N)
32          {
33              dep[i] = inf;
34          }
35          queue<int> q;
36          q.push(S); dep[S] = 0;
37          while(!q.empty())
38          {
39              int u = q.front();q.pop();
40              for(int i = head[u];~i;i = e[i].nxt)
41              {
42                  int v = e[i].v,w = e[i].w;
43                  if(w > 0 && dep[u] + 1 < dep[v])
44                  {
45                      dep[v] = dep[u] + 1;
46                      if(v == T)
47                      {
48                          return 1;
49                      }
50                      q.push(v);
51                  }
52              }
53          }
54          return dep[T] != inf;
55      }
56      int dfs(int s,int mw)
57      {
58          if(s == T) return mw;
59          for(int i = cur[s];~i;i=e[i].nxt)
60          {
```

```
            cur[s] = i;
            int v = e[i].v,w=e[i].w;
            if(w <= 0 || dep[v] != dep[s] + 1)
            {
                continue;
            }
            int cw = dfs(v,min(w,mw));
            if(cw <= 0)
                continue;
            e[i].w -= cw;
            e[i^1].w += cw;
            return cw;
        }
        return 0;
    }
    ll dinic()
    {
        ll res = 0;
        while(bfs())
        {
            rep(i,0,N)
            {
                cur[i] = head[i];
            }
            while(int d = dfs(S,inf))
            {
                res += 1ll * d;
            }
        }
        return res;
    }
} Dinic;
```

- MCMF1

```
namespace mincostflow {
    const int INF=0x3f3f3f3f;
```

```
3     struct node {
4         int to; int cap,cost; int rev;
5         node(int t=0,int c=0,int _c=0,int n=0):
6             to(t),cap(c),cost(_c),rev(n) {};
7     }; vector<node> edge[maxn];
8     void addedge(int from,int to,int cap,int cost) {
9         edge[from].push_back(node(to,cap,cost,edge[to].size()));
10        edge[to].push_back(node(from,0,-cost,edge[from].size()-1));
11    }
12    int dis[maxn];
13    bool mark[maxn];
14    void spfa(int s,int t,int n) {
15        memset(dis+1,0x3f,n*sizeof(int));
16        memset(mark+1,0,n*sizeof(bool));
17        static int Q[maxn],ST,ED;
18        dis[s]=0; ST=ED=0; Q[ED++]=s;
19        while (ST!=ED) {
20            int v=Q[ST]; mark[v]=0;
21            if ((++ST)==maxn) ST=0;
22            for (node &e:edge[v]) {
23                if (e.cap>0&&dis[e.to]>dis[v]+e.cost) {
24                    dis[e.to]=dis[v]+e.cost;
25                    if (!mark[e.to]) {
26                        if (ST==ED||dis[Q[ST]]<=dis[e.to]) {
27                            Q[ED]=e.to,mark[e.to]=1;
28                            if ((++ED)==maxn) ED=0;
29                        } else {
30                            if ((--ST)<0) ST+=maxn;
31                            Q[ST]=e.to,mark[e.to]=1;
32                        }
33                    }
34                }
35            }
36        }
37    } int cur[maxn];
38    int dfs(int x,int t,int flow) {
39        if (x==t||!flow) return flow;
```

```
        int ret=0; mark[x]=1;
        for (int &i=cur[x];i<(int)edge[x].size();i++) {
            node &e=edge[x][i];
            if (!mark[e.to]&&e.cap) {
                if (dis[x]+e.cost==dis[e.to]) {
                    int f=dfs(e.to,t,min(flow,e.cap));
                    e.cap-=f; edge[e.to][e.rev].cap+=f;
                    ret+=f; flow-=f;
                    if (flow==0) break;
                }
            }
        } mark[x]=0;
        return ret;
    }
    pair<int,int> min_costflow(int s,int t,int n) {
        int ret=0,ans=0;
        int flow = INF;
        while (flow) {
            spfa(s,t,n); if (dis[t]==INF) break;
            memset(cur+1,0,n*sizeof(int));
            int len=dis[t],f;
            while ((f=dfs(s,t,flow))>0)
                ret+=f,ans+=len*f,flow-=f;
        } return make_pair(ret,ans);//最大流，最小费用
    }
    void init(int n) {
        int i; for (int i = 1; i <= n; i++) edge[i].clear();
    }
}
```

- MCMF2

```
const int maxn = 2e4+10;
namespace MCMF {
    const int inf=0x3f3f3f3f;
    struct Edge {
        int to; int cap,cost; int rev;
```

```
6         Edge(int t=0,int c=0,int _c=0,int n=0):
7             to(t),cap(c),cost(_c),rev(n) {};
8     };
9     vector<Edge> edge[maxn];
10    void adde(int from,int to,int cap,int cost)
11    {
12        edge[from].push_back(Edge(to,cap,cost,edge[to].size()));
13        edge[to].push_back(Edge(from,0,-cost,edge[from].size()-1));
14    }
15
16    int dis[maxn];
17    bool mark[maxn];
18
19    void spfa(int s,int t,int n)
20    {
21        memset(dis,0x3f,sizeof dis);
22        memset(mark,0,sizeof mark);
23        static int Q[maxn],ST,ED;
24        dis[s]=0; ST=ED=0; Q[ED++]=s;
25        while (ST!=ED)
26        {
27            int v=Q[ST]; mark[v]=0;
28            if ((++ST)==maxn) ST=0;
29            for (Edge &e:edge[v])
30            {
31                if (e.cap>0&&dis[e.to]>dis[v]+e.cost)
32                {
33                    dis[e.to]=dis[v]+e.cost;
34                    if (!mark[e.to])
35                    {
36                        if (ST==ED||dis[Q[ST]]<=dis[e.to])
37                        {
38                            Q[ED]=e.to,mark[e.to]=1;
39                            if ((++ED)==maxn) ED=0;
40                        }
41                        else
42                        {
```

