1 Surroudings

1.1 bashrc

```
1  oj() {
2    g++ -o "/tmp/out" "$1" && "/tmp/out"
3  }
```

2 Data_Structure

2.1 Dynamic_KD_tree

```
1   template<typename T,size_t kd>//有kd個維度
2   struct kd_tree{
3     struct point{
4       T d[kd];
5       T dist(const point &x)const{
6         T ret=0;
7         for(size_t i=0;i<kd;++i)ret+=abs(d[i]-x.d[i]);
8         return ret;
9       }
10      bool operator==(const point &p){
11        for(size_t i=0;i<kd;++i)
12          if(d[i]!=p.d[i])return 0;
13        return 1;
14      }
15      bool operator<(const point &b)const{
16        return d[0]<b.d[0];
17      }
18    };
19  private:
20    struct node{
21      node *l,*r;
22      point pid;
23      int s;
24      node(const point &p):l(0),r(0),pid(p),s(1){}
25      ~node(){delete l,delete r;}
26      void up(){s=(l?l->s:0)+1+(r?r->s:0);}
27    }*root;
28    const double alpha,loga;
29    const T INF;//記得要給INF，表示極大值
30    int maxn;
31    struct __cmp{
32      int sort_id;
33      bool operator()(const node*x,const node*y)const{
34        return operator()(x->pid,y->pid);
35      }
36      bool operator()(const point &x,const point &y)const{
37        if(x.d[sort_id]!=y.d[sort_id])
38          return x.d[sort_id]<y.d[sort_id];
39        for(size_t i=0;i<kd;++i)
40          if(x.d[i]!=y.d[i])return x.d[i]<y.d[i];
41        return 0;
42      }
43    }cmp;
44    int size(node *o){return o?o->s:0;}
45    vector<node*> A;
46    node* build(int k,int l,int r){
47      if(l>r) return 0;
48      if(k==kd) k=0;
49      int mid=(l+r)/2;
50      cmp.sort_id = k;
51      nth_element(A.begin()+l,A.begin()+mid,A.begin()+r+1,cmp);
52      node *ret=A[mid];
53      ret->l = build(k+1,l,mid-1);
54      ret->r = build(k+1,mid+1,r);
55      ret->up();
56      return ret;
57    }
58    bool isbad(node*o){
59      return size(o->l)>alpha*o->s||size(o->r)>alpha*o->s;
60    }
61    void flatten(node *u,typename vector<node*>::iterator &it){
62      if(!u)return;
63      flatten(u->l,it);
64      *it=u;
65      flatten(u->r,++it);
66    }
67    void rebuild(node*&u,int k){
68      if((int)A.size()<u->s)A.resize(u->s);
69      auto it=A.begin();
70      flatten(u,it);
71      u=build(k,0,u->s-1);
72    }
73    bool insert(node*&u,int k,const point &x,int dep){
74      if(!u) return u=new node(x), dep<=0;
75      ++u->s;
76      cmp.sort_id=k;
77      if(insert(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x,dep-1)){
78        if(!isbad(u))return 1;
79        rebuild(u,k);
80      }
81      return 0;
82    }
83    node *findmin(node*o,int k){
84      if(!o)return 0;
85      if(cmp.sort_id==k)return o->l?findmin(o->l,(k+1)%kd):o;
86      node *l=findmin(o->l,(k+1)%kd);
87      node *r=findmin(o->r,(k+1)%kd);
88      if(l&&!r)return cmp(l,o)?l:o;
89      if(!l&&r)return cmp(r,o)?r:o;
90      if(!l&&!r)return o;
91      if(cmp(l,r))return cmp(l,o)?l:o;
92      return cmp(r,o)?r:o;
93    }
94    bool erase(node *&u,int k,const point &x){
95      if(!u)return 0;
96      if(u->pid==x){
97        if(u->r);
98        else if(u->l) u->r=u->l, u->l=0;
99        else return delete(u),u=0, 1;
100       --u->s;
101       cmp.sort_id=k;
102       u->pid=findmin(u->r,(k+1)%kd)->pid;
103       return erase(u->r,(k+1)%kd,u->pid);
104     }
105     cmp.sort_id=k;
106     if(erase(cmp(x,u->pid)?u->l:u->r,(k+1)%kd,x))
107       return --u->s, 1;
108     return 0;
109   }
110   T heuristic(const T h[])const{
111     T ret=0;
112     for(size_t i=0;i<kd;++i)ret+=h[i];
113     return ret;
114   }
115   int qM;
116   priority_queue<pair<T,point>> pQ;
117   void nearest(node *u,int k,const point &x,T *h,T &mndist){
118     if(u==0||heuristic(h)>=mndist)return;
119     T dist=u->pid.dist(x),old=h[k];
120     /*mndist=std::min(mndist,dist);*/
121     if(dist<mndist){
122       pQ.push(std::make_pair(dist,u->pid));
123       if((int)pQ.size()==qM+1)
124         mndist=pQ.top().first,pQ.pop();
125     }
126     if(x.d[k]<u->pid.d[k]){
127       nearest(u->l,(k+1)%kd,x,h,mndist);
128       h[k] = abs(x.d[k]-u->pid.d[k]);
129       nearest(u->r,(k+1)%kd,x,h,mndist);
130     }else{
131       nearest(u->r,(k+1)%kd,x,h,mndist);
132       h[k] = abs(x.d[k]-u->pid.d[k]);
133       nearest(u->l,(k+1)%kd,x,h,mndist);
134     }
135     h[k]=old;
136   }
137   vector<point>in_range;
138   void range(node *u,int k,const point&mi,const point&ma){
139     if(!u)return;
140     bool is=1;
141     for(int i=0;i<kd;++i)
142       if(u->pid.d[i]<mi.d[i]||ma.d[i]<u->pid.d[i])
143         { is=0;break; }
144     if(is) in_range.push_back(u->pid);
145     if(mi.d[k]<=u->pid.d[k])range(u->l,(k+1)%kd,mi,ma);
146     if(ma.d[k]>=u->pid.d[k])range(u->r,(k+1)%kd,mi,ma);
147   }
148 public:
149   kd_tree(const T &INF,double a=0.75):
150   root(0),alpha(a),loga(log2(1.0/a)),INF(INF),maxn(1){}
151   ~kd_tree(){delete root;}
152   void clear(){delete root,root=0,maxn=1;}
153   void build(int n,const point *p){
154     delete root,A.resize(maxn=n);
155     for(int i=0;i<n;++i)A[i]=new node(p[i]);
156     root=build(0,0,n-1);
157   }
158   void insert(const point &x){
159     insert(root,0,x,__lg(size(root))/loga);
160     if(root->s>maxn)maxn=root->s;
161   }
162   bool erase(const point &p){
163     bool d=erase(root,0,p);
164     if(root&&root->s<alpha*maxn)rebuild();
165     return d;
166   }
167   void rebuild(){
168     if(root)rebuild(root,0);
169     maxn=root->s;
170   }
171   T nearest(const point &x,int k){
172     qM=k;
173     T mndist=INF,h[kd]={};
174     nearest(root,0,x,h,mndist);
175     mndist=pQ.top().first;
176     pQ = priority_queue<pair<T,point>>();
177     return mndist;//回傳離x第k近的點的距離
```

```
178    }
179    const vector<point> &range(const point&mi,const point&ma){
180      in_range.clear();
181      range(root,0,mi,ma);
182      return in_range;//回傳介於mi到ma之間的點vector
183    }
184    int size(){return root?root->s:0;}
185  };
```

2.2 FenwickTree

```
1  // 區間加值 BIT 只支援 1-based   O(Q*log(N)) 閉區間
2  class RangeUpdateBIT {
3    private:
4      ll d[maxn], dd[maxn];
5      ll sum(int i) {
6          ll s = 0, ss = 0;
7          int c = i + 1;
8          while (i > 0) s += d[i], ss += dd[i], i -= i & -i;
9          return c * s - ss;
10     }
11     void add(int i, ll v) {
12         int c = i;
13         while (i < maxn)
14             d[i] += v, dd[i] += c * v, i += i & -i;
15     }
16   public:
17     RangeUpdateBIT() {
18         memset(d, 0, sizeof(d));
19         memset(dd, 0, sizeof(dd));
20     }
21     ll sum(int l, int r) { return sum(r) - sum(l - 1); }
22     void add(int l, int r, ll v) {
23         add(l, v), add(r + 1, -v);
24     }
25  };
```

2.3 FenwickTree2D

```
1  /** 支援單點增值和區間查詢，O((A+Q)*log(A))，A
2   * 是矩陣面積。只能 用於 1-based **/
3  const int R = 256, C = 256;
4  class BIT2D {
5    private:
6      ll a[R + 1][C + 1];
7      ll sum(int x, int y) {
8          ll ret = 0;
9          for (int i = x; i; i -= (i & -i))
10             for (int j = y; j; j -= (j & -j))
11                 ret += a[i][j];
12         return ret;
13     }
14   public:
15     // 建立元素都是零的 R*C 大小的矩陣。
16     BIT2D() { memset(a, 0, sizeof(a)); }
17     // 單點增值，注意 1-based。
18     void add(int x, int y, ll v) {
19         for (int i = x; i <= R; i += (i & -i))
20             for (int j = y; j <= C; j += (j & -j))
```

```
21                 a[i][j] += v;
22     }
23     // 區間和，注意 1-based 。二維都是閉區間。
24     ll sum(int x0, int y0, int x1, int y1) {
25         return sum(x1, y1) - sum(x0 - 1, y1) -
26                sum(x1, y0 - 1) + sum(x0 - 1, y0 - 1);
27     }
28  };
```

2.4 HeavyLight

```
1  #include<vector>
2  #define MAXN 100005
3  int siz[MAXN],max_son[MAXN],pa[MAXN],dep[MAXN];
4  int link_top[MAXN],link[MAXN],cnt;
5  vector<int> G[MAXN];
6  void find_max_son(int u){
7    siz[u]=1;
8    max_son[u]=-1;
9    for(auto v:G[u]){
10     if(v==pa[u])continue;
11     pa[v]=u;
12     dep[v]=dep[u]+1;
13     find_max_son(v);
14     if(max_son[u]==-1||siz[v]>siz[max_son[u]])max_son[u]=v;
15     siz[u]+=siz[v];
16   }
17 }
18 void build_link(int u,int top){
19   link[u]=++cnt;
20   link_top[u]=top;
21   if(max_son[u]==-1)return;
22   build_link(max_son[u],top);
23   for(auto v:G[u]){
24     if(v==max_son[u]||v==pa[u])continue;
25     build_link(v,v);
26   }
27 }
28 int find_lca(int a,int b){
29   //求LCA，可以在過程中對區間進行處理
30   int ta=link_top[a],tb=link_top[b];
31   while(ta!=tb){
32     if(dep[ta]<dep[tb]){
33       swap(ta,tb);
34       swap(a,b);
35     }
36     //這裡可以對a所在的鏈做區間處理
37     //區間為 (link[ta],link[a])
38     ta=link_top[a=pa[ta]];
39   }
40   //最後a,b會在同一條鏈，若a!=b還要在進行一次區間處理
41   return dep[a]<dep[b]?a:b;
42 }
```

2.5 Link_Cut_Tree

```
1  struct splay_tree{
2    int ch[2],pa;//子節點跟父母
3    bool rev;//反轉的懶惰標記
```

```
4    splay_tree():pa(0),rev(0){ch[0]=ch[1]=0;}
5  };
6  vector<splay_tree> nd;
7  //有的時候用vector會TLE，要注意
8  //這邊以node[0]作為null節點
9  bool isroot(int x){//判斷是否為這棵splay tree的根
10   return nd[nd[x].pa].ch[0]!=x&&nd[nd[x].pa].ch[1]!=x;
11 }
12 void down(int x){//懶惰標記下推
13   if(nd[x].rev){
14     if(nd[x].ch[0])nd[nd[x].ch[0]].rev^=1;
15     if(nd[x].ch[1])nd[nd[x].ch[1]].rev^=1;
16     swap(nd[x].ch[0],nd[x].ch[1]);
17     nd[x].rev=0;
18   }
19 }
20 void push_down(int x){//所有祖先懶惰標記下推
21   if(!isroot(x))push_down(nd[x].pa);
22   down(x);
23 }
24 void up(int x){}//將子節點的資訊向上更新
25 void rotate(int x){//旋轉，會自行判斷轉的方向
26   int y=nd[x].pa,z=nd[y].pa,d=(nd[y].ch[1]==x);
27   nd[x].pa=z;
28   if(!isroot(y))nd[z].ch[nd[z].ch[1]==y]=x;
29   nd[y].ch[d]=nd[x].ch[d^1];
30   nd[nd[y].ch[d]].pa=y;
31   nd[y].pa=x,nd[x].ch[d^1]=y;
32   up(y),up(x);
33 }
34 void splay(int x){//將x伸展到splay tree的根
35   push_down(x);
36   while(!isroot(x)){
37     int y=nd[x].pa;
38     if(!isroot(y)){
39       int z=nd[y].pa;
40       if((nd[z].ch[0]==y)^(nd[y].ch[0]==x))rotate(y);
41       else rotate(x);
42     }
43     rotate(x);
44   }
45 }
46 int access(int x){
47   int last=0;
48   while(x){
49     splay(x);
50     nd[x].ch[1]=last;
51     up(x);
52     last=x;
53     x=nd[x].pa;
54   }
55   return last;//access後splay tree的根
56 }
57 void access(int x,bool is=0){//is=0就是一般的access
58   int last=0;
59   while(x){
60     splay(x);
61     if(is&&!nd[x].pa){
62       //printf("%d\n",max(nd[last].ma,nd[nd[x].ch[1]].ma));
63     }
64     nd[x].ch[1]=last;
65     up(x);
66     last=x;
67     x=nd[x].pa;
```

```
68      }
69  }
70  void query_edge(int u,int v){
71      access(u);
72      access(v,1);
73  }
74  void make_root(int x){
75      access(x),splay(x);
76      nd[x].rev^=1;
77  }
78  void make_root(int x){
79      nd[access(x)].rev^=1;
80      splay(x);
81  }
82  void cut(int x,int y){
83      make_root(x);
84      access(y);
85      splay(y);
86      nd[y].ch[0]=0;
87      nd[x].pa=0;
88  }
89  void cut_parents(int x){
90      access(x);
91      splay(x);
92      nd[nd[x].ch[0]].pa=0;
93      nd[x].ch[0]=0;
94  }
95  void link(int x,int y){
96      make_root(x);
97      nd[x].pa=y;
98  }
99  int find_root(int x){
100     x=access(x);
101     while(nd[x].ch[0])x=nd[x].ch[0];
102     splay(x);
103     return x;
104 }
105 int query(int u,int v){
106 //傳回uv路徑splay tree的根結點
107 //這種寫法無法求LCA
108     make_root(u);
109     return access(v);
110 }
111 int query_lca(int u,int v){
112 //假設求鏈上點權的總和，sum是子樹的權重和，data是節點的權重
113     access(u);
114     int lca=access(v);
115     splay(u);
116     if(u==lca){
117         //return nd[lca].data+nd[nd[lca].ch[1]].sum
118     }else{
119         //return nd[lca].data+nd[nd[lca].ch[1]].sum+nd[u].sum
120     }
121 }
122 struct EDGE{
123     int a,b,w;
124 }e[10005];
125 int n;
126 vector<pair<int,int>> G[10005];
127 //first表示子節點，second表示邊的編號
128 int pa[10005],edge_node[10005];
129 //pa是父母節點，暫存用的，edge_node是每個編被存在哪個點裡面的陣列
130 void bfs(int root){
```

```
131 //在建構的時候把每個點都設成一個splay tree
132     queue<int > q;
133     for(int i=1;i<=n;++i)pa[i]=0;
134     q.push(root);
135     while(q.size()){
136         int u=q.front();
137         q.pop();
138         for(auto P:G[u]){
139             int v=P.first;
140             if(v!=pa[u]){
141                 pa[v]=u;
142                 nd[v].pa=u;
143                 nd[v].data=e[P.second].w;
144                 edge_node[P.second]=v;
145                 up(v);
146                 q.push(v);
147             }
148         }
149     }
150 }
151 void change(int x,int b){
152     splay(x);
153     //nd[x].data=b;
154     up(x);
155 }
```

2.6 MaxSumSegmentTree

```
1  /** 計算最大區間連續和的線段樹，限定 1-based 。
2   * 複雜度 O(Q*log(N)) **/
3  #define ls i << 1
4  #define rs i << 1 | 1
5  class MaxSumSegmentTree {
6      private:
7      struct node {
8          ll lss, rss, ss, ans;
9          void set(ll v) { lss = rss = ss = ans = v; }
10     };
11     int n;
12     vector<node> a;  // 萬萬不可用普通陣列，要用 vector
13     vector<ll> z;
14     void pull(int i) {
15         a[i].ss = a[ls].ss + a[rs].ss;
16         a[i].lss = max(a[ls].lss, a[ls].ss + a[rs].lss);
17         a[i].rss = max(a[rs].rss, a[rs].ss + a[ls].rss);
18         a[i].ans = max(max(a[ls].ans, a[rs].ans),
19                        a[ls].rss + a[rs].lss);
20     }
21     void build(int i, int l, int r) {
22         if (l == r) return a[i].set(z[l]), void();
23         int m = (l + r) >> 1;
24         build(ls, l, m), build(rs, m + 1, r), pull(i);
25     }
26     void set(int i, int l, int r, int q, ll v) {
27         if (l == r) return a[i].set(v), void();
28         int m = (l + r) >> 1;
29         if (q <= m) set(ls, l, m, q, v);
30         else set(rs, m + 1, r, q, v);
31         pull(i);
32     }
33     node query(int i, int l, int r, int ql, int qr) {
34         if (ql <= l && r <= qr) return a[i];
35         int m = (l + r) >> 1;
36         if (qr <= m) return query(ls, l, m, ql, qr);
37         if (m < ql) return query(rs, m + 1, r, ql, qr);
38         node lo = query(ls, l, m, ql, qr),
39              ro = query(rs, m + 1, r, ql, qr), ans;
40         ans.ss = lo.ss + ro.ss;
41         ans.lss = max(lo.lss, lo.ss + ro.lss);
42         ans.rss = max(ro.rss, ro.ss + lo.rss);
43         ans.ans = max(max(lo.ans, ro.ans), lo.rss + ro.lss);
44         return ans;
45     }
46
47     public:
48     MaxSumSegmentTree(int n) : n(n) {
49         a.resize(n << 2), z.resize(n << 2);
50         build(1, 1, n);
51     }
52     // 單點設值。限定 1-based 。
53     inline void set(int i, ll v) { set(1, 1, n, i, v); }
54     // 問必區間 [l, r] 的最大子區間連續和。限定 1-based 。
55     inline ll query(int l, int r) {
56         return query(1, 1, n, l, r).ans;
57     }
58 };
```

2.7 PersistentSegmentTree

```
1  int a[maxn], b[maxn], root[maxn], cnt;
2  struct node {
3      int sum, L_son, R_son;
4  } tree[maxn << 5];
5  int create(int _sum, int _L_son, int _R_son) {
6      int idx = ++cnt;
7      tree[idx].sum = _sum, tree[idx].L_son = _L_son, tree[idx].R_son = _R_son;
8      return idx;
9  }
10 void Insert(int &root, int pre_rt, int pos, int L, int R) {
11     root = create(tree[pre_rt].sum+1, tree[pre_rt].L_son,
12         tree[pre_rt].R_son);
13     if(L==R) return;
14     int M = (L+R)>>1;
15     if(pos<=M) Insert(tree[root].L_son, tree[pre_rt].L_son,
16         pos, L, M);
17     else Insert(tree[root].R_son, tree[pre_rt].R_son, pos, M
18         +1, R);
19 }
20 int query(int L_id, int R_id, int L, int R, int K) {
21     if(L==R) return L;
22     int M = (L+R)>>1;
23     int s = tree[tree[R_id].L_son].sum - tree[tree[L_id].
24         L_son].sum;
25     if(K<=s) return query(tree[L_id].L_son, tree[R_id].L_son,
26         L, M, K);
27     return query(tree[L_id].R_son, tree[R_id].R_son, M+1, R,
28         K-s);
29 }
30 int main() {
31     int n,m; cin >> n >> m;
32     for(int i=1; i<=n; i++) {
33         cin >> a[i]; b[i] = a[i];
34     } sort(b+1,b+1+n); //離散化
```

```
29        int b_sz = unique(b+1, b+1+n) - (b+1);
30        cnt = root[0] = 0;
31        for(int i=1; i<=n; i++) {
32            int pos = lower_bound(b+1, b+1+b_sz, a[i]) - b;
33            Insert(root[i], root[i-1], pos, 1, b_sz);
34        }
35        while(m--) {
36            int l, r, k; cin >> l >> r >> k;
37            int pos = query(root[l-1],root[r],1,b_sz,k);
38            cout << b[pos] << endl;
39        } return 0;
40 }
```