```
43                      if ((--ST)<0) ST+=maxn;
44                      Q[ST]=e.to,mark[e.to]=1;
45                  }
46              }
47          }
48      }
49  }
50  int cur[maxn];
51  int dfs(int x,int t,int flow)
52  {
53      if (x==t||!flow) return flow;
54      int ret=0; mark[x]=1;
55      for (int &i=cur[x];i<(int)edge[x].size();i++)
56      {
57          Edge &e=edge[x][i];
58          if (!mark[e.to]&&e.cap)
59          {
60              if (dis[x]+e.cost==dis[e.to])
61              {
62                  int f=dfs(e.to,t,min(flow,e.cap));
63                  e.cap-=f; edge[e.to][e.rev].cap+=f;
64                  ret+=f; flow-=f;
65                  if (flow==0) break;
66              }
67          }
68      }
69      mark[x]=0;
70      return ret;
71  }
72  pair<int,ll> mc(int s,int t,int n)
73  {
74      int ret=0;
75      ll ans=0;
76      int flow = inf;
77      while(flow)
78      {
79
```

```
            spfa(s,t,n); if (dis[t]==inf) break;
            memset(cur,0,sizeof cur);
            int len=dis[t],f;
            while ((f=dfs(s,t,flow))>0)
                ret+=f,ans+=(ll)len*(ll)f,flow-=f;
        }
        return make_pair(ret,ans);//最大流，最小费用
    }
    void init(int n)
    {
        for(int i = 1; i <= n; i++) edge[i].clear();
    }
}
```

4 数学

4.1 高斯消元

```
const int N = 307;
int x[N],a[N][N];// x[N]解集，a[N][N]系数
bool free_x[N];
int gcd(int a,int b){return b ? gcd(b,a % b) : a;}
int lcm(int a,int b){return a / gcd(b,a % b) * b;}
int Gauss(int equ,int var)//equ个方程，var个变元
{
    int
        free_x_num,i,j,row,max_r,col;//row表示行,col表示列,max_r表示列最大的行,free_x_num变元数量
    int free_index,LCM,ta,tb,temp;// free_index变元下标
    for(i = 0;i <= var;++i){
        x[i] = 0;
        free_x[i] = true;//第i个元素是否是变元
    }
    for(row = 0,col = 0;row < equ && col < var;++row,++col){
        max_r = row;
        //找到col最大的行，进行交换（除法时减小误差）
        for(i = row + 1;i < equ;++i) if(abs(a[i][col]) > abs(a[max_r][col])) max_r =
```

```
            i;
        //与第row行交换
        if(max_r != row) for(j = row;j < var + 1;++j) swap(a[row][j],a[max_r][j]);
        if(a[row][col]==0){
            //说明该col列第row行以下全是0了，则处理当前行的下一列.
            row--;
            continue;
        }
        for(i = row + 1;i < equ;++i)//枚举被删行
            if(a[i][col]){
                LCM = lcm(abs(a[i][col]),abs(a[row][col]));
                ta = LCM / abs(a[i][col]);
                tb = LCM / abs(a[row][col]);
                if(a[i][col] * a[row][col] < 0)tb = -tb;//异号的情况是相加
                for(j = col;j < var + 1;++j)
                    a[i][j] = a[i][j] * ta - a[row][j] * tb;
            }
        /*求解小数解,防止溢出
        for(int i = row + 1; i < equ; ++i)
            if(fabs(a[i][col]) > eps){
                double t1 = a[i][col]/a[row][col];
                for(int j = col; j <= var;++j) a[i][j] -= a[row][j] * t1;
            }*/
    }
    for (i = row;i < equ;++i) if(a[i][col]) return -1; // 无解
    if (row < var){// 多解
        for(i = row - 1;i >= 0;--i){
            free_x_num = 0;
            for (j = 0;j < var;++j)
                if(a[i][j] && free_x[j]) free_x_num++,free_index = j;
            if (free_x_num > 1) continue; // 无法求解出确定的变元.
            temp = a[i][var];
            for (j = 0;j < var;++j) if (a[i][j] && j != free_index) temp -= a[i][j]
                * x[j];
            x[free_index] = temp / a[i][free_index]; //求出该变元.
            free_x[free_index] = 0; //该变元是确定的.
        }
```

```
53          return var - row; //自由变元有 var - row 个.
54      }
55      for (i = var - 1; i >= 0;--i){// 唯一解
56          temp = a[i][var];
57          for (j = i + 1;j < var;++j)
58              if (a[i][j]) temp -= a[i][j] * x[j];
59          if (temp % a[i][i]) return -2; // 说明有浮点数解,但无整数解.
60          x[i] = temp / a[i][i];
61      }
62      return 0;
63  }
```

4.1.1 fft