2.8 RangeUpdateSegmentTree

```
1  //閉區間，1-based
2  #define ls i << 1
3  #define rs i << 1 | 1
4  const ll rr = 0x6891139;  // 亂數，若跟題目碰撞會吃 WA 或 RE
5  class RangeUpdateSegmentTree {
6    private:
7    struct node { //s : sum, x : max
8        int l, r; ll adt = 0, stt = rr, s = 0, x = 0;
9    };
10   vector<node> a; // 萬萬不可以用普通陣列，要用 vector
11   void push(int i) {
12       if (a[i].stt != rr) {
13           a[ls].stt = a[rs].stt = a[i].stt;
14           a[ls].adt = a[rs].adt = 0;
15           a[ls].x = a[rs].x = a[i].stt;
16           a[ls].s = (a[ls].r - a[ls].l + 1) * a[i].stt;
17           a[rs].s = (a[rs].r - a[rs].l + 1) * a[i].stt;
18           a[i].stt = rr;
19       }
20       if (a[i].adt) {
21           a[ls].adt += a[i].adt, a[rs].adt += a[i].adt;
22           a[ls].x += a[i].adt, a[rs].x += a[i].adt;
23           a[ls].s += a[i].adt * (a[ls].r - a[ls].l + 1);
24           a[rs].s += a[i].adt * (a[rs].r - a[rs].l + 1);
25           a[i].adt = 0;
26       }
27   }
28   void pull(int i) {
29       a[i].s = a[ls].s + a[rs].s;
30       a[i].x = max(a[ls].x, a[rs].x);
31   }
32   void build(int l, int r, int i) {
33       a[i].l = l, a[i].r = r;
34       if (l == r) return;
35       int mid = (l + r) >> 1;
36       build(l, mid, ls), build(mid + 1, r, rs);
37   }
38   public:
39   RangeUpdateSegmentTree(int n) : a(n << 2) {
40       build(1, n, 1);
41   }
42   void set(int l, int r, ll val, int i = 1) {
43       if (a[i].l >= l && a[i].r <= r) {
44           a[i].s = val * (a[i].r - a[i].l + 1);
45           a[i].x = a[i].stt = val;
46           a[i].adt = 0;
47           return;
```

```
48       }
49       push(i);
50       int mid = (a[i].l + a[i].r) >> 1;
51       if (l <= mid) set(l, r, val, ls);
52       if (r > mid) set(l, r, val, rs);
53       pull(i);
54   }
55   void add(int l, int r, ll val, int i = 1) {
56       if (a[i].l >= l && a[i].r <= r) {
57           a[i].s += val * (a[i].r - a[i].l + 1);
58           a[i].x += val;
59           a[i].adt += val;
60           return;
61       }
62       push(i);
63       int mid = (a[i].l + a[i].r) >> 1;
64       if (l <= mid) add(l, r, val, ls);
65       if (r > mid) add(l, r, val, rs);
66       pull(i);
67   }
68   ll maxx(int l, int r, int i = 1) {
69       if (l <= a[i].l && a[i].r <= r) return a[i].x;
70       push(i);
71       ll ret = -9e18;
72       int mid = (a[i].l + a[i].r) >> 1;
73       if (l <= mid) ret = max(ret, maxx(l, r, ls));
74       if (r > mid) ret = max(ret, maxx(l, r, rs));
75       pull(i);
76       return ret;
77   }
78   ll sum(int l, int r, int i = 1) {
79       if (l <= a[i].l && a[i].r <= r) return a[i].s;
80       push(i);
81       ll ret = 0;
82       int mid = (a[i].l + a[i].r) >> 1;
83       if (l <= mid) ret += sum(l, r, ls);
84       if (r > mid) ret += sum(l, r, rs);
85       pull(i);
86       return ret;
87   }
88 };
```

2.9 SparseTable

```
1  #define flg(a) floor(log2(a))
2  struct SparseTable {
3      vector<vector<ll>> a;
4      SparseTable(vector<ll>& data) {
5          int n = data.size();
6          a.assign(flg(n) + 1, vector<ll>(n));
7          a[0] = data;
8          for (int i = 1; (1 << i) <= n; i++)
9              for (int j = 0, k = n - (1 << i); j <= k; j++)
10                 a[i][j] = max(a[i - 1][j],
11                               a[i - 1][j + (1 << (i - 1))]);
12     }
13     ll maxx(int l, int r) { // [l, r], 0/1-based
14         int k = flg(r - l + 1);
15         return max(a[k][l], a[k][r - (1 << k) + 1]);
16     }
17 };
```

2.10 Treap

```
1  // 區間加值、反轉、rotate、刪除、插入元素、求區間
2  // srand(time(0))
3  class Treap {
4    private:
5    struct Node {
6        int pri = rand(), size = 1;
7        ll val, mn, inc = 0; bool rev = 0;
8        Node *lc = 0, *rc = 0;
9        Node(ll v) { val = mn = v; }
10   };
11   Node* root = 0;
12   void rev(Node* t) {
13       if (!t) return;
14       swap(t->lc, t->rc), t->rev ^= 1;
15   }
16   void update(Node* t, ll v) {
17       if (!t) return;
18       t->val += v, t->inc += v, t->mn += v;
19   }
20   void push(Node* t) {
21       if (t->rev) rev(t->lc), rev(t->rc), t->rev = 0;
22       update(t->lc, t->inc), update(t->rc, t->inc);
23       t->inc = 0;
24   }
25   void pull(Node* t) {
26       t->size = 1 + size(t->lc) + size(t->rc);
27       t->mn = t->val;
28       if (t->lc) t->mn = min(t->mn, t->lc->mn);
29       if (t->rc) t->mn = min(t->mn, t->rc->mn);
30   }
31   void discard(Node* t) { // 看要不要釋放記憶體
32       if (!t) return;
33       discard(t->lc), discard(t->rc);
34       delete t;
35   }
36   void split(Node* t, Node*& a, Node*& b, int k) {
37       if (!t) return a = b = 0, void();
38       push(t);
39       if (size(t->lc) < k) {
40           a = t;
41           split(t->rc, a->rc, b, k - size(t->lc) - 1);
42           pull(a);
43       } else {
44           b = t;
45           split(t->lc, a, b->lc, k);
46           pull(b);
47       }
48   }
49   Node* merge(Node* a, Node* b) {
50       if (!a || !b) return a ? a : b;
51       if (a->pri > b->pri) {
52           push(a);
53           a->rc = merge(a->rc, b);
54           pull(a);
55           return a;
56       } else {
57           push(b);
58           b->lc = merge(a, b->lc);
59           pull(b);
60           return b;
61       }
62   }
63   inline int size(Node* t) { return t ? t->size : 0; }
```

```
 64    public:
 65     int size() { return size(root); }
 66     void add(int l, int r, ll val) {
 67         Node *a, *b, *c, *d;
 68         split(root, a, b, r);
 69         split(a, c, d, l - 1);
 70         update(d, val);
 71         root = merge(merge(c, d), b);
 72     }
 73     // 反轉區間 [l, r]
 74     void reverse(int l, int r) {
 75         Node *a, *b, *c, *d;
 76         split(root, a, b, r);
 77         split(a, c, d, l - 1);
 78         swap(d->lc, d->rc);
 79         d->rev ^= 1;
 80         root = merge(merge(c, d), b);
 81     }
 82     // 區間 [l, r] 向右 rotate k 次，k < 0 表向左 rotate
 83     void rotate(int l, int r, int k) {
 84         int len = r - l + 1;
 85         Node *a, *b, *c, *d, *e, *f;
 86         split(root, a, b, r);
 87         split(a, c, d, l - 1);
 88         k = (k + len) % len;
 89         split(d, e, f, len - k);
 90         root = merge(merge(c, merge(f, e)), b);
 91     }
 92     // 插入一個元素 val 使其 index = i <= size
 93     void insert(int i, ll val) {
 94         if (i == size() + 1) {
 95             push_back(val); return;
 96         }
 97         assert(i <= size());
 98         Node *a, *b;
 99         split(root, a, b, i - 1);
100         root = merge(merge(a, new Node(val)), b);
101     }
102     void push_back(ll val) {
103         root = merge(root, new Node(val));
104     }
105     void remove(int l, int r) {
106         int len = r - l + 1;
107         Node *a, *b, *c, *d;
108         split(root, a, b, l - 1);
109         split(b, c, d, len);
110         discard(c); // 看你要不要釋放記憶體
111         root = merge(a, d);
112     }
113     ll minn(int l, int r) {
114         Node *a, *b, *c, *d;
115         split(root, a, b, r);
116         split(a, c, d, l - 1);
117         int ans = d->mn;
118         root = merge(merge(c, d), b);
119         return ans;
120     }
121 };
```

3 Flow_Matching

3.1 Dinic

```
 1 class Dinic {
 2   private:
 3     struct edge { int d, r; ll c; };
 4     vector<vector<edge>> adj; vector<int> lv, ve; int n;
 5     bool mklv(int s, int d) {
 6         lv.assign(n, -1); lv[s] = 0; queue<int> q({s});
 7         while (!q.empty()) {
 8             int v = q.front(); q.pop();
 9             for (auto& e : adj[v]) {
10                 if (e.c == 0 || lv[e.d] != -1) continue;
11                 lv[e.d] = lv[v] + 1, q.push(e.d);
12             }
13         }
14         return lv[d] > 0;
15     }
16     ll aug(int v, ll f, int d) {
17         if (v == d) return f;
18         for (; ve[v] < adj[v].size(); ve[v]++) {
19             auto& e = adj[v][ve[v]];
20             if (lv[e.d] != lv[v] + 1 || !e.c) continue;
21             ll sent = aug(e.d, min(f, e.c), d);
22             if (sent > 0) {
23                 e.c -= sent, adj[e.d][e.r].c += sent;
24                 return sent;
25             }
26         }
27         return 0;
28     }
29   public:
30     // 建空圖。n 為節點數量 (含 source 和 sink)。
31     Dinic(int n) : n(n + 1) { clear(); }
32     void clear() { adj.assign(n, {}); }
33     void add_edge(int src, int dst, ll cap) {
34         edge ss{dst, (int)adj[dst].size(), cap};
35         edge dd{src, (int)adj[src].size(), 0};
36         adj[src].push_back(ss), adj[dst].push_back(dd);
37     }
38     ll max_flow(int s, int d) {
39         ll ret = 0;
40         while (mklv(s, d)) {
41             ve.assign(n, 0);
42             while (ll f = aug(s, inf, d)) ret += f;
43         }
44         return ret;
45     }
46 };
```

3.2 Ford_Fulkerson

```
 1 const int maxn = 1e5 + 10, INF = 1e9;
 2 const long long INF64 = 1e18;
 3 struct edge{ int to, cap, rev; };
 4 vector<edge> G[maxn];
 5 int n, m, s, t, a, b, c;
 6 bool vis[maxn];
 7 int dfs(int v, int t, int f) {
```

```
 8     cout << v << ' ' << t << ' ' << f << '\n';
 9     if (v == t) return f;
10     vis[v] = true;
11     for (edge &e: G[v]) {
12         if (!vis[e.to] && e.cap > 0) {
13             int d = dfs(e.to, t, min(f, e.cap));
14             if (d > 0) {
15                 e.cap -= d, G[e.to][e.rev].cap += d;
16                 return d;
17             }
18         }
19     }
20     return 0;
21 }
22 int ford_fulkerson(int s, int t) {
23     int flow = 0, f;
24     for (int i = 0; i < n; i++) {
25         cout << i << " : ";
26         for (edge e: G[i])
27             cout << '(' << e.to << ',' << e.cap << ')' << ' ';
28         cout << '\n';
29     }
30     do {
31         memset(vis, false, sizeof(vis));
32         f = dfs(s, t, INF);
33         for (int i = 0; i < n; i++) {
34             cout << i << " : ";
35             for (edge e: G[i])
36                 cout << '(' << e.to << ',' << e.cap << ')' << ' ';
37             cout << '\n';
38         }
39         cout << f << '\n';
40         flow += f;
41     } while (f > 0);
42     return flow;
43 }
44 void init(int n) {
45     for (int i = 0; i < n; i++) G[i].clear();
46 }
47 int main() {
48     cin >> n >> m >> s >> t;
49     init(n);
50     while (m--) {
51         cin >> a >> b >> c;
52         G[a].push_back((edge){b, c, (int)G[b].size()});
53         G[b].push_back((edge){a, 0, (int)G[a].size() - 1});
54     }
55     cout << ford_fulkerson(s, t) << '\n';
56     return 0;
57 }
```

3.3 Hopcroft_Karp

```
 1 // 匈牙利算法的優化，二分圖最大匹配 O(E√V)
 2 int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
 3 vector<int> edge[maxn]; // for Left
 4 bool dfs(int u) {
 5     vis[u] = true;
 6     for (vector<int>::iterator it = edge[u].begin();
 7          it != edge[u].end(); ++it) {
 8         int v = pr2[*it];
```

```
 9        if (v == -1 ||
10            (!vis[v] && level[u] < level[v] && dfs(v))) {
11            pr[u] = *it, pr2[*it] = u;
12            return true;
13        }
14    } return false;
15 }
16 int hopcroftKarp() {
17    memset(pr, -1, sizeof(pr));
18    memset(pr2, -1, sizeof(pr2));
19    for (int match = 0;;) {
20        queue<int> Q;
21        for (int i = 1; i <= n; ++i) {
22            if (pr[i] == -1) level[i] = 0, Q.push(i);
23            else level[i] = -1;
24        }
25        while (!Q.empty()) {
26            int u = Q.front(); Q.pop();
27            for (vector<int>::iterator it = edge[u].begin();
28                 it != edge[u].end(); ++it) {
29                int v = pr2[*it];
30                if (v != -1 && level[v] < 0)
31                    level[v] = level[u] + 1, Q.push(v);
32            }
33        }
34        for (int i = 1; i <= n; ++i) vis[i] = false;
35        int d = 0;
36        for (int i = 1; i <= n; ++i)
37            if (pr[i] == -1 && dfs(i)) ++d;
38        if (d == 0) return match;
39        match += d;
40    }
41 }
```

3.4   Hungarian

```
 1 // Time: O(VE)
 2 const int INF = 2e9;
 3 const int N = ? ;        // 男女總人數；女 id: 0 ~ p，男 id: p
       +1 ~ N-1
 4 int vis[N], rnd, m[N];   // 跑完匈牙利後配對結果儲存於此， -1
       表示人醜
 5 vector<int> g[N];        // 關係表
 6 int dfs(int s) {
 7    for (int x : g[s]) {
 8        if (vis[x]) continue;
 9        vis[x] = 1;
10        if (m[x] == -1 || dfs(m[x])) {
11            m[x] = s, m[s] = x;
12            return 1;
13        }
14    } return 0;
15 }
16 int hungarian(int p) {  // p : 女性人數
17    memset(m, -1, sizeof(m));
18    int c = 0;
19    for (int i = 0; i < p; i++) {
20        if (m[i] == -1) {
21            memset(vis, 0, sizeof(vis));
22            c += dfs(i);
23        }
24    } return c; // 成功結婚對數
```

3.5   KM

```
 1 /* 時間複雜度 O(N^3)
 2 求完美匹配中的最大權匹配
 3 如果不存在完美匹配，求最大匹配
 4 如果存在數個最大匹配，求數個最大匹配當中最大權匹配 */
 5 const ll INF = 5e18;
 6 const int N = ?;   // maxn
 7 int n;             // count of vertex (one side)
 8 ll g[N][N];        // weights
 9 class KM {
10    private:
11    ll lx[N], ly[N], s[N];
12    int px[N], py[N], m[N], p[N];
13    void adj(int y) {  // 把增廣路上所有邊反轉
14        m[y] = py[y];
15        if (px[m[y]] != -2)
16            adj(px[m[y]]);
17    }
18    bool dfs(int x) {  // DFS找增廣路
19        for (int y = 0; y < n; ++y) {
20            if (py[y] != -1) continue;
21            ll t = lx[x] + ly[y] - g[x][y];
22            if (t == 0) {
23                py[y] = x;
24                if (m[y] == -1) {
25                    adj(y);
26                    return 1;
27                }
28                if (px[m[y]] != -1) continue;
29                px[m[y]] = y;
30                if (dfs(m[y])) return 1;
31            } else if (s[y] > t) {
32                s[y] = t, p[y] = x;
33            }
34        }
35        return 0;
36    }
37
38    public:
39    ll max_weight() {
40        memset(ly, 0, sizeof(ly));
41        memset(m, -1, sizeof(m));
42        for (int x = 0; x < n; ++x) {
43            lx[x] = -INF;
44            for (int y = 0; y < n; ++y)
45                lx[x] = max(lx[x], g[x][y]);
46        }
47        for (int x = 0; x < n; ++x) {
48            for (int y = 0; y < n; ++y) s[y] = INF;
49            memset(px, -1, sizeof(px));
50            memset(py, -1, sizeof(py));
51            px[x] = -2;
52            if (dfs(x)) continue;
53            bool flag = 1;
54            while (flag) {
55                ll cut = INF;
56                for (int y = 0; y < n; ++y)
57                    if (py[y] == -1 && cut > s[y]) cut = s[y
                        ];
```

```
58                    for (int j = 0; j < n; ++j) {
59                        if (px[j] != -1) lx[j] -= cut;
60                        if (py[j] != -1) ly[j] += cut;
61                        else s[j] -= cut;
62                    }
63                    for (int y = 0; y < n; ++y) {
64                        if (py[y] == -1 && s[y] == 0) {
65                            py[y] = p[y];
66                            if (m[y] == -1) {
67                                adj(y);
68                                flag = 0;
69                                break;
70                            }
71                            px[m[y]] = y;
72                            if (dfs(m[y])) {
73                                flag = 0;
74                                break;
75                            }
76                        }
77                    }
78                }
79            }
80            ll ans = 0;
81            for (int y = 0; y < n; ++y)
82                if (g[m[y]][y] != -INF) ans += g[m[y]][y];
83            return ans;
84        }
85 };
```

3.6   Min_Cost_Max_Flow

```
 1 class MCMF { // 0/1-based
 2    private:
 3    struct edge { int to, r; ll rest, c; };
 4    int n; ll f = 0, c = 0;
 5    vector<vector<edge>> g;
 6    vector<int> pre, prel;
 7    bool run(int s, int t) {
 8        vector<ll> dis(n, inf); vector<bool> vis(n);
 9        dis[s] = 0; queue<int> q; q.push(s);
10        while (q.size()) {
11            int u = q.front(); q.pop(); vis[u] = 0;
12            for (int i = 0; i < g[u].size(); i++) {
13                int v = g[u][i].to; ll w = g[u][i].c;
14                if (g[u][i].rest <= 0 ||
15                    dis[v] <= dis[u] + w) continue;
16                pre[v] = u, prel[v] = i;
17                dis[v] = dis[u] + w;
18                if (!vis[v]) vis[v] = 1, q.push(v);
19            }
20        }
21        if (dis[t] == inf) return 0;
22        ll tf = inf;
23        for (int v = t, u, l; v != s; v = u) {
24            u = pre[v], l = prel[v];
25            tf = min(tf, g[u][l].rest);
26        }
27        for (int v = t, u, l; v != s; v = u) {
28            u = pre[v], l = prel[v], g[u][l].rest -= tf;
29            g[v][g[u][l].r].rest += tf;
30        }
31        c += tf * dis[t], f += tf;
32        return 1;
```

```
33        }
34    public:
35      MCMF(int n) // 建空圖，n 節點數 (含 src 和 sink)
36        : n(n + 1), g(n + 1), pre(n + 1), prel(n + 1) {}
37      // 加有向邊 u->v ，cap 容量 cost 成本
38      void add_edge(int u, int v, ll cap, ll cost) {
39          g[u].push_back({v, (int)g[v].size(), cap, cost});
40          g[v].push_back({u, (int)g[u].size() - 1, 0, -cost});
41      }
42      pair<ll, ll> query(int src, int sink) {
43          while (run(src, sink));
44          return {f, c};   //{min cost, max flow}
45      }
46  };
```

3.7 SW_MinCut

```
1   // all pair min cut, global min cut
2   struct SW { // O(V^3)
3       static const int MXN = 514;
4       int n, vst[MXN], del[MXN];
5       int edge[MXN][MXN], wei[MXN];
6       void init(int _n){
7           n = _n; FZ(edge); FZ(del);
8       }
9       void addEdge(int u, int v, int w) {
10          edge[u][v] += w; edge[v][u] += w;
11      }
12      void search(int &s, int &t) {
13          FZ(vst); FZ(wei);
14          s = t = -1;
15          while (true){
16              int mx=-1, cur=0;
17              for (int i=0; i<n; i++)
18                  if (!del[i] && !vst[i] && mx<wei[i])
19                      cur = i, mx = wei[i];
20              if (mx == -1) break;
21              vst[cur] = 1;
22              s = t; t = cur;
23              for (int i=0; i<n; i++)
24                  if (!vst[i] && !del[i]) wei[i] += edge[cur][i
                        ];
25          }
26      }
27      int solve() {
28          int res = 2147483647;
29          for (int i=0, x, y; i<n-1; i++) {
30              search(x,y);
31              res = min(res,wei[y]);
32              del[y] = 1;
33              for (int j=0; j<n; j++)
34                  edge[x][j] = (edge[j][x] += edge[y][j]);
35          }
36          return res;
37      }
38  } graph;
```