```
1   namespace fft
2   {
3       struct num
4       {
5           double x,y;
6           num() {x=y=0;}
7           num(double x,double y):x(x),y(y){}
8       };
9       inline num operator+(num a,num b) {return num(a.x+b.x,a.y+b.y);}
10      inline num operator-(num a,num b) {return num(a.x-b.x,a.y-b.y);}
11      inline num operator*(num a,num b) {return num(a.x*b.x-a.y*b.y,a.x*b.y+a.y*b.x);}
12      inline num conj(num a) {return num(a.x,-a.y);}
13
14      int base=1;
15      vector<num> roots={{0,0},{1,0}};
16      vector<int> rev={0,1};
17      const double PI=acosl(-1.0);
18
19      void ensure_base(int nbase)
20      {
21          if(nbase<=base) return;
22          rev.resize(1<<nbase);
23          for(int i=0;i<(1<<nbase);i++)
```

```
24          rev[i]=(rev[i>>1]>>1)+((i&1)<<(nbase-1));
25      roots.resize(1<<nbase);
26      while(base<nbase)
27      {
28          double angle=2*PI/(1<<(base+1));
29          for(int i=1<<(base-1);i<(1<<base);i++)
30          {
31              roots[i<<1]=roots[i];
32              double angle_i=angle*(2*i+1-(1<<base));
33              roots[(i<<1)+1]=num(cos(angle_i),sin(angle_i));
34          }
35          base++;
36      }
37  }
38
39  void fft(vector<num> &a,int n=-1)
40  {
41      if(n==-1) n=a.size();
42      assert((n&(n-1))==0);
43      int zeros=__builtin_ctz(n);
44      ensure_base(zeros);
45      int shift=base-zeros;
46      for(int i=0;i<n;i++)
47          if(i<(rev[i]>>shift))
48              swap(a[i],a[rev[i]>>shift]);
49      for(int k=1;k<n;k<<=1)
50      {
51          for(int i=0;i<n;i+=2*k)
52          {
53              for(int j=0;j<k;j++)
54              {
55                  num z=a[i+j+k]*roots[j+k];
56                  a[i+j+k]=a[i+j]-z;
57                  a[i+j]=a[i+j]+z;
58              }
59          }
60      }
```

```
61      }
62
63      vector<num> fa,fb;
64
65      vector<int> multiply(vector<int> &a, vector<int> &b)
66      {
67          int need=a.size()+b.size()-1;
68          int nbase=0;
69          while((1<<nbase)<need) nbase++;
70          ensure_base(nbase);
71          int sz=1<<nbase;
72          if(sz>(int)fa.size()) fa.resize(sz);
73          for(int i=0;i<sz;i++)
74          {
75              int x=(i<(int)a.size()?a[i]:0);
76              int y=(i<(int)b.size()?b[i]:0);
77              fa[i]=num(x,y);
78          }
79          fft(fa,sz);
80          num r(0,-0.25/sz);
81          for(int i=0;i<=(sz>>1);i++)
82          {
83              int j=(sz-i)&(sz-1);
84              num z=(fa[j]*fa[j]-conj(fa[i]*fa[i]))*r;
85              if(i!=j) fa[j]=(fa[i]*fa[i]-conj(fa[j]*fa[j]))*r;
86              fa[i]=z;
87          }
88          fft(fa,sz);
89          vector<int> res(need);
90          for(int i=0;i<need;i++) res[i]=fa[i].x+0.5;
91          return res;
92      }
93
94      vector<int> multiply_mod(vector<int> &a,vector<int> &b,int m,int eq=0)
95      {
96          int need=a.size()+b.size()-1;
97          int nbase=0;
```

```
98          while((1<<nbase)<need) nbase++;
99          ensure_base(nbase);
100         int sz=1<<nbase;
101         if(sz>(int)fa.size()) fa.resize(sz);
102         for(int i=0;i<(int)a.size();i++)
103         {
104             int x=(a[i]%m+m)%m;
105             fa[i]=num(x&((1<<15)-1),x>>15);
106         }
107         fill(fa.begin()+a.size(),fa.begin()+sz,num{0,0});
108         fft(fa,sz);
109         if(sz>(int)fb.size()) fb.resize(sz);
110         if(eq) copy(fa.begin(),fa.begin()+sz,fb.begin());
111         else
112         {
113             for(int i=0;i<(int)b.size();i++)
114             {
115                 int x=(b[i]%m+m)%m;
116                 fb[i]=num(x&((1<<15)-1),x>>15);
117             }
118             fill(fb.begin()+b.size(),fb.begin()+sz,num{0,0});
119             fft(fb,sz);
120         }
121         double ratio=0.25/sz;
122         num r2(0,-1),r3(ratio,0),r4(0,-ratio),r5(0,1);
123         for(int i=0;i<=(sz>>1);i++)
124         {
125             int j=(sz-i)&(sz-1);
126             num a1=(fa[i]+conj(fa[j]));
127             num a2=(fa[i]-conj(fa[j]))*r2;
128             num b1=(fb[i]+conj(fb[j]))*r3;
129             num b2=(fb[i]-conj(fb[j]))*r4;
130             if(i!=j)
131             {
132                 num c1=(fa[j]+conj(fa[i]));
133                 num c2=(fa[j]-conj(fa[i]))*r2;
134                 num d1=(fb[j]+conj(fb[i]))*r3;
```