4 Geometry

4.1 ClosestPair

```
1   typedef pair<ll, ll> pii;
2   #define x first
3   #define y second
4   ll dd(const pii& a, const pii& b) {
5       ll dx = a.x - b.x, dy = a.y - b.y;
6       return dx * dx + dy * dy;
7   }
8   const ll inf = 1e18;
9   ll dac(vector<pii>& p, int l, int r) {
10      if (l >= r) return inf;
11      int m = (l + r) / 2;
12      ll d = min(dac(p, l, m), dac(p, m + 1, r));
13      vector<pii> t;
14      for (int i = m; i >= l && p[m].x - p[i].x < d; i--)
15          t.push_back(p[i]);
16      for (int i = m + 1; i <= r && p[i].x - p[m].x < d; i++)
17          t.push_back(p[i]);
18      sort(t.begin(), t.end(),
19          [](pii& a, pii& b) { return a.y < b.y; });
20      int n = t.size();
21      for (int i = 0; i < n - 1; i++)
22          for (int j = 1; j < 4 && i + j < n; j++)
23              // 這裡可以知道是哪兩點是最小點對
24              d = min(d, dd(t[i], t[i + j]));
25      return d;
26  }
27  // 給一堆點，求最近點對的距離「的平方」。
28  ll closest_pair(vector<pii>& pp) {
29      sort(pp.begin(), pp.end());
30      return dac(pp, 0, pp.size() - 1);
31  }
```

4.2 Geometry

```
1   //Copy from Jinkela
2   const double PI=atan2(0.0,-1.0);
3   template<typename T>
4   struct point{
5       T x,y;
6       point(){}
7       point(const T&x,const T&y):x(x),y(y){}
8       point operator+(const point &b)const{
9           return point(x+b.x,y+b.y); }
10      point operator-(const point &b)const{
11          return point(x-b.x,y-b.y); }
12      point operator*(const T &b)const{
13          return point(x*b,y*b); }
14      point operator/(const T &b)const{
15          return point(x/b,y/b); }
16      bool operator==(const point &b)const{
17          return x==b.x&&y==b.y; }
18      T dot(const point &b)const{
19          return x*b.x+y*b.y; }
20      T cross(const point &b)const{
21          return x*b.y-y*b.x; }
22      point normal()const{//求法向量
```

```
23          return point(-y,x); }
24      T abs2()const{//向量長度的平方
25          return dot(*this); }
26      T rad(const point &b)const{//兩向量的弧度
27  return fabs(atan2(fabs(cross(b)),dot(b))); }
28      T getA()const{//對 x 軸的弧度
29          T A=atan2(y,x);//超過 180 度會變負的
30          if(A<=-PI/2)A+=PI*2;
31          return A;
32      }
33  };
34  template<typename T>
35  struct line{
36      line(){}
37      point<T> p1,p2;
38      T a,b,c;//ax+by+c=0
39      line(const point<T>&x,const point<T>&y):p1(x),p2(y){}
40      void pton(){//轉成一般式
41          a=p1.y-p2.y;
42          b=p2.x-p1.x;
43          c=-a*p1.x-b*p1.y;
44      }
45      T ori(const point<T> &p)const{//點和有向直線的關係，>0左
                邊、 =0 在線上 <0 右邊
46          return (p2-p1).cross(p-p1);
47      }
48      T btw(const point<T> &p)const{//點投影落在線段上 <=0
49          return (p1-p).dot(p2-p);
50      }
51      bool point_on_segment(const point<T>&p)const{//點是否在線段
                上
52          return ori(p)==0&&btw(p)<=0;
53      }
54      T dis2(const point<T> &p,bool is_segment=0)const{//點跟直線
                /線段的距離平方
55          point<T> v=p2-p1,v1=p-p1;
56          if(is_segment){
57              point<T> v2=p-p2;
58              if(v.dot(v1)<=0)return v1.abs2();
59              if(v.dot(v2)>=0)return v2.abs2();
60          }
61          T tmp=v.cross(v1);
62          return tmp*tmp/v.abs2();
63      }
64      T seg_dis2(const line<T> &l)const{//兩線段距離平方
65          return min({dis2(l.p1,1),dis2(l.p2,1),l.dis2(p1,1),l.dis2
                (p2,1)});
66      }
67      point<T> projection(const point<T> &p)const{//點對直線的投
                影
68          point<T> n=(p2-p1).normal();
69          return p-n*(p-p1).dot(n)/n.abs2();
70      }
71      point<T> mirror(const point<T> &p)const{
72          //點對直線的鏡射，要先呼叫 pton 轉成一般式
73          point<T> R;
74          T d=a*a+b*b;
75          R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
76          R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
77          return R;
78      }
79      bool equal(const line &l)const{//直線相等
80          return ori(l.p1)==0&&ori(l.p2)==0;
```

```cpp
 81    }
 82    bool parallel(const line &l)const{
 83      return (p1-p2).cross(l.p1-l.p2)==0;
 84    }
 85    bool cross_seg(const line &l)const{
 86      return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;
         //直線是否交線段
 87    }
 88    int line_intersect(const line &l)const{//直線相交情況，-1無
         限多點、1交於一點、0不相交
 89      return parallel(l)?(ori(l.p1)==0?-1:0):1;
 90    }
 91    int seg_intersect(const line &l)const{
 92      T c1=ori(l.p1), c2=ori(l.p2);
 93      T c3=l.ori(p1), c4=l.ori(p2);
 94      if(c1==0&&c2==0){//共線
 95        bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
 96        T a3=l.btw(p1),a4=l.btw(p2);
 97        if(b1&&b2&&a3==0&&a4>=0) return 2;
 98        if(b1&&b2&&a3>=0&&a4==0) return 3;
 99        if(b1&&b2&&a3>=0&&a4>=0) return 0;
100        return -1;//無限交點
101      }else if(c1*c2<=0&&c3*c4<=0)return 1;
102      return 0;//不相交
103    }
104    point<T> line_intersection(const line &l)const{/*直線交點*/
105      point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
106      //if(a.cross(b)==0)return INF;
107      return p1+a*(s.cross(b)/a.cross(b));
108    }
109    point<T> seg_intersection(const line &l)const{//線段交點
110      int res=seg_intersect(l);
111      if(res<=0) assert(0);
112      if(res==2) return p1;
113      if(res==3) return p2;
114      return line_intersection(l);
115    }
116  };
117  template<typename T>
118  struct polygon{
119    polygon(){}
120    vector<point<T> > p;//逆時針順序
121    T area()const{//面積
122      T ans=0;
123      for(int i=p.size()-1,j=0;j<(int)p.size();i=j++)
124        ans+=p[i].cross(p[j]);
125      return ans/2;
126    }
127    point<T> center_of_mass()const{//重心
128      T cx=0,cy=0,w=0;
129      for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
130        T a=p[i].cross(p[j]);
131        cx+=(p[i].x+p[j].x)*a;
132        cy+=(p[i].y+p[j].y)*a;
133        w+=a;
134      }
135      return point<T>(cx/3/w,cy/3/w);
136    }
137    char ahas(const point<T>& t)const{//點是否在簡單多邊形內，
         是的話回傳1、在邊上回傳-1、否則回傳0
138      bool c=0;
139      for(int i=0,j=p.size()-1;i<p.size();j=i++)
140        if(line<T>(p[i],p[j]).point_on_segment(t))return -1;
141        else if((p[i].y>t.y)!=(p[j].y>t.y)&&
142          t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x
             )
143          c=!c;
144      return c;
145    }
146    char point_in_convex(const point<T>&x)const{
147      int l=1,r=(int)p.size()-2;
148      while(l<=r){//點是否在凸多邊形內，是的話回傳1、在邊上回傳
             -1、否則回傳0
149        int mid=(l+r)/2;
150        T a1=(p[mid]-p[0]).cross(x-p[0]);
151        T a2=(p[mid+1]-p[0]).cross(x-p[0]);
152        if(a1>=0&&a2<=0){
153          T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
154          return res>0?1:(res>=0?-1:0);
155        }else if(a1<0)r=mid-1;
156        else l=mid+1;
157      }
158      return 0;
159    }
160    vector<T> getA()const{//凸包邊對x軸的夾角
161      vector<T>res;//一定是遞增的
162      for(size_t i=0;i<p.size();++i)
163        res.push_back((p[(i+1)%p.size()]-p[i]).getA());
164      return res;
165    }
166    bool line_intersect(const vector<T>&A,const line<T> &l)
         const{//O(logN)
167      int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-
           A.begin();
168      int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-
           A.begin();
169      return l.cross_seg(line<T>(p[f1],p[f2]));
170    }
171    polygon cut(const line<T> &l)const{//凸包對直線切割，得到直
         線l左側的凸包
172      polygon ans;
173      for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
174        if(l.ori(p[i])>=0){
175          ans.p.push_back(p[i]);
176          if(l.ori(p[j])<0)
177            ans.p.push_back(l.line_intersection(line<T>(p[i],p[
               j])));
178        }else if(l.ori(p[j])>0)
179          ans.p.push_back(l.line_intersection(line<T>(p[i],p[j
             ])));
180      }
181      return ans;
182    }
183    static bool graham_cmp(const point<T>& a,const point<T>& b)
         {//凸包排序函數
184      return (a.x<b.x)||(a.x==b.x&&a.y<b.y);
185    }
186    void graham(vector<point<T> > &s){//凸包
187      sort(s.begin(),s.end(),graham_cmp);
188      p.resize(s.size()+1);
189      int m=0;
190      for(size_t i=0;i<s.size();++i){
191        while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
192        p[m++]=s[i];
193      }
194      for(int i=s.size()-2,t=m+1;i>=0;--i){
195        while(m>=t&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
196        p[m++]=s[i];
197      }
198      if(s.size()>1)--m;
199      p.resize(m);
200    }
201    T diam(){//直徑
202      int n=p.size(),t=1;
203      T ans=0;p.push_back(p[0]);
204      for(int i=0;i<n;i++){
205        point<T> now=p[i+1]-p[i];
206        while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t
             +1)%n;
207        ans=max(ans,(p[i]-p[t]).abs2());
208      }
209      return p.pop_back(),ans;
210    }
211    T min_cover_rectangle(){//最小覆蓋矩形
212      int n=p.size(),t=1,r=1,l;
213      if(n<3)return 0;//也可以做最小周長矩形
214      T ans=1e99;p.push_back(p[0]);
215      for(int i=0;i<n;i++){
216        point<T> now=p[i+1]-p[i];
217        while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=(t
             +1)%n;
218        while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n
             ;
219        if(!i)l=r;
220        while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%
             n;
221        T d=now.abs2();
222        T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(
             p[l]-p[i]))/d;
223        ans=min(ans,tmp);
224      }
225      return p.pop_back(),ans;
226    }
227    T dis2(polygon &pl){//凸包最近距離平方
228      vector<point<T> > &P=p,&Q=pl.p;
229      int n=P.size(),m=Q.size(),l=0,r=0;
230    for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
231    for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
232      P.push_back(P[0]),Q.push_back(Q[0]);
233      T ans=1e99;
234      for(int i=0;i<n;++i){
235        while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
236        ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],
             Q[r+1])));
237        l=(l+1)%n;
238      }
239      return P.pop_back(),Q.pop_back(),ans;
240    }
241    static char sign(const point<T>&t){
242      return (t.y==0?t.x:t.y)<0;
243    }
244    static bool angle_cmp(const line<T>& A,const line<T>& B){
245      point<T> a=A.p2-A.p1,b=B.p2-B.p1;
246      return sign(a)<sign(b)||(sign(a)==sign(b)&&a.cross(b)>0);
247    }
248    int halfplane_intersection(vector<line<T> > &s){//半平面交
249      sort(s.begin(),s.end(),angle_cmp);//線段左側為該線段半平
           面
250      int L,R,n=s.size();
251      vector<point<T> > px(n);
252      vector<line<T> > q(n);
253      q[L=R=0]=s[0];
254      for(int i=1;i<n;++i){
```

```cpp
255      while(L<R&&s[i].ori(px[R-1])<=0)--R;
256      while(L<R&&s[i].ori(px[L])<=0)++L;
257      q[++R]=s[i];
258      if(q[R].parallel(q[R-1])){
259        --R;
260        if(q[R].ori(s[i].p1)>0)q[R]=s[i];
261      }
262      if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
263    }
264    while(L<R&&q[L].ori(px[R-1])<=0)--R;
265    p.clear();
266    if(R-L<=1)return 0;
267    px[R]=q[R].line_intersection(q[L]);
268    for(int i=L;i<=R;++i)p.push_back(px[i]);
269    return R-L+1;
270  }
271 };
272 template<typename T>
273 struct triangle{
274  point<T> a,b,c;
275  triangle(){}
276  triangle(const point<T> &a,const point<T> &b,const point<T>
          &c):a(a),b(b),c(c){}
277  T area()const{
278    T t=(b-a).cross(c-a)/2;
279    return t>0?t:-t;
280  }
281  point<T> barycenter()const{//重心
282    return (a+b+c)/3;
283  }
284  point<T> circumcenter()const{//外心
285    static line<T> u,v;
286    u.p1=(a+b)/2;
287    u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
288    v.p1=(a+c)/2;
289    v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
290    return u.line_intersection(v);
291  }
292  point<T> incenter()const{//內心
293    T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).
          abs2());
294    return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B
          +C);
295  }
296  point<T> perpencenter()const{//垂心
297    return barycenter()*3-circumcenter()*2;
298  }
299 };
300 template<typename T>
301 struct point3D{
302  T x,y,z;
303  point3D(){}
304  point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
305  point3D operator+(const point3D &b)const{
306    return point3D(x+b.x,y+b.y,z+b.z);}
307  point3D operator-(const point3D &b)const{
308    return point3D(x-b.x,y-b.y,z-b.z);}
309  point3D operator*(const T &b)const{
310    return point3D(x*b,y*b,z*b);}
311  point3D operator/(const T &b)const{
312    return point3D(x/b,y/b,z/b);}
313  bool operator==(const point3D &b)const{
314    return x==b.x&&y==b.y&&z==b.z;}
315  T dot(const point3D &b)const{
316    return x*b.x+y*b.y+z*b.z;}
317  point3D cross(const point3D &b)const{
318    return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
319  T abs2()const{//向量長度的平方
320    return dot(*this);}
321  T area2(const point3D &b)const{//和b、原點圍成面積的平方
322    return cross(b).abs2()/4;}
323 };
324 template<typename T>
325 struct line3D{
326  point3D<T> p1,p2;
327  line3D(){}
328  line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2
          (p2){}
329  T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直
          線/線段的距離平方
330    point3D<T> v=p2-p1,v1=p-p1;
331    if(is_segment){
332      point3D<T> v2=p-p2;
333      if(v.dot(v1)<=0)return v1.abs2();
334      if(v.dot(v2)>=0)return v2.abs2();
335    }
336    point3D<T> tmp=v.cross(v1);
337    return tmp.abs2()/v.abs2();
338  }
339  pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> &
          l)const{
340    point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
341    point3D<T> N=v1.cross(v2),ab(p1-l.p1);
342    //if(N.abs2()==0)return NULL;平行或重合
343    T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//最近點對距離
344    point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1
          ;
345    T t1=(G.cross(d2)).dot(D)/D.abs2();
346    T t2=(G.cross(d1)).dot(D)/D.abs2();
347    return make_pair(p1+d1*t1,l.p1+d2*t2);
348  }
349  bool same_side(const point3D<T> &a,const point3D<T> &b)
          const{
350    return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
351  }
352 };
353 template<typename T>
354 struct plane{
355  point3D<T> p0,n;//平面上的點和法向量
356  plane(){}
357  plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n)
          {}
358  T dis2(const point3D<T> &p)const{//點到平面距離的平方
359    T tmp=(p-p0).dot(n);
360    return tmp*tmp/n.abs2();
361  }
362  point3D<T> projection(const point3D<T> &p)const{
363    return p-n*(p-p0).dot(n)/n.abs2();
364  }
365  point3D<T> line_intersection(const line3D<T> &l)const{
366    T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該平面
367    return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
368  }
369  line3D<T> plane_intersection(const plane &pl)const{
370    point3D<T> e=n.cross(pl.n),v=n.cross(e);
371    T tmp=pl.n.dot(v);//等於0表示平行或重合該平面
372    point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/tmp);
373    return line3D<T>(q,q+e);
374  }
375 };
376 template<typename T>
377 struct triangle3D{
378  point3D<T> a,b,c;
379  triangle3D(){}
380  triangle3D(const point3D<T> &a,const point3D<T> &b,const
          point3D<T> &c):a(a),b(b),c(c){}
381  bool point_in(const point3D<T> &p)const{//點在該平面上的投
          影在三角形中
382    return line3D<T>(b,c).same_side(p,a)&&line3D<T>(a,c).
          same_side(p,b)&&line3D<T>(a,b).same_side(p,c);
383  }
384 };
385 template<typename T>
386 struct tetrahedron{//四面體
387  point3D<T> a,b,c,d;
388  tetrahedron(){}
389  tetrahedron(const point3D<T> &a,const point3D<T> &b,const
          point3D<T> &c,const point3D<T> &d):a(a),b(b),c(c),d(d)
          {}
390  T volume6()const{//體積的六倍
391    return (d-a).dot((b-a).cross(c-a));
392  }
393  point3D<T> centroid()const{
394    return (a+b+c+d)/4;
395  }
396  bool point_in(const point3D<T> &p)const{
397    return triangle3D<T>(a,b,c).point_in(p)&&triangle3D<T>(c,
          d,a).point_in(p);
398  }
399 };
400 template<typename T>
401 struct convexhull3D{
402  static const int MAXN=1005;
403  struct face{
404    int a,b,c;
405    face(int a,int b,int c):a(a),b(b),c(c){}
406  };
407  vector<point3D<T>> pt;
408  vector<face> ans;
409  int fid[MAXN][MAXN];
410  void build(){
411    int n=pt.size();
412    ans.clear();
413    memset(fid,0,sizeof(fid));
414    ans.emplace_back(0,1,2);//注意不能共線
415    ans.emplace_back(2,1,0);
416    int ftop = 0;
417    for(int i=3, ftop=1; i<n; ++i,++ftop){
418      vector<face> next;
419      for(auto &f:ans){
420        T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.
              c]-pt[f.a]));
421        if(d<=0) next.push_back(f);
422        int ff=0;
423        if(d>0) ff=ftop;
424        else if(d<0) ff=-ftop;
425        fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
426      }
427      for(auto &f:ans){
428        if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
429          next.emplace_back(f.a,f.b,i);
430        if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
431          next.emplace_back(f.b,f.c,i);
432        if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
```

```
433            next.emplace_back(f.c,f.a,i);
434        }
435        ans=next;
436      }
437    }
438    point3D<T> centroid()const{
439      point3D<T> res(0,0,0);
440      T vol=0;
441      for(auto &f:ans){
442        T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
443        res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
444        vol+=tmp;
445      }
446      return res/(vol*4);
447    }
448 };
```