```
135                 num d2=(fb[j]-conj(fb[i]))*r4;
136                 fa[i]=c1*d1+c2*d2*r5;
137                 fb[i]=c1*d2+c2*d1;
138             }
139             fa[j]=a1*b1+a2*b2*r5;
140             fb[j]=a1*b2+a2*b1;
141         }
142         fft(fa,sz);fft(fb,sz);
143         vector<int> res(need);
144         for(int i=0;i<need;i++)
145         {
146             ll aa=fa[i].x+0.5;
147             ll bb=fb[i].x+0.5;
148             ll cc=fa[i].y+0.5;
149             res[i]=(aa+((bb%m)<<15)+((cc%m)<<30))%m;
150         }
151         return res;
152     }
153     vector<int> square_mod(vector<int> &a,int m)
154     {
155         return multiply_mod(a,a,m,1);
156     }
157 };
```

4.2 线性基

HDU6579 [l,r] 最大异或和

```
1  struct LB
2  {
3      ll p[33];
4      int g[33];
5      void ins(ll x,int pos)
6      {
7          per(i,0,30)
8          {
9              if((x>>i) & 1)
```

```
10          {
11              if(p[i])
12              {
13                  if(g[i] <= pos)
14                  {
15                      x ^= p[i];
16                      p[i] ^= x;
17                      swap(g[i],pos);
18                  }
19                  else
20                      x ^= p[i];
21              }
22              else
23              {
24                  p[i] = x;
25                  g[i] = pos;
26                  break;
27              }
28          }
29       }
30   }
31   ll query(int l)
32   {
33       ll res = 0;
34       per(i,0,30)
35       {
36          if(g[i] >= l)
37          {
38              res = max(res,res^p[i]);
39          }
40       }
41       return res;
42   }
43 } base[maxn];
44
45 int n,m;
46
```

```
47 int gao(int x,int lastans)
48 {
49     return (x^lastans) % n + 1;
50 }
51
52 int T;
53 int x;
54
55 int main(int argc, char const *argv[])
56 {
57     // ios_base::sync_with_stdio(false), cin.tie(0), cout.tie(0);
58     scanf("%d",&T);
59     while(T--)
60     {
61         scanf("%d%d",&n,&m);
62         rep(i,1,n+1)
63         {
64             scanf("%d",&x);
65             base[i] = base[i-1];
66             base[i].ins(x,i);
67         }
68         ll ans = 0;
69         int l,r;
70         while(m--)
71         {
72             int op;
73             scanf("%d",&op);
74             if(!op)
75             {
76                 scanf("%d%d",&l,&r);
77                 l = gao(l,ans);r=gao(r,ans);
78                 if(l>r) swap(l,r);
79                 ans = base[r].query(l);
80                 printf("%lld\n",ans);
81             }
82             else
83             {
```

```
84                  n++;
85                  scanf("%d",&l);
86                  base[n] = base[n-1];
87                  base[n].ins(l^ans,n);
88              }
89          }
90      }
91      return 0;
92  }
```

5 计算几何

5.1 处理平面内所有直线围成的所有多边形

```
1   const int MAXN=1e6+10;
2   const double eps=1e-8;
3   const double pi=acos(-1.0);
4   const ll INF=0x3f3f3f3f3f3f3f3f;
5
6   inline int dcmp(double x){
7       if(fabs(x)<eps) return 0;
8       return (x>0? 1: -1);
9   }
10
11  inline double sqr(double x){ return x*x; }
12
13  struct Point{
14      double x,y;
15      Point(){ x=0,y=0; }
16      Point(double _x,double _y):x(_x),y(_y){}
17      void input(){ scanf("%lf%lf",&x,&y); }
18      void output(){ printf("%.2f %.2f\n",x,y); }
19      friend istream &operator >>(istream &os,Point &b){
20          os>>b.x>>b.y;
21          return os;
22      }
```

```
23      friend ostream &operator <<(ostream &os,Point &b){
24          os<<b.x<<' '<<b.y;
25          return os;
26      }
27      bool operator ==(const Point &b)const{
28          return (dcmp(x-b.x)==0&&dcmp(y-b.y)==0);
29      }
30      bool operator !=(const Point &b)const{
31          return !((dcmp(x-b.x)==0&&dcmp(y-b.y)==0));
32      }
33      bool operator <(const Point &b)const{
34          return (dcmp(x-b.x)==0? dcmp(y-b.y)<0 : x<b.x);
35      }
36      double operator ^(const Point &b)const{ //叉积
37          return x*b.y-y*b.x;
38      }
39      double operator *(const Point &b)const{ //点积
40          return x*b.x+y*b.y;
41      }
42      Point operator +(const Point &b)const{
43          return Point(x+b.x,y+b.y);
44      }
45      Point operator -(const Point &b)const{
46          return Point(x-b.x,y-b.y);
47      }
48      Point operator *(double a){
49          return Point(x*a,y*a);
50      }
51      Point operator /(double a){
52          return Point(x/a,y/a);
53      }
54      double len2(){ //长度平方
55          return sqr(x)+sqr(y);
56      }
57      double len(){ //长度
58          return sqrt(len2());
59      }
```