4.3 HyperbolaGeometry

```
1  #define x first
2  #define y second
3  // 看要做整數運算還是浮點數運算 (甚至分數運算,請自行實作加減
      乘除)
4  // 若是整數運算,某些運算可能失真 (如求兩直線交點)
5  #define T double
6
7  // 兩個 eq 選一個
8  inline bool eq(double a, double b) { return abs(a - b) < 1e
      -7; }
9  inline bool eq(long long a, long long b) { return a == b; }
10
11 #define point vec
12 struct vec {
13   T x, y;   // 向量或坐標的 x,y 值
14   vec operator+(vec o) { return {x + o.x, y + o.y}; }
15   vec operator-(vec o) { return {x - o.x, y - o.y}; }
16   vec operator*(T o) { return {x * o, y * o}; }
17   vec operator/(T o) { return {x / o, y / o}; }
18   T operator%(vec o) { return x * o.x + y * o.y; }    //
        內積
19   T operator*(vec o) { return x * o.y - y * o.x; }    //
        外積
20   T abs() { return x * x + y * y; }                   //
        絕對值平方
21   bool samedir(vec o) { return eq(x * o.y, y * o.x); }  //
        兩向量方向是否相同或相反
22 };
23 vec makevec(point src, point dst) { return {dst.x - src.x,
     dst.y - src.y}; }
24
25 #define seg line
26 struct line {
27   point s, t;   // 此直線經過 s,t;或此線段始於 s 且止於 t
28   vec d;        // 此直線的向量
29   T a, b, c;    // ax+by=c
30
31   line(point p, point q) {   // 此直線經過 p,q;或此線段為始
        於 p 且止於 q
32     s = p, t = q, d = makevec(p, q);
33     a = p.y - q.y, b = q.x - p.x, c = a * p.x + b * p.y;
34   }
```

```
35   // 點是否在直線上
36   bool passLine(point p) { return d.samedir(p - s); }
37   bool passSeg(point p) { // 點是否在線段上
38     vec ap = makevec(s, p), bp = makevec(t, p);
39     return passLine(p) && ap % bp < 0;
40   }
41   // 兩直線是否重合
42   bool sameLine(line o) { return d.samedir(o.d) && passLine
        (o.s); }
43   // 兩直線是否平行且不重合
44   bool para(line o) { return d.samedir(o.d) && !passLine(o.
        s); }
45   point proj(point p) {   // 求某點在此直線上的投影座標
46     vec e = {p - s};
47     T t = e % d / d.abs();
48     vec dst = {d.x * t, d.y * t};
49     return s + dst;
50   }
51   // 點與直線距離平方
52   T dist2(point p) { return (proj(p) - p).abs(); }
53   // 兩平行直線距離平方
54   T dist2(line o) { return (o.proj(s) - s).abs(); }
55   // 此直線是否將兩點隔開
56   bool split(point p, point q) { return (a * p.x + b * p.y
        < 0) != (a * q.x + b * q.y < 0); }
57   // 兩非平行線段是否相交
58   bool meet(seg o) { return split(o.s, o.t) && o.split(s, t
        ); }
59   point intersect(line o) {   // 兩非平行直線相交座標
60     return {(c * o.b - b * o.c) / (a * o.b - b * o.a),
61             (a * o.c - c * o.a) / (a * o.b - b * o.a)};
62   }
63   double cosangle(line o) {   // 兩直線夾角之 cos 值
64     return (d % o.d) / (sqrt(d.abs() * o.d.abs()));
65   }
66 };
67
68 #define rr (r * r)   // 半徑平方
69 #define usevars                                        \
     //打字加速
70   double x1 = c.x, x2 = o.c.x, y1 = c.y, y2 = o.c.y;    \
71   double r1 = r, r2 = o.r, r12 = r1 * r1, r22 = r2 * r2; \
72   double dx = x2 - x1, dy = y2 - y1, dd = dx * dx + dy * dy
     , d = sqrt(dd);
73 const double PI = acos(-1);
74 struct circle {
75   point c;    // 圓心
76   double r;   // 半徑
77   // 求直線與圓的交點並回傳交點數量。若有兩點,存於 ans1 與
        ans2,若有一點,存於 ans1。
78   int meetLine(line l, point& ans1, point& ans2) {
79     double d2 = l.dist2(c);
80     if (eq(d2, rr)) return ans1 = l.proj(c), 1;   // 交於
          一點
81     if (d2 > rr) return 0;                         // 無交
          點
82     l = {l.s - c, l.t - c};
83     double s = l.a * l.a + l.b * l.b, w = rr - l.c * l.c
          / s, m = sqrt(w / s);
84     double x = -l.a * l.c / s, y = -l.b * l.c / s;
85     ans1 = {x + l.b * m, y - l.a * m}, ans2 = {x - l.b *
          m, y + l.a * m};
86     ans1 = ans1 + c, ans2 = ans2 + c;
```

```
87     return 2;
88   }
89   // 求線段與圓的交點並回傳交點數量。
90   int meetSeg(seg l, point& ans1, point& ans2) {
91     int res = meetLine(l, ans1, ans2);
92     if (res == 0) return 0;
93     if (res == 1) return l.passSeg(ans1);
94     return (int)l.passSeg(ans1) + l.passSeg(ans2);
95   }
96   // 求圓與圓的交點並回傳交點數量。
97   int meetCircle(circle o, point& ans1, point& ans2) {
98     usevars;
99     if (d > r1 + r2) return 0;          // 互斥
100    if (d < abs(r1 - r2)) return 0;   // 完全包含
101    point A = {(x1 + x2) / 2, (y1 + y2) / 2};
102    double f = (r12 - r22) / (2 * dd);
103    point B = {dx * f, dy * f};
104    double h = (r12 - r22);
105    f = sqrt(2 * (r12 + r22) / dd - h * h / (dd * dd) -
          1) / 2;
106    point C = {dy * f, -dx * f};
107    ans1 = A + B + C, ans2 = A + B - C;
108    return eq(d, r1 + r2) ? 1 : 2;
109  }
110  double coverArea(circle o) { // 求兩圓重疊部分面積
111    if (r < o.r) return o.coverArea(*this);
112    usevars;
113    if (d > r1 + r2) return 0;               // 互斥
114    if (d < abs(r1 - r2)) return PI * r2 * r2;   // 完全包
          含
115    double d1 = (r12 - r22 + dd) / (2 * d), d2 = d - d1;
116    return r12 * acos(d1 / r1) - d1 * sqrt(r12 - d1 * d1)
          + r22 * acos(d2 / r2) - d2 * sqrt(r22 - d2 * d2
          );
117  }
118 };
119
120 double len(point a, point b) { return sqrt((a - b).abs()); }
     // 打字加速
121 struct tri {
122   point a, b, c;
123   T area2() { return abs((b - a) * (c - a)); }  // 求面積之
        兩倍
124   point barycenter() { return (a + b + c) / 3; }  // 重心
125   point perpencenter() { return barycenter() * 3 -
        circumcenter() * 2; }  // 垂心
126   point circumcenter() {  // 外心
127     point p1 = (a + b) / 2, p2 = {p1.x - a.y + b.y, p1.y
          + a.x - b.x};
128     line u = {p1, p2};
129     p1 = (a + c) / 2, p2 = {p1.x - a.y + c.y, p1.y + a.x
          - c.x};
130     line v = {p1, p2};
131     return u.intersect(v);
132   }
133   point incentre() {  // 內心
134     T A = len(b, c), B = len(a, c), C = len(a, b);
135     point p = {A * a.x + B * b.x + C * c.x, A * a.y + B *
          b.y + C * c.y};
136     return p / (A + B + C);
137   }
138   // 費馬點
```

```
139    // 若有一角 >= 120 (cos(x) <= -0.5)，費馬點為該角對應的
       // 點
140    // 否則三角型三條邊對外做正三角形，得到三個頂點 A', B', C
       // '
141    // 費馬點為 AA' BB' CC' 三線之交點
142 };
```

4.4 MinRect

```
1  // 全部浮點數運算，先製作凸包，然後呼叫 minrect
2  typedef long double dd;
3  typedef pair<dd, dd> pii;
4  #define x first
5  #define y second
6  #define in inline
7  #define cp const pii&
8  #define op operator
9  #define ab (cp a, cp b)
10 const dd eps = 1e-8;
11 in pii op+ab { return {a.x + b.x, a.y + b.y}; }
12 in pii op-ab { return {a.x - b.x, a.y - b.y}; }
13 in pii op*(cp p, dd v) { return {v * p.x, v * p.y}; }
14 in dd  op^ab { return a.x * b.x + a.y * b.y; }
15 in dd  op*ab { return a.x * b.y - a.y * b.x; }
16 in dd  op%ab {
17     dd dx = a.x - b.x, dy = a.y - b.y;
18     return dx * dx + dy * dy;
19 }
20 in dd crzf(cp o, cp a, cp b) { return (a - o) * (b - o); }
21 in dd dotf(cp o, cp a, cp b) { return (a - o) ^ (b - o); }
22
23 #define judge \
24     crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= eps
25 vector<pii> makepoly(vector<pii>& pp) {
26     sort(pp.begin(), pp.end());
27     pp.erase(unique(pp.begin(), pp.end()), pp.end());
28     int n = pp.size(); vector<pii> ret;
29     for (int i = 0; i < n; i++) {
30         while (ret.size() >= 2 && judge) ret.pop_back();
31         ret.push_back(pp[i]);
32     }
33     for (int i = n - 2, s = ret.size() + 1; i >= 0; i--) {
34         while (ret.size() >= s && judge) ret.pop_back();
35         ret.push_back(pp[i]);
36     }
37     if (n >= 2) ret.pop_back(); return ret;
38 }
39
40 // 給凸包，問最小覆蓋矩形面積以及該矩形頂點座標 (存於 rec)
41 // 。頂點座標按照凸包製作方式排序。如果不需要矩形座標，把跟
42 // rec 有關的程式碼移除。
43 #define xx(i) ((i + 1) % n)
44 in pii foot(cp s1, cp s2, cp q) {
45 return s1 + (s2 - s1) * dotf(s1, s2, q) * (1 / (s1 % s2));
46 }
47 dd minrect(const vector<pii>& poly, vector<pii>& rec) {
48     int n = poly.size(); if (n < 3) return 0;
49     dd minn = 1e50; rec.resize(4);
50     int j = 1, k = 1, r;
51     for (int i = 0; i < n; i++) {
52         while (crzf(poly[i], poly[xx(i)], poly[xx(j)]) -
53                crzf(poly[i], poly[xx(i)], poly[j]) > -eps)
54             j = xx(j);
55         while (dotf(poly[i], poly[xx(i)], poly[xx(k)]) -
56                dotf(poly[i], poly[xx(i)], poly[k]) > -eps)
57             k = xx(k);
58         if (i == 0) r = k;
59         while (dotf(poly[i], poly[xx(i)], poly[xx(r)]) -
60                dotf(poly[i], poly[xx(i)], poly[r]) < eps)
61             r = xx(r);
62         dd a = crzf(poly[i], poly[xx(i)], poly[j]) *
63                (dotf(poly[i], poly[xx(i)], poly[k]) -
64                 dotf(poly[i], poly[xx(i)], poly[r])) /
65                (poly[i] % poly[xx(i)]);
66         a = abs(a); if (a < minn) { minn = a;
67             rec[0] = foot(poly[i], poly[xx(i)], poly[r]);
68             rec[1] = foot(poly[i], poly[xx(i)], poly[k]);
69             pii toss = foot(poly[i], poly[xx(i)], poly[j]);
70             rec[2] = poly[j] + rec[0] - toss;
71             rec[3] = poly[j] + rec[1] - toss;
72         }
73     }
74     rec = makepoly(rec); return minn;
75 }
```

4.5 Rectangle_Union_Area

```
1  const int maxn = 1e5 + 10;
2  struct rec{
3      int t, b, l, r;
4  } r[maxn];
5  int n, cnt[maxn << 2];
6  long long st[maxn << 2], ans = 0;
7  vector<int> x, y;
8  vector<pair<pair<int, int>, pair<int, int>>> v;
9  void modify(int t, int l, int r, int ql, int qr, int v) {
10     if (ql <= l && r <= qr) cnt[t] += v;
11     else {
12         int m = (l + r) >> 1;
13         if (qr <= m) modify(t << 1, l, m, ql, qr, v);
14         else if (ql >= m) modify(t << 1 | 1, m, r, ql, qr, v)
               ;
15         else modify(t << 1, l, m, ql, m, v), modify(t << 1 |
                1, m, r, m, qr, v);
16     }
17     if (cnt[t]) st[t] = y[r] - y[l];
18     else if (r - l == 1) st[t] = 0;
19     else st[t] = st[t << 1] + st[t << 1 | 1];
20 }
21 int main() {
22     cin >> n;
23     for (int i = 0; i < n; i++) {
24         cin >> r[i].l >> r[i].r >> r[i].b >> r[i].t;
25         if (r[i].l > r[i].r) swap(r[i].l, r[i].r);
26         if (r[i].b > r[i].t) swap(r[i].b, r[i].t);
27         x.push_back(r[i].l);
28         x.push_back(r[i].r);
29         y.push_back(r[i].b);
30         y.push_back(r[i].t);
31     }
32     sort(x.begin(), x.end());
33     sort(y.begin(), y.end());
34     x.erase(unique(x.begin(), x.end()), x.end());
35     y.erase(unique(y.begin(), y.end()), y.end());
36     for (int i = 0; i < n; i++) {
37         r[i].l = lower_bound(x.begin(), x.end(), r[i].l) - x.
               begin();
38         r[i].r = lower_bound(x.begin(), x.end(), r[i].r) - x.
               begin();
39         r[i].b = lower_bound(y.begin(), y.end(), r[i].b) - y.
               begin();
40         r[i].t = lower_bound(y.begin(), y.end(), r[i].t) - y.
               begin();
41         v.emplace_back(make_pair(r[i].l, 1), make_pair(r[i].b
               , r[i].t));
42         v.emplace_back(make_pair(r[i].r, -1), make_pair(r[i].
               b, r[i].t));
43     }
44     sort(v.begin(), v.end(), [](pair<pair<int, int>, pair<int
           , int>> a, pair<pair<int, int>, pair<int, int>> b){
45         if (a.first.first != b.first.first) return a.first.
               first < b.first.first;
46         return a.first.second > b.first.second;
47     });
48     for (int i = 0; i < v.size(); i++) {
49         if (i) ans += (x[v[i].first.first] - x[v[i - 1].first
               .first]) * st[1];
50         modify(1, 0, y.size(), v[i].second.first, v[i].second
               .second, v[i].first.second);
51     }
52     cout << ans << '\n';
53     return 0;
54 }
```

4.6 SmallestCircle

```
1  using PT = point<T>;
2  using CPT = const PT;
3  PT circumcenter(CPT &a, CPT &b, CPT &c) {
4      PT u = b-a, v =c-a;
5      T c1 = u.abs2()/2, c2 = v.abs2()/2;
6      T d = u.cross(v);
7      return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*c2-v.x*c1)/d);
8  }
9  void solve(PT p[], int n, PT &c, T &r2){
10     random_shuffle(p,p+n);
11     c = p[0]; r2 = 0; // c,r2 = 圓心,半徑平方
12     for(int i=1; i<n; i++)
13         if( (p[i]-c).abs2() > r2) {
14             c=p[i]; r2=0;
15             for(int j=0; j<i; j++)
16                 if( (p[j]-c).abs2() > r2) {
17                     c.x = (p[i].x+p[j].x)/2;
18                     c.y = (p[i].y+p[j].y)/2;
19                     r2 = (p[j]-c).abs2();
20                     for(int k=0; k<j; k++)
21                         if( (p[k]-c).abs2() > r2) {
22                             c = circumcenter(p[i], p[j], p[k]);
23                             r2 = (p[i]-c).abs2();
24                         }
25                 }
26         }
27 }
```

4.7 旋轉卡尺

```cpp
typedef pair<ll, ll> pii;
#define x first
#define y second
#define ii (i + 1) % n  // 打字加速！
inline pii operator-(const pii& a, const pii& b) {
    return {a.x - b.x, a.y - b.y};
}  // const 不可省略
inline ll operator*(const pii& a, const pii& b) {
    return a.x * b.y - a.y * b.x;
}
inline ll crzf(const pii& o, const pii& a, const pii& b) {
    return (a - o) * (b - o);
}
inline ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
// 給平面上任意個點，求其凸包。返回順序為逆時針。此方法會移除
//     重複點。
#define jud \
    crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= 0
vector<pii> makepoly(vector<pii>& pp) {
    int n = pp.size();
    sort(pp.begin(), pp.end());
    pp.erase(unique(pp.begin(), pp.end()), pp.end());
    vector<pii> ret;
    for (int i = 0; i < n; i++) {
        while (ret.size() >= 2 && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    for (int i = n - 2, t = ret.size() + 1; i >= 0; i--) {
        while (ret.size() >= t && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    if (n >= 2) ret.pop_back();
    return ret;
}
// (shoelace formula)
// 給凸包，問其面積「的兩倍」。若凸包少於三個點，回傳零。
ll area(vector<pii>& poly) {
    int n = poly.size();
    ll ret = 0;
    for (int i = 0; i < n; i++)
        ret += (poly[i].x * poly[ii].y);
    for (int i = 0; i < n; i++)
        ret -= (poly[i].y * poly[ii].x);
    return ret;
}
// 給凸包，問其兩點最遠距離「的平方」。若要問平面上任意個點的
//     兩點最遠
// 距離，請先轉成凸包。若凸包少於兩個點，回傳零。
#define kk (k + 1) % n
ll maxdist(vector<pii>& poly) {
    int k = 1, n = poly.size();
    if (n < 2) return 0;
    if (n == 2) return dd(poly[0], poly[1]);
    ll ret = 0;
    for (int i = 0; i < n; i++) {
        while (abs(crzf(poly[kk], poly[i], poly[ii])) >=
                   abs(crzf(poly[k], poly[i], poly[ii])))
            k = kk;
        ret = max(ret, max(dd(poly[i], poly[k]),
                           dd(poly[ii], poly[k])));
    }
```

```cpp
    return ret;
}
```

5 Graph

5.1 BCC_edge

```cpp
邊雙連通
任意兩點間至少有兩條不重疊的路徑連接，找法：
1. 標記出所有的橋
2. 對全圖進行 DFS，不走橋，每一次 DFS 就是一個新的邊雙連通
// from BCW
struct BccEdge {
    static const int MXN = 100005;
    struct Edge { int v,eid; };
    int n,m,step,par[MXN],dfn[MXN],low[MXN];
    vector<Edge> E[MXN];
    DisjointSet djs;
    void init(int _n) {
        n = _n; m = 0;
        for (int i=0; i<n; i++) E[i].clear();
        djs.init(n);
    }
    void add_edge(int u, int v) {
        E[u].PB({v, m});
        E[v].PB({u, m});
        m++;
    }
    void DFS(int u, int f, int f_eid) {
        par[u] = f;
        dfn[u] = low[u] = step++;
        for (auto it:E[u]) {
            if (it.eid == f_eid) continue;
            int v = it.v;
            if (dfn[v] == -1) {
                DFS(v, u, it.eid);
                low[u] = min(low[u], low[v]);
            } else {
                low[u] = min(low[u], dfn[v]);
            }
        }
    }
    void solve() {
        step = 0;
        memset(dfn, -1, sizeof(int)*n);
        for (int i=0; i<n; i++) {
            if (dfn[i] == -1) DFS(i, i, -1);
        }
        djs.init(n);
        for (int i=0; i<n; i++) {
            if (low[i] < dfn[i]) djs.uni(i, par[i]);
        }
    }
} graph;
```

5.2 LCA

```cpp
/* 三種 0/1-based。 只支援無向樹 */
/* Time: O(N+Q) Space: O(N^2) online */
class SsadpTarjan {
    private:
    int n;
    vector<int> par, dep; vector<vector<int>> ca;
    int dfs(int u, vector<vector<int>>& edge, int d) {
        dep[u] = d;
        for (int a = 0; a < n; a++)
            if (dep[a] != -1)
                ca[a][u] = ca[u][a] = parent(a);
        for (int a : edge[u]) {
            if (dep[a] != -1) continue;
            dfs(a, edge, d + 1);
            par[a] = u;
        }
    }
    int parent(int x) {
        if (par[x] == x) return x;
        return par[x] = parent(par[x]);
    }
    public:
    SsadpTarjan(vector<vector<int>>& edge, int root)
        : n(edge.size()) {
        dep.assign(n, -1); par.resize(n);
        ca.assign(n, vector<int>(n));
        for (int i = 0; i < n; i++) par[i] = i;
        dfs(root, edge, 0);
    }
    int lca(int a, int b) { return ca[a][b]; }
    int dist(int a, int b) {
        return dep[a] + dep[b] - 2 * dep[ca[a][b]];
    }
};
/* Time: O(N+Q)  Space: O(N+Q) only offline */
#define x first
#define y second
class OfflineTarjan {
    private:
    vector<int> par, anc, dep, ans, rank;
    vector<vector<pii>> qry;
    vector<vector<int>>& edge; // 安全考量可把 & 去掉
    int root, n;
    void merge(int a, int b) {
        a = parent(a), b = parent(b);
        if (rank[a] < rank[b]) swap(a, b);
        else if (rank[a] == rank[b]) rank[a]++;
        par[b] = a;
    }
    void dfs(int u, int d) {
        anc[parent(u)] = u, dep[u] = d;
        for (int a : edge[u]) {
            if (dep[a] != -1) continue;
            dfs(a, d + 1);
            merge(a, u);
            anc[parent(u)] = u;
        }
        for (auto q : qry[u])
            if (dep[q.first] != -1)
                ans[q.second] = anc[parent(q.first)];
    }
    int parent(int x) {
        if (par[x] == x) return x;
        return par[x] = parent(par[x]);
    }
```

```
66    void solve(vector<pii>& query) {
67        dep.assign(n, -1), rank.assign(n, 0);
68        par.resize(n), anc.resize(n), qry.resize(n);
69        for (int i = 0; i < n; i++) anc[i] = par[i] = i;
70        ans.resize(query.size());
71        for (int i = 0; i < query.size(); i++) {
72            auto& q = query[i];
73            qry[q.first].emplace_back(q.second, i);
74            qry[q.second].emplace_back(q.first, i);
75        }
76        dfs(root, 0);
77    }
78  public:
79    // edge 是傳 reference ，完成所有查詢不可改。
80    OfflineTarjan(vector<vector<int>>& edge, int root)
81        : edge(edge), root(root), n(edge.size()) {}
82    // 離線查詢， query 陣列包含所有詢問 {src, dst} 。呼叫一
          次無
83    // 論 query 量多少，複雜度都是 O(N) 。所以應盡量只呼叫一
          次。
84    vector<int> lca(vector<pii>& query) {
85        solve(query); return ans;
86    }
87    vector<int> dist(vector<pii>& query) {
88        solve(query);
89        for (int i = 0; i < query.size(); i++) {
90            auto & q = query[i];
91            ans[i] = dep[q.first] + dep[q.second]
92                    - 2 * dep[ans[i]];
93        } return ans;
94    }
95 };
96 /* Udchen Time: O(QlgN) Space: O(NlgN) 。支援非離線。*/
97 class SparseTableTarjan {
98   private:
99    int maxlg;
100    vector<vector<int>> anc;
101    vector<int> dep;
102    void dfs(int u, vector<vector<int>>& edge, int d) {
103        dep[u] = d;
104        for (int i = 1; i < maxlg; i++)
105            if (anc[u][i - 1] == -1) break;
106            else anc[u][i] = anc[anc[u][i - 1]][i - 1];
107        for (int a : edge[u]) {
108            if (dep[a] != -1) continue;
109            anc[a][0] = u;
110            dfs(a, edge, d + 1);
111        }
112    }
113  public:
114    SparseTableTarjan(vector<vector<int>>& edge, int root) {
115        int n = edge.size();
116        maxlg = ceil(log2(n));
117        anc.assign(n, vector<int>(maxlg, -1));
118        dep.assign(n, -1);
119        dfs(root, edge, 0);
120    }
121    int lca(int a, int b) {
122        if (dep[a] > dep[b]) swap(a, b);
123        for (int k = 0; dep[b] - dep[a]; k++)
124            if (((dep[b] - dep[a]) >> k) & 1) b = anc[b][k];
125        if (a == b) return a;
126        for (int k = maxlg - 1; k >= 0; k--)
127            if (anc[a][k] != anc[b][k])
128                a = anc[a][k], b = anc[b][k];
```

```
129            return anc[a][0];
130    }
131    int dist(int a, int b) {
132        return dep[a] + dep[b] - 2 * dep[lca(a, b)];
133    }
134 };
```

5.3 MahattanMST

```
1  #define REP(i,n) for(int i=0;i<n;i++)
2  typedef long long LL;
3  const int N=200100;
4  int n,m;
5  struct PT {int x,y,z,w,id;} p[N];
6  inline int dis(const PT &a,const PT &b){return abs(a.xb.x)+
      abs(a.y-b.y);}
7  inline bool cpx(const PT &a,const PT &b)
8  {return a.x!=b.x? a.x>b.x:a.y>b.y;}
9  inline bool cpz(const PT &a,const PT &b){return a.z<b.z;}
10 struct E{int a,b,c;}e[8*N];
11 bool operator<(const E&a,const E&b){return a.c<b.c;}
12 struct Node{ int L,R,key; } node[4*N];
13 int s[N];
14 int F(int x) {return s[x]==x ? x : s[x]=F(s[x]); }
15 void U(int a,int b) {s[F(b)]=F(a);}
16 void init(int id,int L,int R) {
17     node[id] = (Node){L,R,-1};
18     if(L==R)return;
19     init(id*2,L,(L+R)/2);
20     init(id*2+1,(L+R)/2+1,R);
21 }
22 void ins(int id,int x) {
23     if(node[id].key==-1 || p[node[id].key].w>p[x].w)
24         node[id].key=x;
25     if(node[id].L==node[id].R) return;
26     if(p[x].z<=(node[id].L+node[id].R)/2) ins(id*2,x);
27     else ins(id*2+1,x);
28 }
29 int Q(int id,int L,int R){
30     if(R<node[id].L || L>node[id].R)return -1;
31     if(L<=node[id].L && node[id].R<=R)return node[id].key;
32     int a=Q(id*2,L,R),b=Q(id*2+1,L,R);
33     if(b==-1 || (a!=-1 && p[a].w<p[b].w)) return a;
34     else return b;
35 }
36 void calc() {
37     REP(i,n) {
38         p[i].z = p[i].y-p[i].x;
39         p[i].w = p[i].x+p[i].y;
40     }
41     sort(p,p+n,cpz);
42     int cnt = 0, j, k;
43     for(int i=0; i<n; i=j){
44         for(j=i+1; p[j].z==p[i].z && j<n; j++);
45         for(k=i, cnt++; k<j; k++) p[k].z = cnt;
46     }
47     init(1,1,cnt);
48     sort(p,p+n,cpx);
49     REP(i,n) {
50         j=Q(1,p[i].z,cnt);
51         if(j!=-1) e[m++] = (E){p[i].id, p[j].id, dis(p[i],p[j
              ])};
52         ins(1,i);
```

```
53    }
54 }
55 LL MST() {
56     LL r=0;
57     sort(e, e+m);
58     REP(i, m) {
59         if(F(e[i].a)==F(e[i].b)) continue;
60         U(e[i].a, e[i].b);
61         r += e[i].c;
62     }
63     return r;
64 }
65 int main() {
66     int ts;
67     scanf("%d", &ts);
68     while (ts--) {
69         m = 0;
70         scanf("%d",&n);
71         REP(i,n) {scanf("%d%d",&p[i].x,&p[i].y);p[i].id=s[i]=
              i;}
72         calc();
73         REP(i,n)p[i].y= -p[i].y;
74         calc();
75         REP(i,n)swap(p[i].x,p[i].y);
76         calc();
77         REP(i,n)p[i].x=-p[i].x;
78         calc();
79         printf("%lld\n",MST()*2);
80     }
81     return 0;
82 }
```

5.4 MinMeanCycle

```
1  #include<cfloat> //for DBL_MAX
2  int dp[MAXN][MAXN]; // 1-base,O(NM)
3  vector<tuple<int,int,int>> edge;
4  double mmc(int n){ //allow negative weight
5      const int INF = 0x3f3f3f3f;
6      for(int t=0; t<n; ++t){
7          memset(dp[t+1],0x3f,sizeof(dp[t+1]));
8          for(const auto &e:edge) {
9              int u, v, w; tie(u,v,w) = e;
10             dp[t+1][v] = min(dp[t+1][v],dp[t][u]+w);
11         }
12     }
13     double res = DBL_MAX;
14     for(int u=1; u<=n; ++u) {
15         if(dp[n][u]==INF) continue;
16         double val = -DBL_MAX;
17         for(int t=0;t<n;++t)
18             val = max(val,(dp[n][u]-dp[t][u])*1.0/(n-t));
19         res = min(res,val);
20     } return res;
21 }
```

5.5 Tarjan

```
1 割點
```

```
2  點 u 為割點 if and only if 滿足 1. or 2.
3  1. u 為樹根，且 u 有多於一個子樹。
4  2. u 不為樹根，且滿足存在 (u,v) 為樹枝邊 (或稱父子邊，即 u 為
       v 在搜索樹中的父親)，使得 DFN(u) <= Low(v)。
5  ----------------------------------------------------
6  橋
7  一條無向邊 (u,v) 是橋 if and only if (u,v) 為樹枝邊，且滿足
       DFN(u) < Low(v)。
8  // 0 base
9  struct TarjanSCC{
10     static const int MAXN = 1000006;
11     int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
12     vector<int> G[MAXN];
13     stack<int> stk;
14     bool ins[MAXN];
15     void tarjan(int u) {
16         dfn[u] = low[u] = ++count;
17         stk.push(u);
18         ins[u] = true;
19         for(auto v:G[u]) {
20             if(!dfn[v]) {
21                 tarjan(v);
22                 low[u] = min(low[u], low[v]);
23             } else if(ins[v]) {
24                 low[u] = min(low[u], dfn[v]);
25             }
26         }
27         if(dfn[u] == low[u]) {
28             int v;
29             do {
30                 v = stk.top(); stk.pop();
31                 scc[v] = scn;
32                 ins[v] = false;
33             } while(v != u);
34             scn++;
35         }
36     }
37     void getSCC(){
38         memset(dfn,0,sizeof(dfn));
39         memset(low,0,sizeof(low));
40         memset(ins,0,sizeof(ins));
41         memset(scc,0,sizeof(scc));
42         count = scn = 0;
43         for(int i = 0 ; i < n ; i++ )
44             if(!dfn[i]) tarjan(i);
45     }
46 } SCC;
```

5.6 Two_SAT

```
1  const int N = 5010 * 2;  // 變數最大數量的兩倍
2  namespace Two_Sat {
3  vector<int> a[N], b[N], stk;
4  int vis[N], res[N];
5  void dfs(int u, vector<int>* g, int sc) {
6      vis[u] = 1, res[u] = sc;
7      for (int v : g[u]) if (!vis[v]) dfs(v, g, sc);
8      if (g == a) stk.push_back(u);
9  }
10 // 先呼叫 imply 來設定約束，然後呼叫 scc 跑分析。
11 // var[x] 的真值對應 i = x * 2 ; var[x] 的假值對應 i = x * 2
       + 1
```

```
12 // e.g. 若 var[3] 為真則 var[6] 必為假，則呼叫 imply(6, 13)
13 void imply(int u, int v) {  // if u then v
14     a[u].push_back(v), b[v].push_back(u);
15 }
16 // 跑 two_sat ，回傳 true 表示有解。解答存於 Two_Sat::res
17 // e.g. 若 res[13] == 1 表 var[6] 必為假
18 // e.g. 若 res[0] == 1 且 res[1] == 1 ，表 var[0] 必為真且必
       為假，矛盾，無解。
19 int scc(int n /*變數實際數量的兩倍*/) {
20     memset(vis, 0, sizeof(vis));
21     for (int i = 0; i < n; i++) if (!vis[i]) dfs(i, a, -1);
22     memset(vis, 0, sizeof(vis));
23     int sc = 0;
24     while (!stk.empty()) {
25         if (!vis[stk.back()]) dfs(stk.back(), b, sc++);
26         stk.pop_back();
27     }
28     for (int i = 0; i < n; i += 2) {
29         if (res[i] == res[i + 1]) return 0;
30         if (res[i] > res[i + 1]) res[i] = 1, res[i + 1] = 0;
31         else res[i] = 0, res[i + 1] = 1;
32     }
33     return 1;
34 }
35 }  // namespace Two_Sat
```

6 Math

6.1 ax+by=gcd(a,b)

```
1  // 給 a,b ，解 ax+by=gcd(a,b)
2  typedef pair<ll, ll> pii;
3  pii extgcd(ll a, ll b) {
4      if (b == 0) return {1, 0};
5      ll k = a / b;
6      pii p = extgcd(b, a - k * b);
7      return {p.second, p.first - k * p.second};
8  }
```

6.2 Discrete_sqrt

```
1  int order(ll b, ll p) {
2      if (__gcd(b, p) != 1) return -1;
3      int ret = 2;
4      while (++ret)
5          if (fastpow(b, ret, p) == 1) break;
6      return ret;
7  }
8  // 把 fastpow 也抄過來，會用到。
9  // 問 (x^2 = y) mod p 的解。回傳 -1 表示 x 無解。
10 ll dsqrt(ll y, ll p) {
11     if (__gcd(y, p) != 1) return -1;
12     if (fastpow(y, (p - 1 / 2), p) == p - 1) return -1;
13     int e = 0;
14     ll s = p - 1;
15     while (!(s & 1)) s >>= 1, e++;
16     int q = 2;
```

```
17     while (1)
18         if (fastpow(q, (p - 1) / 2, p) == p - 1)
19             break;
20         else q++;
21     ll x = fastpow(y, (s + 1) / 2, p);
22     ll b = fastpow(y, s, p);
23     ll g = fastpow(q, s, p);
24     while (1) {
25         int m;
26         for (m = 0; m < e; m++) {
27             int o = order(p, b);
28             if (o == -1) return -1;
29             if (o == fastpow(2, m, p)) break;
30         }
31         if (m == 0) return x;
32         x = x * fastpow(g, fastpow(2, e - m - 1), p) % p;
33         g = fastpow(g, fastpow(2, e - m), p), p);
34         b = b * g % p;
35         if (b == 1) return x;
36         e = m;
37     }
38 }
```

6.3 EulerFunction

```
1  // 查詢 phi(x) 亦即比 x 小且與 x 互質的數的數量。
2  int phi(int x) {
3      int r = x;
4      for (int p = 2; p * p <= x; p++) {
5          if (x % p == 0) {
6              while (x % p == 0) x /= p;
7              r -= r / p;
8          }
9      }
10     if (x > 1) r -= r / x;
11     return r;
12 }
13 // 查詢所有 phi(x)，x in [0, n) 回傳陣列。
14 vector<int> phi_in(int n) {
15     vector<bool> p(n, 1); vector<int> r(n);
16     for (int i = 0; i < n; i++) r[i] = i;
17     r[1] = p[0] = p[1] = 0;
18     for (int i = 2; i < n; i++) {
19         if (!p[i]) continue;
20         r[i]--;
21         for (int j = i * 2; j < n; j += i)
22             p[j] = 0, r[j] = r[j] / i * (i - 1);
23     } return r;
24 }
```

6.4 Expression

```
1  /*支援處理四則運算的工具。給四則運算的字串，檢查格式並計算
2    其值。如果格式不合法，會丟出錯誤。複雜度 O(字串長度) 。
3    支援的符號有四則運算和求餘數，先乘除後加減。可以使用括號
4    、或前置正負號。數字開頭可以為零或禁止為零。可以兼容或禁
5    止多重前置號 (例如 --1 視為 1 、+-+-1 視為 -1) 。
6    空字串視為不合法。運算範圍限於 long long。如果試圖除
7    以零或對零求餘也會丟出錯誤。*/
```

```
8  void req(bool b) { if (!b) throw ""; }
9  const int B = 2; // 可以調整成 B 進位
10 class Expr {
11     private:
12     deque<char> src;
13     Expr(const string& s) : src(s.begin(), s.end()) {}
14     inline char top() {
15         return src.empty() ? '\0' : src.front();
16     }
17     inline char pop() {
18         char c = src.front(); src.pop_front(); return c;
19     }
20     ll n() {
21         ll ret = pop() - '0';
22         // 若要禁止數字以 0 開頭，加上這行
23         // req(ret || !isdigit(top()));
24         while (isdigit(top())) ret = B * ret + pop() - '0';
25         return ret;
26     }
27     ll fac() {
28         if (isdigit(top())) return n();
29         if (top() == '-') { pop(); return -fac(); }
30         if (top() == '(') {
31             pop();
32             ll ret = expr(1);
33             req(pop() == ')');
34             return ret;
35         }
36         // 若要允許前置正號，加上這行
37         // if(top() == '+') { pop(); return fac(); }
38         throw "";
39     }
40     ll term() {
41         ll ret = fac(); char c = top();
42         while (c == '*' || c == '/' || c == '%') {
43             pop();
44             if (c == '*') ret *= fac();
45             else {
46                 ll t = fac(); req(t);
47                 if (c == '/') ret /= t; else ret %= t;
48             }
49             c = top();
50         } return ret;
51     }
52     ll expr(bool k) {
53         ll ret = term();
54         while (top() == '+' || top() == '-')
55             if (pop() == '+') ret += term();
56             else ret -= term();
57         req(top() == (k ? ')' : '\0'));
58         return ret;
59     }
60     public:
61     // 給定數學運算的字串，求其值。若格式不合法，丟出錯誤。
62     static ll eval(const string& s) {
63         // 若要禁止多重前置號，加上這四行
64         // req(s.find("--") == -1); // 禁止多重負號
65         // req(s.find("-+") == -1);
66         // req(s.find("+-") == -1);
67         // req(s.find("++") == -1);
68         return Expr(s).expr(0);
69     }
70 };
```

6.5 FFT

```
1  // int(complex.real() + 0.05)   // .imag()
2  template <typename T, typename VT = vector<complex<T>>>
3  struct FFT {
4      const T pi;
5      FFT(const T pi = acos((T)-1.0)) : pi(pi) {}
6      unsigned bit_reverse (unsigned a, int len) {
7          a = ((a&0x55555555U)<<1)  | ((a&0xAAAAAAAAU)>>1);
8          a = ((a&0x33333333U)<<2)  | ((a&0xCCCCCCCCU)>>2);
9          a = ((a&0x0F0F0F0FU)<<4)  | ((a&0xF0F0F0F0U)>>4);
10         a = ((a&0x00FF00FFU)<<8)  | ((a&0xFF00FF00U)>>8);
11         a = ((a&0x0000FFFFU)<<16) | ((a&0xFFFF0000U)>>16);
12         return a >> (32-len);
13     }
14     void fft (bool is_inv, VT &in, VT &out, int N) {
15         int bitlen = __lg(N), num = is_inv ? -1 : 1;
16         for(int i = 0; i < N; ++i)
17             out[bit_reverse(i, bitlen)] = in[i];
18         for(int step = 2, mh = 1; step <= N; step <<= 1, mh
                <<= 1){
19             for(int i = 0; i < mh; ++i){
20                 complex<T> wi = exp(complex<T>(0, i * num *
                       pi / mh));
21                 for(int j = i, k = i + mh; j < N; j += step,
                       k += step) {
22                     complex<T> u = out[j], t = wi * out[k];
23                     out[j] = u + t, out[k] = u - t;
24                 }
25             }
26         }
27         for (int i = 0; is_inv && i < N; ++i)
28             out[i] /= N;
29     }
30 };
31 int main () { // polynomial multiplication
32     FFT<double> F; int n = 4;
33     vector<complex<double>> a = {1, 2, 0, 0};
34     vector<complex<double>> b = {2, 3, 0, 0};
35     vector<complex<double>> a_fft(n), b_fft(n), ab_fft(n), ab(n
          );
36     F.fft(0, a, a_fft, 4), F.fft(0, b, b_fft, 4);
37     for (int i = 0; i < n; i++)
38         ab_fft[i] = a_fft[i] * b_fft[i];
39     F.fft(1, ab_fft, ab, n);
40     for (auto p : ab)
41         cout << int(p.real() + 1e-6) << " ";
42     return 0;
43 }
```

6.6 FindRealRoot

```
1  // an*x^n + ... + a1x + a0 = 0;
2  int sign(double x){
3      return x < -eps ? -1 : x > eps;
4  }
5  double get(const vector<double>&coef, double x){
6      double e = 1, s = 0;
7      for(auto i : coef) s += i*e, e *= x;
8      return s;
9  }
```

```
10 double find(const vector<double>&coef, int n, double lo,
          double hi){
11     double sign_lo, sign_hi;
12     if( !(sign_lo = sign(get(coef,lo))) ) return lo;
13     if( !(sign_hi = sign(get(coef,hi))) ) return hi;
14     if(sign_lo * sign_hi > 0) return INF;
15     for(int stp = 0; stp < 100 && hi - lo > eps; ++stp){
16         double m = (lo+hi)/2.0;
17         int sign_mid = sign(get(coef,m));
18         if(!sign_mid) return m;
19         if(sign_lo*sign_mid < 0) hi = m;
20         else lo = m;
21     }
22     return (lo+hi)/2.0;
23 }
24 vector<double> cal(vector<double>coef, int n){
25     vector<double>res;
26     if(n == 1){
27         if(sign(coef[1])) res.pb(-coef[0]/coef[1]);
28         return res;
29     }
30     vector<double>dcoef(n);
31     for(int i = 0; i < n; ++i) dcoef[i] = coef[i+1]*(i+1);
32     vector<double>droot = cal(dcoef, n-1);
33     droot.insert(droot.begin(), -INF);
34     droot.pb(INF);
35     for(int i = 0; i+1 < droot.size(); ++i){
36         double tmp = find(coef, n, droot[i], droot[i+1]);
37         if(tmp < INF) res.pb(tmp);
38     }
39     return res;
40 }
41 int main () {
42     vector<double>ve;
43     vector<double>ans = cal(ve, n);
44     // 視情況把答案 +eps，避免 -0
45 }
```