```
60      double polar(){ //向量的极角
61          return atan2(y,x); //返回与x轴正向夹角(-pi~pi]
62      }
63      Point change_len(double r){ //转化为长度为r的向量
64          double l=len();
65          if(dcmp(l)==0) return *this; //零向量
66          return Point(x*r/l,y*r/l);
67      }
68      Point rotate_left(){ //逆时针旋转90度
69          return Point(-y,x);
70      }
71      Point rotate_right(){ //顺时针旋转90度
72          return Point(y,-x);
73      }
74      Point rotate(Point p,double ang){ //绕点p逆时针旋转ang度
75          Point v=(*this)-p;
76          double c=cos(ang),s=sin(ang);
77          return Point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
78      }
79      Point normal(){  //单位化，逆时针旋转90°
80          return Point(-y/len(),x/len());
81      }
82  };
83
84  inline double cross(Point a,Point b){ //叉积
85      return a.x*b.y-a.y*b.x;
86  }
87
88  inline double dot(Point a,Point b){ //点积
89      return a.x*b.x+a.y*b.y;
90  }
91
92
93  double rad(Point a,Point b){ //两个向量的夹角
94      return fabs(atan2(fabs(cross(a,b)),dot(a,b)));
95  }
96
```

```
97  bool is_parallel(Point a,Point b){ //判断向量是否平行
98      double p=rad(a,b);
99      return dcmp(p)==0||dcmp(p-pi)==0;
100 }
101
102 struct Line{
103     Point s,e;
104     Line(){}
105     Line(Point _s,Point _e):s(_s),e(_e){} //两点确定直线
106     Line(Point p,double ang){ //一个点和斜率(弧度制)确定直线
107         s=p;
108         if(dcmp(ang-pi/2)==0){
109             e=s+Point(0,1);
110         }
111         else{
112             e=s+Point(1,tan(ang));
113         }
114     }
115     Line(double a,double b,double c){ //ax+by+c=0
116         if(dcmp(a)==0){
117             s=Point(0,-c/b);
118             e=Point(1,-c/b);
119         }
120         else if(dcmp(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(e,s);
135     }
136     double polar(){ //极角
137         return atan2(e.y-s.y,e.x-s.x); //返回与x轴正向夹角(-pi~pi]
138     }
139     double angle(){  //倾斜角
140         double k=atan2(e.y-s.y,e.x-s.x);
141         if(dcmp(k)<0) k+=pi;
142         if(dcmp(k-pi)==0) k-=pi;
143         return k;
144     }
145     Point operator &(const Line &b)const{ //求两直线交点
146         Point res=s;
147         double t=((s-b.s)^(b.s-b.e))/((s-e)^(b.s-b.e));
148         res.x+=(e.x-s.x)*t;
149         res.y+=(e.y-s.y)*t;
150         return res;
151     }
152 };
153
154 double polygon_area(vector<Point> p){ //多边形的有向面积，加上绝对值就是面积
    正值表示输入点按照逆时针 否则为顺时针
155     int n=p.size(); double area=0;
156     for(int i=1;i<n-1;i++) area+=cross(p[i]-p[0],p[i+1]-p[0]);
157     return fabs(area/2);
158 }
159
160 struct PSLG{  //平面直线图 处理平面内所有直线围成的所有多边形 传入直线交点之间的每条线段
161     struct Edge{
162         int from,to;
163         double ang;
164         Edge(){ ang=from=to=0; }
165         Edge(int s,int t,double a){ from=s,to=t,ang=a; }
166     };
167     int n,m,face_cnt; //平面个数 包括外面最大的多边形
168     double area[MAXN]; //每个多边形面积
169     Point point[MAXN]; //平面内所有的点
```

```
170     vector<Edge>edge;
171     vector<int>G[MAXN];
172     vector<vector<Point> >face;
173     int vis[2*MAXN],left[2*MAXN],pre[2*MAXN]; //left表示这条边的左侧属于哪个面
174     void Init(){
175         face.clear();
176         edge.clear();
177         for(int i=0;i<n;i++) G[i].clear();
178         n=m=0;
179     }
180     PSLG(){ Init(); }
181     void AddEdge(int from, int to){       //需要建立反向边帮助寻找下一条边
182        edge.pb(Edge(from,to,(point[to]-point[from]).polar()));
183        edge.pb(Edge(to,from,(point[from]-point[to]).polar()));
184        m=edge.size();
185        G[from].pb(m-2);
186        G[to].pb(m-1);
187     }
188     void Build(){
189         for(int u=0;u<n;u++){
190             int d=G[u].size();
191             for(int i=0;i<d;i++)
192                 for(int j=i+1;j<d;j++)
193                     if(edge[G[u][i]].ang>edge[G[u][j]].ang)
194                         swap(G[u][i],G[u][j]);
195             for(int i=0;i<d;i++) pre[G[u][(i+1)%d]]=G[u][i];
                 //从u出发的i条边顺时针旋转的第一条边是pre[i]
196         }
197         face_cnt=0; memset(vis,0,sizeof(vis));
198         for(int u=0;u<n;u++){
199             for(int i=0;i<G[u].size();i++){
200                 int e=G[u][i];
201                 if(!vis[e]){
202                     face_cnt++;
203                     vector<Point> polygon;
204                     while(1){
205                         vis[e]=1;
```