6.7 Fraction

```
1  #define cfl(str) (const frac& f) const { return str; }
2  #define cll(str) (ll l) const { return str; }
3  #define lfl(str) (ll l, const frac& f) { return str; }
4  #define ff inline frac operator
5  #define bb inline bool operator
6  #define fff inline friend frac operator
7  #define fbb inline friend bool operator
8
9  class frac {
10     private: ll x, y;
11     public:
12     frac() : x(0), y(1) {}
13     frac(ll v) : x(v), y(1) {}
14     frac(ll xx, ll yy, bool f = 0) : x(xx), y(yy) {
15         assert(y != 0);
16         if (!f) {
17             ll g = __gcd(x, y);
18             x /= g, y /= g;
19             if (y < 0) x *= -1, y *= -1;
20         }
21     }
22     // 以下斟酌使用，不必全抄
```

```
23        ff = (ll l) { return frac(l); }
24        ff - () const { return frac(-x, y, 1); }
25        ff ! () const {  // 倒數
26            return x > 0 ? frac(y, x, 1) : frac(-y, -x, 1);
27        }
28
29        bb >  cfl(x * f.y > y * f.x)
30        bb <  cfl(x * f.y < y * f.x)
31        bb <= cfl(x * f.y <= y * f.x)
32        bb >= cfl(x * f.y >= y * f.x)
33        bb == cfl(x == f.x && y == f.y)
34        bb != cfl(x != f.x || y != f.y)
35        ff +  cfl(frac(x * f.y + y * f.x, y * f.y))
36        ff -  cfl(frac(x * f.y - y * f.x, y * f.y))
37        ff *  cfl(frac(x * f.x, y * f.y))
38        ff /  cfl(frac(x * f.y, y * f.x))
39
40        bb >  cll(x > l * y)
41        bb <  cll(x < l * y)
42        bb >= cll(x >= l * y)
43        bb <= cll(x <= l * y)
44        bb == cll(x == l * y)
45        bb != cll(x != l * y)
46        ff +  cll(frac(x + l * y, y))
47        ff -  cll(frac(x - l * y, y))
48        ff *  cll(frac(l * x, y))
49        ff /  cll(frac(x, l * y))
50
51        fbb <  lfl(f > l)
52        fbb >  lfl(f < l)
53        fbb <= lfl(f >= l)
54        fbb >= lfl(f <= l)
55        fbb == lfl(f == l)
56        fbb != lfl(f != l)
57        fff +  lfl(f + l)
58        fff -  lfl(-f + l)
59        fff *  lfl(f * l)
60        fff /  lfl(!f * l)
61
62        inline operator double() { return (double)x / y; }
63        inline friend frac abs(const frac& f) {
64            return frac(abs(f.x), f.y, 1);
65        }
66        inline friend ostream& operator <<
67            (ostream & out, const frac& f) {
68            out << f.x;
69            if (f.y != 1) out << '/' << f.y;
70            return out;
71        }
72    };
```

6.8   Karatsuba

```
 1 // N is power of 2
 2 template<typename Iter>
 3 void DC(int N, Iter tmp, Iter A, Iter B, Iter res){
 4     fill(res,res+2*N,0);
 5     if (N<=32){
 6         for (int i=0; i<N; i++)
 7             for (int j=0; j<N; j++)
 8                 res[i+j] += A[i]*B[j];
 9         return;
10     }
```

```
11     int n = N/2;
12     auto a = A+n, b = A;
13     auto c = B+n, d = B;
14     DC(n,tmp+N,a,c,res+2*N);
15     for (int i=0; i<N; i++){
16         res[i+N] += res[2*N+i];
17         res[i+n] -= res[2*N+i];
18     }
19     DC(n,tmp+N,b,d,res+2*N);
20     for (int i=0; i<N; i++){
21         res[i] += res[2*N+i];
22         res[i+n] -= res[2*N+i];
23     }
24     auto x = tmp;
25     auto y = tmp+n;
26     for (int i=0; i<n; i++) x[i] = a[i]+b[i];
27     for (int i=0; i<n; i++) y[i] = c[i]+d[i];
28     DC(n,tmp+N,x,y,res+2*N);
29     for (int i=0; i<N; i++)
30         res[i+n] += res[2*N+i];
31 }
32 // DC(1<<16,tmp.begin(),A.begin(),B.begin(),res.begin());
```

6.9   Matrix

```
 1 struct Matrix {
 2     int r, c;
 3     vector<vector<ll>> m;
 4     Matrix(int r, int c): r(r), c(c), m(r, vector<ll>(c)) {}
 5     vector<ll> &operator[](int i) { return m[i]; }
 6     Matrix operator+(const Matrix &a) {
 7         Matrix rev(r, c);
 8         for (int i = 0; i < r; ++i)
 9             for (int j = 0; j < c; ++j)
10                 rev[i][j] = m[i][j] + a.m[i][j];
11         return rev;
12     }
13     Matrix operator-(const Matrix &a) {
14         Matrix rev(r, c);
15         for (int i = 0; i < r; ++i)
16             for (int j = 0; j < c; ++j)
17                 rev[i][j] = m[i][j] - a.m[i][j];
18         return rev;
19     }
20     Matrix operator*(const Matrix &a) {
21         Matrix rev(r, a.c);
22         Matrix tmp(a.c, a.r);
23         for (int i = 0; i < a.r; ++i)
24             for (int j = 0; j < a.c; ++j)
25                 tmp[j][i] = a.m[i][j];
26         for (int i = 0; i < r; ++i)
27             for (int j = 0; j < a.c; ++j)
28                 for (int k = 0; k < c; ++k)
29                     rev.m[i][j] += m[i][k] * tmp[j][k];
30         return rev;
31     }
32     // 回傳反矩陣。注意這是 const 方法所以原矩陣不受影響
33     Matrix inverse() const {
34         Matrix t(r, r + c);
35         for (int y = 0; y < r; y++) {
36             t.m[y][c + y] = 1;
37             for (int x = 0; x < c; x++) t.m[y][x] = m[y][x];
38         }
```

```
39         if (!t.gauss()) return Matrix(0, 0);
40         Matrix ret(c, r);
41         for (int y = 0; y < r; y++)
42             for (int x = 0; x < c; x++)
43                 ret[y][x] = t.m[y][c + x] / t.m[y][y];
44         return ret;
45     }
46     // 做高斯消去 (最高次係數應置於最左，常數應置於最右)
47     // 回傳 det。O(n^3)。如果不是方陣，回傳值無意義。
48     ll gauss() {
49         vector<ll> lazy(r, 1);
50         bool sign = false;
51         for (int i = 0; i < r; ++i) {
52             if (m[i][i] == 0) {
53                 int j = i + 1;
54                 while (j < r && !m[j][i]) j++;
55                 if (j == r) continue;
56                 m[i].swap(m[j]); sign = !sign;
57             }
58             for (int j = 0; j < r; ++j) {
59                 if (i == j) continue;
60                 lazy[j] = lazy[j] * m[i][i];
61                 ll mx = m[j][i];
62                 for (int k = 0; k < c; ++k)
63                     m[j][k] =
64                         m[j][k] * m[i][i] - m[i][k] * mx;
65             }
66         }
67         ll det = sign ? -1 : 1;
68         for (int i = 0; i < r; ++i) {
69             det = det * m[i][i] / lazy[i];
70             for (auto &j : m[i]) j /= lazy[i];
71         }
72         return det;
73     }
74 };
```

6.10   MillerRabin

```
 1 //From jacky860226
 2 typedef long long LL;
 3 inline LL mul(LL a,LL b,LL m){//a*b%m
 4     return (a%m)*(b%m)%m;
 5 }
 6 /*LL mul(LL a,LL b,LL m){//a*b%m
 7     a %= m, b %= m;
 8     LL y = (LL)((double)a*b/m+0.5); //fast for m < 2^58
 9     LL r = (a*b-y*m)%m;
10     return r<0 ? r+m : r;
11 }*/
12 template<typename T> T pow(T a,T b,T mod) { //a^b%mod
13     T ans = 1;
14     while (b) {
15         if(b&1) ans = mul(ans,a,mod);
16         a = mul(a,a,mod);
17         b >>= 1;
18     } return ans;
19 }
20 template<typename T> bool isprime(T n, int num) { //num = 3,7
21     int sprp[3] = {2,7,61}; //int範圍可解
22     //int llsprp[7] =
           {2,325,9375,28178,450775,9780504,1795265022}; //至少
```

```
          unsigned long long範圍
23   if(n==2) return true;
24   if(n<2 || n%2==0) return false;
25   //n-1 = u * 2^t
26   int t = 0;  T u = n-1;
27   while(u%2==0) u >>= 1, t++;
28   for(int i=0; i<num; i++) {
29       T a = sprp[i]%n;
30       if(a==0 || a==1 || a==n-1) continue;
31       T x = pow(a,u,n);
32       if(x==1 || x==n-1) continue;
33       for(int j=1; j<t; j++) {
34           x = mul(x,x,n);
35           if(x==1) return false;
36           if(x==n-1) break;
37       }
38       if(x!=n-1) return false;
39   } return true;
40 }
```

6.11 ModInv

```
1
2 // O(n) 求 1...n 對 p 的模逆元
3 ll precalc(int n, ll p){
4     for(int i = 1; i <= n; i++){
5         if(i == 1) inv[i] = 1;
6         else inv[i] = (p - (p / i) * inv[p % i] % p) % p;
7     }
8 }
```

6.12 NTT

```
1  template<typename T,typename VT=std::vector<T> >
2  struct NTT{
3    const T P,G;
4    NTT(T p=(1<<23)*7*17+1,T g=3):P(p),G(g){}
5    inline unsigned int bit_reverse(unsigned int a,int len){
6      a=((a&0x55555555U)<<1)|((a&0xAAAAAAAAU)>>1);
7      a=((a&0x33333333U)<<2)|((a&0xCCCCCCCCU)>>2);
8      a=((a&0x0F0F0F0FU)<<4)|((a&0xF0F0F0F0U)>>4);
9      a=((a&0x00FF00FFU)<<8)|((a&0xFF00FF00U)>>8);
10     a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
11     return a>>(32-len);
12   }
13   inline T pow_mod(T n,T k,T m){
14     T ans=1;
15     for(n=(n>=m?n%m:n);k;k>>=1){
16       if(k&1)ans=ans*n%m;
17       n=n*n%m;
18     } return ans;
19   }
20   inline void ntt(bool is_inv,VT &in,VT &out,int N){
21     int bitlen=std::__lg(N);
22     for(int i=0;i<N;++i)out[bit_reverse(i,bitlen)]=in[i];
23     for(int step=2,id=1;step<=N;step<<=1,++id){
24       T wn=pow_mod(G,(P-1)>>id,P),wi=1,u,t;
25       const int mh=step>>1;
26       for(int i=0;i<mh;++i){
27         for(int j=i;j<N;j+=step){
28           u = out[j], t = wi*out[j+mh]%P;
29           out[j] = u+t;
30           out[j+mh] = u-t;
31           if(out[j]>=P)out[j]-=P;
32           if(out[j+mh]<0)out[j+mh]+=P;
33         }
34         wi = wi*wn%P;
35       }
36     }
37     if(is_inv){
38       for(int i=1;i<N/2;++i)std::swap(out[i],out[N-i]);
39       T invn=pow_mod(N,P-2,P);
40       for(int i=0;i<N;++i)out[i]=out[i]*invn%P;
41     }
42   }
43 };
44 #endif
```

6.13 PrimeList

```
1  12721          13331        14341        75577
2  123457         222557       556679       880301
3  999983         1e6+99       1e9+9        2e9+99
4  1e12+39        1e15+37      1e9+7        1e7+19
5  1097774749     1076767633   100102021
6  999997771      1001010013   1000512343
7  987654361      999991231    999888733
8  98789101       987777733    999991921
9  1010101333     1010102101
10 2305843009213693951    4611686018427387847
11 9223372036854775783    18446744073709551557
```

6.14 SG

```
1  Anti Nim（取走最後一個石子者敗）：
2  先手必勝 if and only if
3  1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
4  2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。
5  -----------------------------------------------------
6  Anti-SG（決策集合為空的遊戲者贏）：
7  定義 SG 值為 0 時，遊戲結束，
8  則先手必勝 if and only if
9  1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
10 2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0
      。
11 -----------------------------------------------------
12 Sprague-Grundy：
13 1. 雙人、回合制
14 2. 資訊完全公開
15 3. 無隨機因素
16 4. 可在有限步內結束
17 5. 沒有和局
18 6. 雙方可採取的行動相同
19
20 SG(S) 的值為 0：後手(P)必勝
21 不為 0：先手(N)必勝
```

```
22 int mex(set S) {
23   // find the min number >= 0 that not in the S
24   // e.g. S = {0, 1, 3, 4} mex(S) = 2
25 }
26 state = []
27 int SG(A) {
28   if (A not in state) {
29     S = sub_states(A)
30     if( len(S) > 1 ) state[A] = reduce(operator.xor, [SG(B)
                for B in S])
31     else state[A] = mex(set(SG(B) for B in next_states(A)))
32   } return state[A]
33 }
```

6.15 Simplex

```
1  /*target:
2    max \sum_{j=1}^n A_{0,j}*x_j
3  condition:
4    \sum_{j=1}^n A_{i,j}*x_j <= A_{i,0} |i=1~m
5    x_j >= 0 |j=1~n
6  VDB = vector<double>*/
7  template<class VDB>
8  VDB simplex(int m,int n,vector<VDB> a){
9    vector<int> left(m+1), up(n+1);
10   iota(left.begin(), left.end(), n);
11   iota(up.begin(), up.end(), 0);
12   auto pivot = [&](int x, int y){
13     swap(left[x], up[y]);
14     auto k = a[x][y]; a[x][y] = 1;
15     vector<int> pos;
16     for(int j = 0; j <= n; ++j){
17       a[x][j] /= k;
18       if(a[x][j] != 0) pos.push_back(j);
19     }
20     for(int i = 0; i <= m; ++i){
21       if(a[i][y]==0 || i == x) continue;
22       k = a[i][y], a[i][y] = 0;
23       for(int j : pos) a[i][j] -= k*a[x][j];
24     }
25   };
26   for(int x,y;;){
27     for(int i=x=1; i <= m; ++i)
28       if(a[i][0]<a[x][0]) x = i;
29     if(a[x][0]>=0) break;
30     for(int j=y=1; j <= n; ++j)
31       if(a[x][j]<a[x][y]) y = j;
32     if(a[x][y]>=0) return VDB();//infeasible
33     pivot(x, y);
34   }
35   for(int x,y;;){
36     for(int j=y=1; j <= n; ++j)
37       if(a[0][j] > a[0][y]) y = j;
38     if(a[0][y]<=0) break;
39     x = -1;
40     for(int i=1; i<=m; ++i) if(a[i][y] > 0)
41       if(x == -1 || a[i][0]/a[i][y]
42         < a[x][0]/a[x][y]) x = i;
43     if(x == -1) return VDB();//unbounded
44     pivot(x, y);
45   }
46   VDB ans(n + 1);
47   for(int i = 1; i <= m; ++i)
```

```
48      if(left[i] <= n) ans[left[i]] = a[i][0];
49    ans[0] = -a[0][0];
50    return ans;
51  }
```

6.16 外星模運算

```
1   //a[0]^(a[1]^a[2]^...)
2   #define maxn 1000000
3   int euler[maxn+5];
4   bool is_prime[maxn+5];
5   void init_euler(){
6     is_prime[1] = 1; //一不是質數
7     for(int i=1; i<=maxn; i++) euler[i]=i;
8     for(int i=2; i<=maxn; i++) {
9       if(!is_prime[i]) { //是質數
10        euler[i]--;
11        for(int j=i<<1; j<=maxn; j+=i) {
12          is_prime[j]=1;
13          euler[j] = euler[j]/i*(i-1);
14        }
15      }
16    }
17  }
18  LL pow(LL a, LL b, LL mod) { //a^b%mod
19    LL ans=1;
20    for(; b; a=a*a%mod, b>>=1)
21      if(b&1) ans = ans*a%mod;
22    return ans;
23  }
24  bool isless(LL *a, int n, int k) {
25    if(*a==1)return k>1;
26    if(--n==0)return *a<k;
27    int next=0;
28    for(LL b=1;b<k;++next)
29      b *= *a;
30    return isless(a+1, n, next);
31  }
32  LL high_pow(LL *a, int n, LL mod){
33    if(*a==1||--n==0)return *a%mod;
34    int k = 0, r = euler[mod];
35    for(LL tma=1;tma!=pow(*a,k+r,mod);++k)
36      tma = tma*(*a)%mod;
37    if(isless(a+1,n,k))return pow(*a,high_pow(a+1,n,k),mod);
38    int tmd = high_pow(a+1,n,r), t = (tmd-k+r)%r;
39    return pow(*a,k+t,mod);
40  }
41  LL a[1000005]; int t,mod;
42  int main(){
43    init_euler();
44    scanf("%d", &t);
45    #define n 4
46    while(t--){
47      for(int i=0;i<n;++i)scanf("%lld", &a[i]);
48      scanf("%d", &mod);
49      printf("%lld\n", high_pow(a,n,mod));
50    }
51    return 0;
52  }
```

6.17 質因數分解

```
1   LL func(const LL n,const LL mod,const int c) {
2     return (LLmul(n,n,mod)+c+mod)%mod;
3   }
4   LL pollorrho(const LL n, const int c) {//循環節長度
5     LL a=1, b=1;
6     a=func(a,n,c)%n;
7     b=func(b,n,c)%n; b=func(b,n,c)%n;
8     while(gcd(abs(a-b),n)==1) {
9       a=func(a,n,c)%n;
10      b=func(b,n,c)%n; b=func(b,n,c)%n;
11    }
12    return gcd(abs(a-b),n);
13  }
14  void prefactor(LL &n, vector<LL> &v) {
15    for(int i=0;i<12;++i) {
16      while(n%prime[i]==0) {
17        v.push_back(prime[i]);
18        n/=prime[i];
19      }
20    }
21  }
22  void smallfactor(LL n, vector<LL> &v) {
23    if(n<MAXPRIME) {
24      while(isp[(int)n]) {
25        v.push_back(isp[(int)n]);
26        n/=isp[(int)n];
27      }
28      v.push_back(n);
29    } else {
30      for(int i=0;i<primecnt&&prime[i]*prime[i]<=n;++i) {
31        while(n%prime[i]==0) {
32          v.push_back(prime[i]);
33          n/=prime[i];
34        }
35      }
36      if(n!=1) v.push_back(n);
37    }
38  }
39  void comfactor(const LL &n, vector<LL> &v) {
40    if(n<1e9) {
41      smallfactor(n,v);
42      return;
43    }
44    if(Isprime(n)) {
45      v.push_back(n);
46      return;
47    }
48    LL d;
49    for(int c=3;;++c) {
50      d = pollorrho(n,c);
51      if(d!=n) break;
52    }
53    comfactor(d,v);
54    comfactor(n/d,v);
55  }
56  void Factor(const LL &x, vector<LL> &v) {
57    LL n = x;
58    if(n==1) { puts("Factor 1"); return; }
59    prefactor(n,v);
60    if(n==1) return;
61    comfactor(n,v);
62    sort(v.begin(),v.end());
63  }
```

```
64  void AllFactor(const LL &n,vector<LL> &v) {
65    vector<LL> tmp;
66    Factor(n,tmp);
67    v.clear();
68    v.push_back(1);
69    int len;
70    LL now=1;
71    for(int i=0;i<tmp.size();++i) {
72      if(i==0 || tmp[i]!=tmp[i-1]) {
73        len = v.size();
74        now = 1;
75      }
76      now*=tmp[i];
77      for(int j=0;j<len;++j)
78        v.push_back(v[j]*now);
79    }
80  }
```

7 String

7.1 AC 自動機

```
1   template<char L='a',char R='z'>
2   class ac_automaton{
3     struct joe{
4       int next[R-L+1], fail, efl, ed, cnt_dp, vis;
5       joe():ed(0),cnt_dp(0),vis(0){
6         for(int i=0; i<=R-L; i++) next[i]=0;
7       }
8     };
9   public:
10    std::vector<joe> S;
11    std::vector<int> q;
12    int qs,qe,vt;
13    ac_automaton():S(1),qs(0),qe(0),vt(0){}
14    void clear(){
15      q.clear();
16      S.resize(1);
17      for(int i=0; i<=R-L; i++) S[0].next[i] = 0;
18      S[0].cnt_dp = S[0].vis = qs = qe = vt = 0;
19    }
20    void insert(const char *s){
21      int o = 0;
22      for(int i=0,id; s[i]; i++){
23        id = s[i]-L;
24        if(!S[o].next[id]){
25          S.push_back(joe());
26          S[o].next[id] = S.size()-1;
27        }
28        o = S[o].next[id];
29      }
30      ++S[o].ed;
31    }
32    void build_fail(){
33      S[0].fail = S[0].efl = -1;
34      q.clear();
35      q.push_back(0);
36      ++qe;
37      while(qs!=qe){
38        int pa = q[qs++], id, t;
39        for(int i=0;i<=R-L;i++){
```

```
40        t = S[pa].next[i];
41        if(!t)continue;
42        id = S[pa].fail;
43        while(~id && !S[id].next[i]) id = S[id].fail;
44        S[t].fail = ~id ? S[id].next[i] : 0;
45        S[t].efl = S[S[t].fail].ed ? S[t].fail : S[S[t].fail
          ].efl;
46        q.push_back(t);
47        ++qe;
48      }
49    }
50 }
51 /*DP出每個前綴在字串s出現的次數並傳回所有字串被s匹配成功的
      次數O(N+M)*/
52 int match_0(const char *s){
53   int ans = 0, id, p = 0, i;
54   for(i=0; s[i]; i++){
55     id = s[i]-L;
56     while(!S[p].next[id] && p) p = S[p].fail;
57     if(!S[p].next[id])continue;
58     p = S[p].next[id];
59     ++S[p].cnt_dp;/*匹配成功則它所有後綴都可以被匹配(DP計算
        )*/
60   }
61   for(i=qe-1; i>=0; --i){
62     ans += S[q[i]].cnt_dp * S[q[i]].ed;
63     if(~S[q[i]].fail) S[S[q[i]].fail].cnt_dp += S[q[i]].
        cnt_dp;
64   }
65   return ans;
66 }
67 /*多串匹配走efl邊並傳回所有字串被s匹配成功的次數O(N*M^1.5)
      */
68 int match_1(const char *s)const{
69   int ans = 0, id, p = 0, t;
70   for(int i=0; s[i]; i++){
71     id = s[i]-L;
72     while(!S[p].next[id] && p) p = S[p].fail;
73     if(!S[p].next[id])continue;
74     p = S[p].next[id];
75     if(S[p].ed) ans += S[p].ed;
76     for(t=S[p].efl; ~t; t=S[t].efl){
77       ans += S[t].ed;/*因為都走efl邊所以保證匹配成功*/
78     }
79   }
80   return ans;
81 }
82 /*枚舉(s的子字串∩A)的所有相異字串各恰一次並傳回次數O(N*M
      ^(1/3))*/
83 int match_2(const char *s){
84   int ans=0, id, p=0, t;
85   ++vt;
86   /*把戳記vt+=1，只要vt沒溢位，所有S[p].vis==vt就會變成
        false
87   這種利用vt的方法可以O(1)歸零vis陣列*/
88   for(int i=0; s[i]; i++){
89     id = s[i]-L;
90     while(!S[p].next[id]&&p)p = S[p].fail;
91     if(!S[p].next[id])continue;
92     p = S[p].next[id];
93     if(S[p].ed && S[p].vis!=vt){
94       S[p].vis = vt;
95       ans += S[p].ed;
96     }
```

```
97      for(t=S[p].efl; ~t && S[t].vis!=vt; t=S[t].efl){
98        S[t].vis = vt;
99        ans += S[t].ed;/*因為都走efl邊所以保證匹配成功*/
100     }
101   }
102   return ans;
103 }
104 /*把AC自動機變成真的自動機*/
105 void evolution(){
106   for(qs=1; qs!=qe;){
107     int p = q[qs++];
108     for(int i=0; i<=R-L; i++)
109       if(S[p].next[i]==0) S[p].next[i] = S[S[p].fail].next[
          i];
110   }
111 }
112 };
```

7.2 BWT

```
1  const int N = 8;            // 字串長度
2  int s[N+N+1] = "suffixes";  // 字串，後面預留一倍空間。
3  int sa[N];                  // 後綴陣列
4  int pivot;
5  int cmp(const void* i, const void* j) {
6      return strncmp(s+*(int*)i, s+*(int*)j, N);
7  }
8  // 此處便宜行事，採用 O(N²logN) 的後綴陣列演算法。
9  void BWT() {
10     strncpy(s + N, s, N);
11     for (int i=0; i<N; ++i) sa[i] = i;
12     qsort(sa, N, sizeof(int), cmp);
13     // 當輸入字串的所有字元都相同，必須當作特例處理。
14     // 或者改用stable sort。
15     for (int i=0; i<N; ++i)
16         cout << s[(sa[i] + N-1) % N];
17     for (int i=0; i<N; ++i)
18         if (sa[i] == 0) {
19             pivot = i;
20             break;
21         }
22 }
23 // Inverse BWT
24 const int N = 8;            // 字串長度
25 char t[N+1] = "xuffessi";   // 字串
26 int pivot;
27 int next[N];
28 void IBWT() {
29     vector<int> index[256];
30     for (int i=0; i<N; ++i)
31         index[t[i]].push_back(i);
32     for (int i=0, n=0; i<256; ++i)
33         for (int j=0; j<index[i].size(); ++j)
34             next[n++] = index[i][j];
35     int p = pivot;
36     for (int i=0; i<N; ++i)
37         cout << t[p = next[p]];
38 }
```

7.3 Kmp

```
1  // KMP fail function.
2  int* kmp_fail(string& s) {
3      int* f = new int[s.size()]; int p = f[0] = -1;
4      for (int i = 1; s[i]; i++) {
5          while (p != -1 && s[p + 1] != s[i]) p = f[p];
6          if (s[p + 1] == s[i]) p++;
7          f[i] = p;
8      }
9      return f;
10 }
11 // 問 sub 在 str 中出現幾次。
12 int kmp_count(string& str, string& sub) {
13     int* fail = kmp_fail(sub); int p = -1, ret = 0;
14     for (int i = 0; i < str.size(); i++) {
15         while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
16         if (sub[p + 1] == str[i]) p++;
17         if (p == sub.size() - 1) p = fail[p], ret++;
18     }
19     delete[] fail; return ret;
20 }
21 // 問 sub 在 str 第一次出現的開頭 index 。 -1 表示找不到。
22 int kmp(string& str, string& sub) {
23     int* fail = kmp_fail(sub);
24     int i, j = 0;
25     while (i < str.size() && j < sub.size()) {
26         if (sub[j] == str[i]) i++, j++;
27         else if (j == 0) i++;
28         else j = fail[j - 1] + 1;
29     }
30     delete[] fail;
31     return j == sub.size() ? (i - j) : -1;
32 }
```

7.4 LPS

```
1  char t[1001];              // 原字串
2  char s[1001 * 2];          // 穿插特殊字元之後的t
3  int z[1001 * 2], L, R;     // 源自Gusfield's Algorithm
4  // 由a往左、由b往右，對稱地作字元比對。
5  int extend(int a, int b) {
6      int i = 0;
7      while (a-i>=0 && b+i<N && s[a-i] == s[b+i]) i++;
8      return i;
9  }
10 void longest_palindromic_substring() {
11     int N = strlen(t);
12     // t穿插特殊字元，存放到s。
13     // （實際上不會這麼做，都是細算索引值。）
14     memset(s, '.', N*2+1);
15     for (int i=0; i<N; ++i) s[i*2+1] = t[i];
16     N = N*2+1;
17     // s[N] = '\0';     // 可做可不做
18     // Manacher's Algorithm
19     z[0] = 1; L = R = 0;
20     for (int i=1; i<N; ++i) {
21         int ii = L - (i - L);   // i的映射位置
22         int n = R + 1 - i;
23         if (i > R) {
```

```
24          z[i] = extend(i, i);
25          L = i;
26          R = i + z[i] - 1;
27        } else if (z[ii] == n)  {
28          z[i] = n + extend(i-n, i+n);
29          L = i;
30          R = i + z[i] - 1;
31        } else z[i] = min(z[ii], n);
32      }
33      // 尋找最長迴文子字串的長度。
34      int n = 0, p = 0;
35      for (int i=0; i<N; ++i)
36        if (z[i] > n)  n = z[p = i];
37      // 記得去掉特殊字元。
38      cout << "最長迴文子字串的長度是 " << (n-1) / 2;
39      // 印出最長迴文子字串，記得別印特殊字元。
40      for (int i=p-z[p]+1; i<=p+z[p]-1; ++i)
41        if (i & 1)  cout << s[i];
42 }
```

7.5 Manacher

```
1  // Longest Palindromic Substring
2  int manacher (string str) { // O(n)
3    int len = (s.length() << 1) | 1;
4    vector<int> z(len);
5    string s(len, '$');
6    for (int i = 1; i < len; i += 2)
7      s[i] = str[i >> 1];
8    int r = 0, p = 0, ans = 0;
9    for (int i = 0, j = p << 1; i < len; i++, j--) {
10     z[i] = (i >= r) ? 1 : min(z[j], r - i +1);
11     while(0 <= i - z[i] && i + z[i] < len && s[i - z[i]] == s
            [i + z[i]])
12       z[i]++;
13     if (r < i + z[i] - 1)
14       r = i + z[i] - 1, p = i;
15     ans = max(ans, z[i]);
16   }
17   return ans - 1;
18 }
```

7.6 RollHash

```
1  // 問 pat 在 str 第一次出現的開頭 index 。-1 表示找不到。
2  int rollhash(string& str, string& pat) {
3    const ll x = 1e6 + 99;  // 隨意大質數，建議 1e6
4    const ll m = 1e9 + 9;   // 隨意大質數，建議 1e9
5    assert(pat.size());     // pat 不能是空字串
6    ll xx = 1, sh = 0;
7    for (char c : pat)
8      sh = (sh * x + c) % m, xx = xx * x % m;
9    deque<ll> hash = {0};
10   int ret = 0;
11   for (char c : str) {
12     hash.push_back((hash.back() * x + c) % m);
13     if (hash.size() <= pat.size()) continue;
14     ll h = hash.back() - hash.front() * xx;
```

```
15     h = (h % m + m) % m;
16     if (h == sh) return ret;
17     hash.pop_front();
18     ret++;
19   } return -1;
20 }
```

7.7 suffix_array

```
1  // qsort suffix array, 0-based only, O(T * log^2 T)
2  const int N = ? ;  // 字串最大長度
3  namespace SA {
4    int sa[N], t0[N], t1[N];
5    struct CMP {
6      int *r, n, X;
7      bool operator()(int i, int j) {
8        if (r[i] != r[j]) return r[i] < r[j];
9        int a = (i + n < X) ? r[i + n] : -1;
10       int b = (j + n < X) ? r[j + n] : -1;
11       return a < b;
12     }
13   };
14   // str = 字串，可為 vector 或 string 或 char[] 等
15   // n = 字串長(含$)
16   // 結果存在 SA::sa
17   template <typename T>
18   void build(const T &str) {
19     int n = str.size();
20     int *a = t0, *aa = t1;
21     for (int i = 0; i < n; i++) sa[i] = i, a[i] = str[i];
22     for (int m = 2; m <= n; m *= 2) {
23       CMP cmp = {a, m / 2, n};
24       sort(sa, sa + n, cmp);
25       int r = 0;
26       aa[sa[0]] = r;
27       for (int i = 1; i < n; i++) {
28         if (cmp(sa[i - 1], sa[i])) r++;
29         aa[sa[i]] = r;
30       }
31       swap(a, aa);
32       if (r == n - 1) break;
33     }
34   }
35 }  // namespace SA
36
37 // 卦長的 IS suffix array ，0-based only
38 // N = 字串最大長度 , A = 最大字元 ascii
39 // 複雜度 O(N+A)
40 const int N = ?, A = ?;
41 namespace SA {
42 #define pushS(x) sa[--b[s[x]]] = x
43 #define pushL(x) sa[b[s[x]]++] = x
44 #define induce_sort(v)                                         \
45   {                                                            \
46     fill_n(sa, n, 0);                                          \
47     copy_n(bb, A, b);                                          \
48     for (i = n1 - 1; ~i; --i) pushS(v[i]);                     \
49     copy_n(bb, A - 1, b + 1);                                  \
50     for (i = 0; i < n; ++i)                                    \
51       if (sa[i] && !t[sa[i] - 1]) pushL(sa[i] - 1);  \
52     copy_n(bb, A, b);                                          \
53     for (i = n - 1; ~i; --i)                                   \
```

```
54       if (sa[i] && t[sa[i] - 1]) pushS(sa[i] - 1);  \
55   }
56 template <typename T>
57 void sais(const T s, int n, int *sa, int *bb, int *p, bool *t
       , int A) {
58   int *r = p + n, *s1 = p + n / 2, *b = bb + A;
59   int n1 = 0, i, j, x = t[n - 1] = 1, y = r[0] = -1, cnt =
       -1;
60   for (i = n - 2; ~i; --i) t[i] = (s[i] == s[i + 1] ? t[i +
       1] : s[i] < s[i + 1]);
61   for (i = 1; i < n; ++i) r[i] = t[i] && !t[i - 1] ? (p[n1]
       = i, n1++) : -1;
62   fill_n(bb, A, 0);
63   for (i = 0; i < n; ++i) ++bb[s[i]];
64   for (i = 1; i < A; ++i) bb[i] += bb[i - 1];
65   induce_sort(p);
66   for (i = 0; i < n; ++i)
67     if (~(x = r[sa[i]]))
68       j = y < 0 || memcmp(s + p[x], s + p[y], (p[x + 1]
           - p[x]) * sizeof(s[0])), s1[y = x] = cnt +=
           j;
69   if (cnt + 1 < n1)
70     sais(s1, n1, sa, b, r, t + n, cnt + 1);
71   else
72     for (i = 0; i < n1; ++i) sa[s1[i]] = i;
73   for (i = 0; i < n1; ++i) s1[i] = p[sa[i]];
74   induce_sort(s1);
75 }
76 int sa[N];
77 int b[N + A], p[N * 2];
78 bool t[N * 2];
79 // 計算 suffix array ，字串須為 char[] 或 int[]， 不可為
       string 或 vector
80 // s = 字串
81 // n = 字串長度(含$)
82 // 結果存在 SA::sa
83 template <typename T>
84 void build(const T s, int n) { sais(s, n, sa, b, p, t, A); }
85 }  // namespace SA
```

7.8 Trie

```
1  class Trie {
2  private:
3    struct Node {
4      int cnt = 0, sum = 0;
5      Node *tr[128] = {};
6      ~Node() {
7        for (int i = 0; i < 128; i++)
8          if (tr[i]) delete tr[i];
9      }
10   };
11   Node *root;
12 public:
13   void insert(char *s) {
14     Node *ptr = root;
15     for (; *s; s++) {
16       if (!ptr->tr[*s]) ptr->tr[*s] = new Node();
17       ptr = ptr->tr[*s];
18       ptr->sum++;
19     }
20     ptr->cnt++;
```

Left column:

```
21        }
22    inline int count(char *s) {
23        Node *ptr = find(s);
24        return ptr ? ptr->cnt : 0;
25    }
26    Node *find(char *s) {
27        Node *ptr = root;
28        for (; *s; s++) {
29            if (!ptr->tr[*s]) return 0;
30            ptr = ptr->tr[*s];
31        } return ptr;
32    }
33    bool erase(char *s) {
34        Node *ptr = find(s);
35        if (!ptr) return false;
36        int num = ptr->cnt;
37        if (!num) return false;
38        ptr = root;
39        for (; *s; s++) {
40            Node *tmp = ptr;
41            ptr = ptr->tr[*s];
42            ptr->sum -= num;
43            if (!ptr->sum) {
44                delete ptr;
45                tmp->tr[*s] = 0;
46                return true;
47            }
48        }
49    }
50    Trie() { root = new Node(); }
51    ~Trie() { delete root; }
52 };
```

7.9 Z

```
1 void z_build(string &s, vector<int> &z) {
2     int bst = z[0] = 0;
3     for (int i = 1; s[i]; i++) {
4         if (z[bst] + bst < i) z[i] = 0;
5         else z[i] = min(z[bst] + bst - i, z[i - bst]);
6         while (s[z[i]] == s[i + z[i]]) z[i]++;
7         if (z[i] + i > z[bst] + bst) bst = i;
8     }
9 }
10 // Queries how many times s appears in t
11 int z_match(string &s, string &t) {
12     int ans = 0;
13     int lens = s.length(), lent = t.length();
14     vector<int> z(lens + lent + 1);
15     string st = s + "$" + t;
16     z_build(st, z);
17     for (int i = lens + 1; i <= lens + lent; i++)
18         if (z[i] == lens) ans++;
19     return ans;
20 }
```

Middle column:

8 DP

8.1 Bounded_Knapsack

```
1  namespace {
2      static const int MAXW = 1000005;
3      static const int MAXN = 1005;
4      struct BB {
5          int w, v, c;
6          BB(int w = 0, int v = 0, int c = 0): w(w), v(v), c(c)
                {}
7          bool operator<(const BB &x) const {
8              return w * c  < x.w * x.c;
9          }
10     };
11     static int run(BB A[], int dp[], int W, int N) {
12         static int MQ[MAXW][2];
13         for (int i = 0, sum = 0; i < N; i++) {
14             int w = A[i].w, v = A[i].v, c = A[i].c;
15             sum = min(sum + w*c, W);
16             for (int j = 0; j < w; j++) {
17                 int l = 0, r = 0;
18                 MQ[l][0] = 0, MQ[l][1] = dp[j];
19                 for (int k = 1, tw = w+j, tv = v; tw <= sum
                        && k <= c; k++, tw += w, tv += v) {
20                     int dpv = dp[tw] - tv;
21                     while (l <= r && MQ[r][1] <= dpv) r--;
22                     r++;
23                     MQ[r][0] = k, MQ[r][1] = dpv;
24                     dp[tw] = max(dp[tw], MQ[l][1] + tv);
25                 }
26                 for (int k = c+1, tw = (c+1)*w+j, tv = (c+1)*
                        v; tw <= sum; k++, tw += w, tv += v) {
27                     if (k - MQ[l][0] > c) l++;
28                     int dpv = dp[tw] - tv;
29                     while (l <= r && MQ[r][1] <= dpv) r--;
30                     r++;
31                     MQ[r][0] = k, MQ[r][1] = dpv;
32                     dp[tw] = max(dp[tw], MQ[l][1] + tv);
33                 }
34             }
35         }
36     }
37     static int knapsack(int C[][3], int N, int W) { // O(WN)
38         vector<BB> A;
39         for (int i = 0; i < N; i++) {
40             int w = C[i][0], v = C[i][1], c = C[i][2];
41             A.push_back(BB(w, v, c));
42         }
43         assert(N < MAXN);
44         static int dp1[MAXW+1], dp2[MAXW+1];
45         BB Ar[2][MAXN];
46         int ArN[2] = {};
47         memset(dp1, 0, sizeof(dp1[0])*(W+1));
48         memset(dp2, 0, sizeof(dp2[0])*(W+1));
49         sort(A.begin(), A.end());
50         int sum[2] = {};
51         for (int i = 0; i < N; i++) {
52             int ch = sum[1] < sum[0];
53             Ar[ch][ArN[ch]] = A[i];
54             ArN[ch]++;
55             sum[ch] = min(sum[ch] + A[i].w*A[i].c, W);
56         }
```

Right column:

```
57             run(Ar[0], dp1, W, ArN[0]);
58             run(Ar[1], dp2, W, ArN[1]);
59             int ret = 0;
60             for (int i = 0, j = W, mx = 0; i <= W; i++, j--) {
61                 mx = max(mx, dp2[i]);
62                 ret = max(ret, dp1[j] + mx);
63             }
64             return ret;
65         }
66 }
67 int main() {
68     int W, N;
69     assert(scanf("%d %d", &W, &N) == 2);
70     int C[MAXN][3];
71     for (int i = 0; i < N; i++)
72         assert(scanf("%d %d %d", &C[i][1], &C[i][0], &C[i
                ][2]) == 3);
73     printf("%d\n", knapsack(C, N, W));
74     return 0;
75 }
```

8.2 DP_1D1D

```
1  int t, n, L, p;
2  char s[MAXN][35];
3  ll sum[MAXN] = {0};
4  long double dp[MAXN] = {0};
5  int prevd[MAXN] = {0};
6  long double pw(long double a, int n) {
7      if ( n == 1 ) return a;
8      long double b = pw(a, n/2);
9      if ( n & 1 ) return b*b*a;
10     else return b*b;
11 }
12 long double f(int i, int j) {
13     // cout << (sum[i] - sum[j]+i-j-1-L) << endl;
14     return pw(abs(sum[i] - sum[j]+i-j-1-L), p) + dp[j];
15 }
16 struct INV {
17     int L, R, pos;
18 };
19 INV stk[MAXN*10];
20 int top = 1, bot = 1;
21 void update(int i) {
22     while ( top > bot && i < stk[top].L && f(stk[top].L, i) <
                f(stk[top].L, stk[top].pos) ) {
23         stk[top - 1].R = stk[top].R;
24         top--;
25     }
26     int lo = stk[top].L, hi = stk[top].R, mid, pos = stk[top
                ].pos;
27     // if ( i >= lo ) lo = i + 1;
28     while ( lo != hi ) {
29         mid = lo + (hi - lo) / 2;
30         if ( f(mid, i) < f(mid, pos) ) hi = mid;
31         else lo = mid + 1;
32     }
33     if ( hi < stk[top].R ) {
34         stk[top + 1] = (INV) { hi, stk[top].R, i };
35         stk[top++].R = hi;
36     }
37 }
38 int main() {
```

```
39      cin >> t;
40      while ( t-- ) {
41          cin >> n >> L >> p;
42          dp[0] = sum[0] = 0;
43          for ( int i = 1 ; i <= n ; i++ ) {
44              cin >> s[i];
45              sum[i] = sum[i-1] + strlen(s[i]);
46              dp[i] = numeric_limits<long double>::max();
47          }
48          stk[top] = (INV) {1, n + 1, 0};
49          for ( int i = 1 ; i <= n ; i++ ) {
50              if ( i >= stk[bot].R ) bot++;
51              dp[i] = f(i, stk[bot].pos);
52              update(i);
53              // cout << (ll) f(i, stk[bot].pos) << endl;
54          }
55          if ( dp[n] > 1e18 ) {
56              cout << "Too hard to arrange" << endl;
57          } else {
58              vector<PI> as;
59              cout << (ll)dp[n] << endl;
60          }
61      } return 0;
62  }
```