```
206                     left[e]=face_cnt;
207                     int from=edge[e].from;
208                     polygon.pb(point[from]);
209                     e=pre[e^1];       //逆时针旋转最多的一条边即为顺时针转动的第一条边
210                     if(e==G[u][i]) break;
211                 }
212                 face.pb(polygon);
213             }
214         }
215     }
216     for(int i=0;i<face_cnt;i++) area[i]=polygon_area(face[i]);
217 }
218 vector<pair<double,int> >tmp[MAXN];
219 void Insert(Line *line,int m){
220     for(int i=0;i<m;i++){
221         for(int j=i+1;j<m;j++){
222             if(!is_parallel(line[i].e-line[i].s,line[j].e-line[j].s)){
223                 Point inter=line[i]&line[j];
224                 point[n++]=inter;
225                 tmp[i].pb({dot(inter-line[i].s,line[i].e-line[i].s),n-1});
226                 tmp[j].pb({dot(inter-line[j].s,line[j].e-line[j].s),n-1});
227             }
228         }
229         sort(tmp[i].begin(),tmp[i].end());
230         for(int j=1;j<tmp[i].size();j++) AddEdge(tmp[i][j-1].se,tmp[i][j].se);
231     }
232     Build();
233 }
234 }pslg;
235
236 Line line[MAXN];
237
238 int main(void){
239     int n; scanf("%d",&n);
240     for(int i=0;i<n;i++) line[i].input();
241     pslg.Insert(line,n);
242     sort(pslg.area,pslg.area+pslg.face_cnt);
```

```
243     printf("%d %.6f
            %.6f\n",pslg.face_cnt-1,pslg.area[pslg.face_cnt-2],pslg.area[0]);
244     int q; scanf("%d",&q);
245     while(q--){
246         int p; scanf("%d",&p);
247         if(p>=pslg.face_cnt) puts("Invalid question");
248         else   printf("%.6f\n",pslg.area[pslg.face_cnt-p-1]);
249     }
250     return 0;
251 }
```

6 字符串

6.1 kmp

- border

```
1  void get_fail(int f[],char s[])
2  {
3      int j = f[0] = 0;
4      int n = strlen(s);
5      rep(i,1,n)
6      {
7          while(j && s[i] != s[j]) j = f[j-1];
8          f[i] = j += s[i] == s[j];
9      }
10 }
```

- kmp

```
1  void kmp(int f[],char p[],char s[])
2  {
3      int n = strlen(s);
4      int m = strlen(p);
5      int j = 0;
6      rep(i,0,n)
```

```
    {
        while(j && s[i] != p[j]) j = f[j-1];
        if(s[i] == p[j]) j++;
        if(j == m)
        {
            cout << i - j + 2 << endl;
            j = f[j-1];
        }
    }
}
```

6.2 SA

```
const int N=4e5+100;
const int maxn = 2e5+100;
const int inf=1e9+9;

namespace SA {
    char s[N];
    int sa[N],x[N],y[N],hep[N],height[N],n,m;
    void init()
    {
        n = 0;
    }
    void add(char c)
    {
        // c -= 'a';
        n++;
        s[n]=c;
    }
    void Sort() {
        for(int i=0;i<=m;++i) hep[i]=0;
        for(int i=1;i<=n;++i) ++hep[x[i]];
        for(int i=1;i<=m;++i) hep[i]+=hep[i-1];
        for(int i=n;i>=1;--i) sa[hep[x[y[i]]]--]=y[i];
    }
```

```
24    void Pre_sa() {
25        for(int i=1;i<=n;++i) x[i]=s[i],y[i]=i;
26        m=223;Sort();
27        for(int w=1,p=0;m=p,p<n;w<<=1) {
28            p=0;
29            for(int i=1;i<=w;++i) y[++p]=n-w+i;
30            for(int i=1;i<=n;++i) if(sa[i]>w) y[++p]=sa[i]-w;
31            Sort(),swap(x,y),x[sa[1]]=p=1;
32            for(int i=2;i<=n;++i)
33                x[sa[i]]=(y[sa[i]]==y[sa[i-1]]&&y[sa[i]+w]==y[sa[i-1]+w])?p:++p;
34        }return;
35    }
36    ll Pre_height() {
37        for(int i=1;i<=n;++i) x[sa[i]]=i;
38        int k=0,res=0;
39        for(int i=1;i<=n;++i) {
40            k-=k>0;
41            int j=sa[x[i]-1];
42            while(i+k<=n&&j+k<=n&&s[i+k]==s[j+k]) ++k;
43            height[x[i]]=k,res+=k;
44        }return res;//直接返回height数组的和
45    }
46    ll solve()
47    {
48        /**
49        给你一个长为N的字符串，求不同的子串的个数?
50        对于一个后缀sa[i]，它产生了n-sa[i]个前缀，减去height[i]个相同的前缀(与前一个比较),
51        则产生了n-sa[i]-height[i]个子串。累加后即结果。
52        */
53        ll ans = 0;
54        for (int i = 1; i <= n; i++)
55        {
56            ans += n + 1 - sa[i] - height[i];
57        }
58        return ans;
59    }
60    ll gao()
```