8.3 LCIS

```
1  vector<int> LCIS(vector<int> a, vector<int> b) {
2      int n = a.size(), m = b.size();
3      int dp[LEN][LEN] = {}, pre[LEN][LEN] = {};
4      for(int i=1; i<=n; i++) {
5          int p = 0;
6          for(int j=1; j<=m; j++)
7              if(a[i-1]!=b[j-1]) {
8                  dp[i][j] = dp[i-1][j], pre[i][j] = j;
9                  if( a[i-1]>b[j-1] && dp[i-1][j]>dp[i-1][p] )
10                     p = j;
11             } else {
12                 dp[i][j] = dp[i-1][p]+1, pre[i][j] = p;
13             }
14     }
15     int len = 0, p = 0;
16     for(int j=1; j<=m; j++)
17         if(dp[n][j]>len) len = dp[n][j], p = j;
18     vector<int> ans;
19     for(int i=n; i>=1; i--) {
20         if(a[i-1]==b[p-1] && p!=pre[i][p])
21             ans.push_back(b[p-1]);
22         p = pre[i][p];
23     }
24     reverse(ans.begin(), ans.end());
25     return ans;
26 }
```

9 Other

9.1 BuiltIn

```
1  //gcc專用
2  //unsigned int ffs
3  //unsigned long ffsl
4  //unsigned long long ffsll
5  unsigned int x; scanf("%u",&x)
6  printf("右起第一個1的位置");
7  printf("%d\n",__builtin_ffs(x));
8  printf("左起第一個1之前0的個數:");
9  printf("%d\n",__builtin_clz(x));
10 printf("右起第一個1之後0的個數:");
11 printf("%d\n",__builtin_ctz(x));
12 printf("1的個數:");
13 printf("%d\n",__builtin_popcount(x));
14 printf("1的個數的奇偶性:");
15 printf("%d\n",__builtin_parity(x));
```

9.2 CNF

```
1  #define MAXN 55
2  struct CNF{
3      int s,x,y;//s->xy | s->x, if y==-1
4      int cost;
5      CNF(){}
6      CNF(int s,int x,int y,int c):s(s),x(x),y(y),cost(c){}
7  };
8  int state;//規則數量
9  map<char,int> rule;//每個字元對應到的規則,小寫字母為終端字符
10 vector<CNF> cnf;
11 void init(){
12     state=0;
13     rule.clear();
14     cnf.clear();
15 }
16 void add_to_cnf(char s,const string &p,int cost){
17     //加入一個s -> <p>的文法,代價為cost
18     if(rule.find(s)==rule.end())rule[s]=state++;
19     for(auto c:p)if(rule.find(c)==rule.end())rule[c]=state++;
20     if(p.size()==1){
21         cnf.push_back(CNF(rule[s],rule[p[0]],-1,cost));
22     }else{
23         int left=rule[s];
24         int sz=p.size();
25         for(int i=0;i<sz-2;++i){
26             cnf.push_back(CNF(left,rule[p[i]],state,0));
27             left=state++;
28         }
29         cnf.push_back(CNF(left,rule[p[sz-2]],rule[p[sz-1]],cost));
30     }
31 }
32 vector<long long> dp[MAXN][MAXN];
33 vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會有無限
                                   小的情形
34 void relax(int l,int r,const CNF &c,long long cost,bool neg_c
       =0){
35     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][
           c.s])){
36         if(neg_c||neg_INF[l][r][c.x]){
37             dp[l][r][c.s]=0;
38             neg_INF[l][r][c.s]=true;
39         }else dp[l][r][c.s]=cost;
```

```
40     }
41 }
42 void bellman(int l,int r,int n){
43     for(int k=1;k<=state;++k)
44         for(auto c:cnf)
45             if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
46 }
47 void cyk(const vector<int> &tok){
48     for(int i=0;i<(int)tok.size();++i){
49         for(int j=0;j<(int)tok.size();++j){
50             dp[i][j]=vector<long long>(state+1,INT_MAX);
51             neg_INF[i][j]=vector<bool>(state+1,false);
52         }
53         dp[i][i][tok[i]]=0;
54         bellman(i,i,tok.size());
55     }
56     for(int r=1;r<(int)tok.size();++r){
57         for(int l=r-1;l>=0;--l){
58             for(int k=l;k<r;++k)
59                 for(auto c:cnf)
60                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c
                           .cost);
61             bellman(l,r,tok.size());
62         }
63     }
64 }
```

9.3 HeapsAlgo

```
1  void heaps(int k, vector<int> &s, int n) {
2      if (k == 1) {
3          // for (int i = 0; i < n; i++)
4          //   cout << s[i] << " \n"[i == n - 1];
5          return;
6      }
7      for (int i = 0; i < k - 1; ++i) {
8          heaps(k - 1, s, n);
9          if (k & 1) swap(s[0], s[k - 1]);
10         else swap(s[i], s[k - 1]);
11     }
12     heaps(k - 1, s, n);
13 }
14 void permutation(int n) {
15     vector<int> v(n);
16     for (int i = 0; i < n; i++) v[i] = i;
17     heaps(n, v, n);
18 }
```

9.4 Reminder

9.4.1 Complexity

1. LCA

Method	Time	Space	離線
SsadpTarjan	$O(N + Q)$	$O(N^2)$	不須離線
OfflineTarjan	$O(N + Q)$	$O(N + Q)$	須離線
SparseTable	$O(N + Q \log N)$	$O(N \log N)$	不須離線

2. Dinic

```
Graph.......... Space........ Time
Gernal        O(V+E)    O(EV^2)
Bipartite     O(V+E)    O(E√V)
UnitNetwork   O(V+E)    O(E min(V^1.5, √E))
```

9.4.2 二分圖匹配

1. 最大匹配數：給定二分圖 G，在 G 的子圖 M 中，M 的任兩條邊都沒有公共節點，則 M 成為此二分圖的匹配，$|EM|$ 最大的匹配則成為最大匹配。
2. 最小點覆蓋：在 VG 中選取最少的點，形成子集合 V，使 E 為所有與 V 中的點 incident 的邊形成的集合。
3. 最大獨立集：在 VG 中選取最多的點，形成子集合 V，且任兩個 V 中的 vertices 都不相鄰。
4. Konig 定理：對於任意二分圖，滿足以下兩個條件

    (a) 最大匹配數 = 最小點覆蓋的頂點數
    (b) 最大獨立集之頂點數 = 總頂點數 - 最大匹配數

9.4.3 Pick 公式

給定頂點坐標均是整點的簡單多邊形，面積 = 內部格點數 + 邊上格點數/2-1

9.4.4 圖論

1. For planner graph，$F = E - V + C + 1$，C 是連通分量數
2. For planner graph，$E \le 3V - 6$
3. 對於連通圖 G，最大獨立點集的大小設為 I(G)，最大匹配大小設為 M(G)，最小點覆蓋設為 Cv(G)，最小邊覆蓋設為 Ce(G)。對於任意連通圖：

    (a) $I(G) + Cv(G) = |V|$
    (b) $M(G) + Ce(G) = |V|$

4. 對於連通二分圖：

    (a) $I(G) = Cv(G)$
    (b) $M(G) = Ce(G)$

5. 最大權閉合圖：

    (a) $C(u,v) = \infty, (u,v) \in E$
    (b) $C(S,v) = W_v, W_v > 0$
    (c) $C(v,T) = -W_v, W_v < 0$
    (d) ans$=\sum_{W_v>0} W_v - flow(S,T)$

6. 最大密度子圖：

    (a) 求 $max\left(\frac{W_e+W_v}{|V'|}\right), e \in E', v \in V'$
    (b) $U = \sum_{v \in V} 2W_v + \sum_{e \in E} W_e$
    (c) $C(u,v) = W_{(u,v)}, (u,v) \in E$，雙向邊
    (d) $C(S,v) = U, v \in V$
    (e) $D_u = \sum_{(u,v) \in E} W_{(u,v)}$
    (f) $C(v,T) = U + 2g - D_v - 2W_v, v \in V$
    (g) 二分搜 g：
       $l = 0, r = U, eps = 1/n^2$
       if$((U \times |V| - flow(S,T))/2 > 0)$ $l = mid$
       else $r = mid$
    (h) ans$=min\_cut(S,T)$
    (i) $|E| = 0$ 要特殊判斷

7. 弦圖：

    (a) 點數大於 3 的環都要有一條弦

(b) 完美消除序列從後往前依次給每個點染色，給每個點染上可以染的最小顏色
(c) 最大團大小 = 色數
(d) 最大獨立集: 完美消除序列從前往後能選就選
(e) 最小覆蓋: 最大獨立集的點和他延伸的邊構成
(f) 區間圖是弦圖
(g) 區間圖的完美消除序列: 將區間按造又端點由小到大排序
(h) 區間圖染色: 用線段樹做

9.4.5 0-1 分數規劃

$x_i = \{0,1\}$，x_i 可能會有其他限制，求 $max\left(\frac{\sum B_i x_i}{\sum C_i x_i}\right)$

1. $D(i,g) = B_i - g \times C_i$
2. $f(g) = \sum D(i,g) x_i$
3. $f(g) = 0$ 時 g 為最佳解，$f(g) < 0$ 沒有意義
4. 因為 $f(g)$ 單調可以二分搜 g
5. 或用 Dinkelbach 通常比較快

```
1  binary_search(){
2    while(r-l>eps){
3      g=(l+r)/2;
4      for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
5      找出一組合法x[i]使f(g)最大;
6      if(f(g)>0) l=g;
7      else r=g;
8    }
9    Ans = r;
10 }
11 Dinkelbach(){
12   g=任意狀態(通常設為0);
13   do{
14     Ans=g;
15     for(i:所有元素)D[i]=B[i]-g*C[i];//D(i,g)
16     找出一組合法x[i]使f(g)最大;
17     p=0,q=0;
18     for(i:所有元素)
19       if(x[i])p+=B[i],q+=C[i];
20     g=p/q;//更新解，注意q=0的情況
21   }while(abs(Ans-g)>EPS);
22   return Ans;
23 }
```

9.4.6 Math

1. $\sum_{d|n} \phi(n) = n$
2. Harmonic series $H_n = \ln(n) + \gamma + 1/(2n) - 1/(12n^2) + 1/(120n^4)$
3. Gray Code $= n \oplus (n >> 1)$
4. $SG(A + B) = SG(A) \oplus SG(B)$
5. Rotate Matrix $M(\theta) = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$
6. $\sum_{d|n} \mu(n) = [n == 1]$
7. $g(m) = \sum_{d|m} f(d) \Leftrightarrow f(m) = \sum_{d|m} \mu(d) \times g(m/d)$
8. $\sum_{i=1}^{n} \sum_{j=1}^{m}$ 互質數量 $= \sum \mu(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
9. $\sum_{i=1}^{n} \sum_{j=1}^{n} lcm(i,j) = n \sum_{d|n} d \times \phi(d)$
10. Josephus Problem
    $f(1,k) = 0, f(n,k) = (f(n-1,k) + k)\%n$

11. Mobius
$$u(n) = \begin{cases} 1 & , n = 1 \\ 0 & , n有平方數因數 \\ (-1)^k & , n = p_1 p_2 p_3 \ldots p_k \end{cases}$$
$u(ab) = u(a)u(b), \sum_{d|n} u(d) = [n == 1]$

12. Mobius Inversion
$f(m) = \sum_{d|n} g(d) \Leftrightarrow g(n) = \sum_{d|n} u(d) \times f(n/d) = \sum_{d|n} u(n/d) \times f(d)$

13. 排組公式

    (a) n-Catalan $C_0 = 1$, $C_{n+1} = \frac{2(2n+1)C_n}{n+2}$
    (b) kn-Catalan $\frac{C_n^{kn}}{n(k-1)+1}$, $C_m^n = \frac{n!}{m!(n-m)!}$
    (c) Stirling number of 2nd, n 人分 k 組方法數目
        i. $S(0,0) = S(n,n) = 1$
        ii. $S(n,0) = 0$
        iii. $S(n,k) = kS(n-1,k) + S(n-1,k-1)$
    (d) Bell number, n 人分任意多組方法數目
        i. $B_0 = 1$
        ii. $B_n = \sum_{i=0}^{n} S(n,i)$
        iii. $B_{n+1} = \sum_{k=0}^{n} C_k^n B_k$
        iv. $B_{p+n} \equiv B_n + B_{n+1} mod p$, p is prime
        v. $B_{p^m+n} \equiv mB_n + B_{n+1} mod p$, p is prime
        vi. From B_0 : 1, 1, 2, 5, 15, 52,
            203, 877, 4140, 21147, 115975
    (e) Derangement, 錯排, 沒有人在自己位置上
        i. $D_n = n!(1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} \ldots + (-1)^n \frac{1}{n!})$
        ii. $D_n = (n-1)(D_{n-1} + D_{n-2}), D_0 = 1, D_1 = 0$
        iii. From D_0 : 1, 0, 1, 2, 9, 44,
            265, 1854, 14833, 133496
    (f) Binomial Equality
        i. $\sum_k \binom{r}{m+k} \binom{s}{n-k} = \binom{r+s}{m+n}$
        ii. $\sum_k \binom{r}{m+k} \binom{s}{n+k} = \binom{r+s}{l-m+n}$
        iii. $\sum_k \binom{r}{m+k} \binom{s+k}{n} (-1)^k = (-1)^{l+m} \binom{s-m}{n-l}$
        iv. $\sum_{k \le l} \binom{l-k}{m} \binom{s}{k-n} (-1)^k = (-1)^{l+m} \binom{s-m-1}{l-n-m}$
        v. $\sum_{0 \le k \le l} \binom{l-k}{m} \binom{q+k}{n} = \binom{l+q+1}{m+n+1}$
        vi. $\binom{r}{k} = (-1)^k \binom{k-r-1}{k}$
        vii. $\binom{r}{m} \binom{m}{k} = \binom{r}{k} \binom{r-k}{m-k}$
        viii. $\sum_{k \le n} \binom{r+k}{k} = \binom{r+n+1}{n}$
        ix. $\sum_{0 \le k \le n} \binom{k}{m} = \binom{n+1}{m+1}$
        x. $\sum_{k \le m} \binom{m+r}{k} x^k y^{m-k} = \sum_{k \le m} \binom{-r}{k} (-x)^k (x+y)^{m-k}$

14. LinearAlgebra

    (a) $tr(A) = \sum_i A_{i,i}$
    (b) eigen vector: $(A - cI)x = 0$

15. 冪次, 冪次和

    (a) $a^b \% P = a^{b\%\varphi(p)+\varphi(p)}, b \ge \varphi(p)$
    (b) $1^3 + 2^3 + 3^3 + \ldots + n^3 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
    (c) $1^4 + 2^4 + 3^4 + \ldots + n^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$
    (d) $1^5 + 2^5 + 3^5 + \ldots + n^5 = \frac{n^6}{6} + \frac{n^5}{2} + \frac{5n^4}{12} - \frac{n^2}{12}$
    (e) $0^k + 1^k + 2^k + \ldots + n^k = P(k), P(k) = \frac{(n+1)^{k+1} - \sum_{i=0}^{k-1} C_i^{k+1} P(i)}{k+1}, P(0) = n + 1$
    (f) $\sum_{k=0}^{m-1} k^n = \frac{1}{n+1} \sum_{k=0}^{n} C_k^{n+1} B_k m^{n+1-k}$
    (g) $\sum_{j=0}^{m} C_j^{m+1} B_j = 0, B_0 = 1$
    (h) 除了 $B_1 = -1/2$，剩下的奇數項都是 0
    (i) $B_2 = 1/6, B_4 = -1/30, B_6 = 1/42, B_8 = -1/30, B_{10} = 5/66, B_{12} = -691/2730, B_{14} = 7/6, B_{16} = -3617/510, B_{18} = 43867/798, B_{20} = -174611/330,$

16. Chinese Remainder Theorem

(a) $gcd(m_i, m_j) = 1$

(b) $x\%m_1 = a_1$

$x\%m_2 = a_2$

$\vdots$

$x\%m_n = a_n$

(c) $M = m_1 m_2 \ldots m_n, M_i = M/m_i$

(d) $t_i m_i = 1 (\mod m_i)$

(e) $x = a_1 t_1 * M_1 + \cdots + a_n t_n * M_n + kM, k \in N$

9.4.7 Burnside's lemma

1. $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$

2. $X^g = t^{c(g)}$

3. G 表示有幾種轉法，X^g 表示在那種轉法下，有幾種是會保持對稱的，t 是顏色數，$c(g)$ 是循環節不動的面數。

4. 正立方體塗三顏色，轉 0 有 3^6 個元素不變，轉 90 有 6 種，每種有 3^3 不變，180 有 3×3^4，120(角) 有 8×3^2，180(邊) 有 6×3^3，全部 $\frac{1}{24}\left(3^6 + 6 \times 3^3 + 3 \times 3^4 + 8 \times 3^2 + 6 \times 3^3\right) = 57$

9.4.8 Probability

1. $e^x(1 - x^2) \leq 1 + x \leq e^x$

2. $n! \leq e n^{\frac{1}{2}} (\frac{n}{e})^n$

3. $Pr[X \geq a] \leq \frac{E[X]}{a}, X \leq 0, a > 0$

4. $\text{Cov}[X, Y] = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$

5. $\text{Var}[\sum_j X_j] = (\sum_j \text{Var}[X_j]) + 2\sum_{i<j} \text{Cov}[X_i, X_j]$

6. $Pr[X \leq a] \leq \min_{t<0} \frac{E[e^{tX}]}{e^{ta}}$

7. $M_X(t) = E[e^{tX}]$

8. $Pr[X \geq a] \leq \min_{t>0} \frac{E[e^{tX}]}{e^{ta}}$

9. $Pr[X \leq a] \leq \min_{t<0} \frac{E[e^{tX}]}{e^{ta}}$

10. $\forall \delta > 0, Pr[X \geq (1+\delta)\mu] \leq (\frac{e^\delta}{(1+\delta)^{(1+\delta)}})^\mu$

11. $\forall 0 < \delta \leq 1, Pr[X \geq (1+\delta)\mu] \leq e^{\frac{-\mu\delta^2}{3}}$

12. $R \geq 6\mu, Pr[X \geq R] \leq 2^{-R}$

13. $0 < \delta < 1, Pr[X \leq (1-\delta)\mu] \leq (\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}})^\mu$

14. $0 < \delta \leq 1, Pr[X \leq (1-\delta)\mu] \leq e^{\frac{-\mu\delta^2}{2}}$

9.4.9 Tree Counting

1. Rooted tree: $s_{n+1} = \frac{1}{n} \sum_{i=1}^{n} (i \times a_i \times \sum_{j=1}^{\lfloor n/i \rfloor} a_{n+1-i \times j})$

2. Unrooted tree:

   (a) Odd:$a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$

   (b) Even:$Odd + \frac{1}{2} a_{n/2}(a_{n/2} + 1)$

3. Spanning Tree

   (a) Cayley: n^{n-2} (Complete Graph)

   (b) Kirchhoff: $M[i][i] = \deg(V_i), M[i][j] = E(i,j)? -1 : 0$. delete any one row and col in A, $ans = det(A)$

9.5 莫隊算法 _ 區間眾數

```cpp
using namespace std;
const int maxn = 1e6 + 10;
struct query { int id, bk, l, r; };
int arr[maxn], cnt[maxn], d[maxn], n, m, bk, mx;
pair<int,int> ans[maxn];
vector<query> q;
bool cmp(query x,query y) {
    return (x.bk < y.bk || (x.bk == y.bk) && x.r < y.r);
}
void add(int pos) {
    d[cnt[arr[pos]]]--;
    cnt[arr[pos]]++;
    d[cnt[arr[pos]]]++;
    if(d[mx + 1] > 0) mx++;
}
void del(int pos) {
    d[cnt[arr[pos]]]--;
    cnt[arr[pos]]--;
    d[cnt[arr[pos]]]++;
    if(d[mx] == 0) mx--;
}
void mo(int n, int m) {
    sort(q.begin(), q.end(), cmp);
    for(int i = 0, cl = 1, cr = 0; i < m; i++) {
        while(cr < q[i].r) add(++cr);
        while(cl > q[i].l) add(--cl);
        while(cr > q[i].r) del(cr--);
        while(cl < q[i].l) del(cl++);
        ans[q[i].id] = make_pair(mx, d[mx]);
    }
}
int main(){
    cin >> n >> m;
    bk = (int)sqrt(n + 0.5);
    for(int i = 1; i <= n; i++) cin >> arr[i];
    q.resize(m);
    for(int i = 0; i < m; i++) {
        cin >> q[i].l >> q[i].r;
        q[i].id = i,q[i].bk = (q[i].l - 1) / bk;
    }
    mo(n, m);
    for(int i = 0; i < m; i++)
        cout << ans[i].first << ' ' << ans[i].second << '\n';
    return 0;
}
```

NCTU-Pusheen Codebook

Contents