```
61      {
62          Pre_sa();
63          Pre_height();
64          return solve();
65      }
66  }
```

6.3 回文树 1

```
1   struct Pal
2   {
3       int ch[maxn][26],f[maxn],len[maxn],s[maxn];
4       int cnt[maxn];
5       int num[maxn];
6       int last,sz,n;
7
8       int newnode(int x)
9       {
10          memset(ch[sz],0,sizeof(ch[sz]));
11          cnt[sz] = num[sz] = 0, len[sz] = x;
12          return sz++;
13      }
14      void init()
15      {
16          sz = 0;
17          newnode(0),newnode(-1);
18          last = n = 0,s[0] = -1,f[0] = 1;
19      }
20
21      int get_fail(int u)
22      {
23          while(s[n - len[u] - 1] != s[n])
24              u = f[u];
25          return u;
26      }
27
```

```
    void add(int c)
    {
        c -= 'a';
        s[++n] = c;
        int u = get_fail(last);
        if(!ch[u][c])
        {
            int np = newnode(len[u] + 2);
            f[np] = ch[get_fail(f[u])][c];
            num[np] = num[f[np]] + 1;
            ch[u][c] = np;
        }
        last = ch[u][c];
        cnt[last]++;
    }

    void count()
    {
        for(int i = sz - 1;~i;i--)
        cnt[f[i]] += cnt[i];
    }
} pa;
```

6.4 回文树 2

```
struct Palindromic_Tree {
    int son[N][26]; //转移边
    int fail[N]; //fail 指针
    int cnt[N]; //当前节点表示的回文串在原串中出现了多少次
    int num[N]; //当前节点 fail 可以向前跳多少次
    int len[N]; //当前节点表示的回文串的长度
    int S[N]; //插入的字符串
    int last; //最后一次访问到的节点，类似 SAM
    int n; //插入的字符串长度
    long long p; //自动机的总状态数

```

```
12  int newnode(int l) {
13      memset(son[p], 0, sizeof(son[p]));
14      cnt[p] = 0;
15      num[p] = 0;
16      len[p] = l;
17      return p++;
18  }
19
20  void init() {
21      p = 0;
22      newnode(0);
23      newnode(-1);
24      last = 0;
25      n = 0;
26      S[n] = -1;
27      fail[0] = 1;
28  }
29
30  int get_fail(int x) {
31      while (S[n - len[x] - 1] != S[n]) x = fail[x];
32      return x;
33  }
34
35  void add(int c) {
36      c -= 'a';
37      S[++n] = c;
38      int cur = get_fail(last); //通过上一次访问的位置去扩展
39      if (!son[cur][c]) { //如果没有对应的节点添加一个新节点
40          int now = newnode(len[cur] + 2);
41          fail[now] = son[get_fail(fail[cur])][c]; //通过当前节点的 fail
                去扩展出新的 fail
42          son[cur][c] = now;
43          num[now] = num[fail[now]] + 1; //记录 fail 跳多少次
44      }
45      last = son[cur][c];
46      cnt[last]++; //表示当前节点访问了一次
47  }
```

```
48      void count() {
49          //如果某个节点出现一次，那么他的 fail 也一定会出现一次，并且在插入的时候没有计数
50          for (int i = p - 1; i >= 0; i--) cnt[fail[i]] += cnt[i];
51      }
52  } AUT;
```

7 杂项

7.1 退火

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   const int maxn = 1e5 + 10;
6   const double eps = 1e-8;
7   const double delta = 0.98;
8   const double inf = 1e18;
9
10  struct Point { double x, y; } p[maxn];
11
12  double dis(Point A, Point B) { return sqrt((A.x - B.x) * (A.x - B.x) + (A.y - B.y)
        * (A.y - B.y)); };
13
14  double Simulate_Annea(int n)
15  {
16      Point S;
17      S.x = S.y = 0;
18      double t = 1000;
19      double res = inf;
20      while(t > eps)
21      {
22          int k = 0;
23          for(int i = 0; i < n; i ++) if(dis(S, p[i]) > dis(S, p[k])) k = i;
24          double d = dis(S, p[k]);
25          res = min(res, d);
```

```
        S.x += (p[k].x - S.x) / d * t;
        S.y += (p[k].y - S.y) / d * t;
        t *= delta;
    }
    return res;
}

int main()
{
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; i ++) scanf("%lf%lf", &p[i].x, &p[i].y);
    printf("%.3f\n", Simulate_Annea(n));
    return 0;
}
```

8 DP

8.1 背包

8.1.1 01 背包

- normal $f_i = max(f_i, f_i v + w)$ 需要按照 i 从大到小的顺序更新，确保每个物品只会选一次

```
memset(dp,0xcf,sizeof dp);
dp[0] = 0;
rep(i,0,n)
{
    cin >> v[i] >> w[i];
    per(j,v[i],m+1)
    {
        dp[j] = max(dp[j],dp[j-v[i]]+w[i]);
    }
}
```

- 计数不超过 m 的方案数 $f_i += f_i v$

- 删除加入物品的顺序不影响结果，假设被删除的物品是最后一次加入的，那么倒过来还原即可。$f_i- = f_iv$ 需要按照 i 从小到大的顺序更新. 给定 ai 和 bi，表示第 i 个商店有 ai 个商品可以买，单价为 bi 元，给出 m 个询问，问用 c 元在 l~r 商店买东西的方案数 一种物品的背包可以看成 $\sum_{i=0}^{a} x^{ib} = \frac{1-x^{(a+1)b}}{1-x^b}$，所以可以先用 (a+1)b 去做一个 01 背包 (系数为负)，再除以一个 x^b 的 (系数为负)01 背包。从生成函数来看，$\frac{1}{1-x^b} = \sum_{i=0}^{\infty} x^{bi}$，即做一遍完全背包就可以等效

然后对可逆背包的预处理，由于 $\frac{1-x^b}{1-x^{(a+1)*b}} = (1-x^b) * \sum_{i=0}^{\infty} x^{i*(a+1)b}$，于是反过来对 x^b 做 01 背包，对 (a+1)b 做完全背包就可以

```
int n,m;
int a[maxn*10],b[maxn*10],f[maxn*10][maxn],g[maxn*10][maxn];

void gao(int *dp,int w)
{
    per(i,w,maxn)
    {
        dp[i] = (dp[i] - dp[i-w]+mod)%mod;
    }
}

void gao2(int *dp,int w)
{
    rep(i,w,maxn)
    {
        dp[i] = (dp[i] + dp[i-w]+mod)%mod;
    }
}

int main(int argc, char const *argv[])
{
    ios_base::sync_with_stdio(false), cin.tie(0);
    // cout.tie(0);
    int T,cas;
    cin >> T;
    cas = 1;
    f[0][0] = 1;
    rep(i,0,maxn)
```

```
29          g[0][i] = 1;
30      while(T--)
31      {
32          prr(cas++);
33          cin >> n >> m;
34          rep(i,1,n+1)
35          {
36              cin >> a[i] >> b[i];
37              a[i] = (a[i] + 1) * b[i];
38          }
39          rep(i,1,n+1)
40          {
41              memcpy(f[i],f[i-1],sizeof(f[i]));
42              memcpy(g[i],g[i-1],sizeof(g[i]));
43              gao(f[i],a[i]);gao2(f[i],b[i]);
44              gao(g[i],b[i]);gao2(g[i],a[i]);
45          }
46          int ans = 0;
47          rep(i,0,m)
48          {
49              int l,r,c;
50              cin >> l >> r >> c;
51              l = (l+ans)%n+1;
52              r = (r+ans)%n+1;
53              if(l > r)
54              {
55                  swap(l,r);
56              }
57              ans = 0;
58              rep(i,0,c+1)
59              {
60                  ans = (ans + 1ll*f[r][i]*g[l-1][c-i])%mod;
61              }
62              printf("%d\n",ans);
63          }
64      }
65      return 0;
```

```
}
```

8.1.2 完全背包

- normal 需要按照 i 从小到大的顺序更新，意为要么停止选，要么接着多选一个。

```
dp[0] = 0;
rep(i,0,n)
{
    cin >> v >> w;
    rep(j,v,m+1)
    {
        dp[j] = max(dp[j],dp[j-v]+w);
    }
}
cout << dp[m] << endl;
```

- 计数 same
- 删除 same

8.1.3 多组背包

- 二进制拆分二进制拆分，将一个物品拆成 O(logk) 个 01 背包的物品。eg:10 = 1+2+4+3，可以表示 1 − 10 O(nmlog(k))

```
memset(dp,0,sizeof dp);
    rep(i,0,n)
    {
        cin >> v[i] >> w[i] >> s[i];
    }
    int cnt = 0;
    rep(i,0,n)
    {
        for(int j = 1;j < s[i];j <<= 1)
        {
            ww[cnt] = j * w[i];
            vv[cnt] = j * v[i];
```

```
            s[i] -= j;
            cnt++;
        }
        if(s[i])
        {
            ww[cnt] = s[i] * w[i];
            vv[cnt++] = s[i] * v[i];
        }
    }
    rep(i,0,cnt)
    {
        per(j,vv[i],m+1)
        {
            dp[j] = max(dp[j],dp[j-vv[i]] + ww[i]);
        }
    }
    cout << dp[m] << endl;
```

- 单调队列按%v 的余数分组，每组滑窗求区间最大值 O(nm), 但不见得比上面快

8.1.4 分组背包

n 个物品，每个物品只能选一个，体积为 vi，种类为 ki。求总体积恰好 m 的情况下能拿走物品种类数的最大值。将所有物品按 k 分组状态：$f_{i,j,k,s}$ 表示考虑前 i 组，这一组内考虑了前 j 个物品，总体积为 k，第 i 组物品是否被选择的情况为 s 时，种类数的最大值。

8.1.5 树形依赖背包

以 1 为根的树上有 n 个节点，每个节点有一个物品，体积 vi，价值 wi。选了一个点就必须选它的父亲。求总体积不超过 m 的情况下能拿走物品总价值的最大值。