

1 Data_Structure

1.1 Sparse Table

```

1 struct SparseTable {
2     const int N;
3     vector<vector<ll>> bitarray;
4     // Constructs a sparse table with size same as vec.
5     SparseTable(const vector<ll>& vec) : N(vec.size()) {
6         bitarray.assign(__lg(N) + 1, vector<ll>(N));
7         bitarray[0] = vec;
8         for (int i = 1; (1 << i) <= N; i++) {
9             int k = N - (1 << i);
10            for (int j = 0; j < k; j++) {
11                // Comparator may be needed.
12                bitarray[i][j] = max(bitarray[i - 1][j],
13                                     bitarray[i - 1][j + (1 << (i - 1))]);
14            }
15        }
16        // Queries the maximum element in [left_inc, right_inc]
17        // where left_inc and right_inc in [0, n).
18        ll query(int left_inc, int right_inc) {
19            int k = __lg(right_inc - left_inc + 1);
20            return max(bitarray[k][left_inc], bitarray[k][
21                right_inc - (1 << k) + 1]);
22        }
23    };

```

1.2 Binary Indexed Tree

```

1 struct BIT {
2     int N;
3     vector<ll> bitarray;
4     // Constructs a Binary Indexed Tree where n is size of
5     // array.
6     BIT(int n) : N(n) { bitarray.assign(N + 1, 0); }
7     ll query(int index) {
8         index++;
9         ll ret = 0;
10        while (index > 0) ret += bitarray[index], index -=
11            index & -index;
12        return ret;
13    }
14    // Increases element at index by val, where index in [1,
15    // n].
16    void add(int index, ll val) {
17        index++;
18        while (index <= N) bitarray[index] += val, index +=
19            index & -index;
20    }
21    // Queries sum in [left_inc, right_inc], where left_inc
22    // and right_inc in [1, n].
23    ll query(int left_inc, int right_inc) { return query(
24        right_inc) - query(left_inc - 1); }
25    };

```

1.3 Fenwick Tree

```

1 struct FenwickTree {
2     const int N;
3     vector<ll> d, dd;
4     // Constructs a Fenwick Tree where n is size of array.
5     FenwickTree(int n) : N(n) {
6         d.assign(N + 2, 0);
7         dd.assign(N + 2, 0);
8     }
9     ll query(int index) {
10        index++;
11        ll s = 0, ss = 0;
12        ll c = index + 1;
13        while (index > 0) {
14            s += d[index], ss += dd[index];
15            index -= index & -index;
16        }
17        return c * s - ss;
18    }
19    void add(int index, ll val) {
20        index++;
21        int c = index;
22        while (index <= N) {
23            d[index] += val, dd[index] += c * val;
24            index += index & -index;
25        }
26    }
27    // Queries sum in [left_inc, right_inc], where left_inc
28    // and right_inc in [1, n].
29    ll query(int left_inc, int right_inc) { return query(
30        right_inc) - query(left_inc - 1); }
31    // Increases all elements in [left_inc, right_inc] by val
32    // , where left_inc and right_inc in [1, n].
33    void add(int left_inc, int right_inc, ll val) { add(
34        left_inc, val), add(right_inc + 1, -val); }
35    };

```

1.4 線段樹

```

1 #define INF 2147483647
2 struct Node {
3     int add_tag, chg_tag, Min, Max, sum;
4     Node():add_tag(0), chg_tag(0) {}
5 };
6 struct ST {
7     int sz;
8     Node nd[maxn*4];
9     ST(){}
10    ST(int a[], int n):sz(n) {
11        build(a, 0, n-1, 1);
12    }
13    void build(int a[], int l, int r, int i) {
14        if(l==r) {
15            nd[i].Min = nd[i].Max = nd[i].sum = a[l];
16            return;
17        }
18        int mid = (l+r)>>1;
19        int lid = i<<1, rid = (i<<1)|1;
20        build(a, l, mid, lid), build(a, mid+1, r, rid);
21        nd[i].Min = min(nd[lid].Min, nd[rid].Min);
22        nd[i].Max = max(nd[lid].Max, nd[rid].Max);

```

```

23        nd[i].sum = nd[lid].sum + nd[rid].sum;
24    }
25    void push(int l, int r, int i, int lid, int rid) {
26        if(nd[i].chg_tag) {
27            nd[i].sum = nd[i].chg_tag * (r-l+1);
28            nd[i].Max = nd[i].Min = nd[i].chg_tag;
29            if(lid) nd[lid].chg_tag = nd[i].chg_tag, nd[lid].
30                add_tag = 0;
31            if(rid) nd[rid].chg_tag = nd[i].chg_tag, nd[rid].
32                add_tag = 0;
33            nd[i].chg_tag = 0;
34        }
35        if(nd[i].add_tag) {
36            nd[i].sum += nd[i].add_tag * (r-l+1);
37            nd[i].Max += nd[i].add_tag, nd[i].Min += nd[i].
38                add_tag;
39            if(lid) nd[lid].add_tag += nd[i].add_tag;
40            if(rid) nd[rid].add_tag += nd[i].add_tag;
41            nd[i].add_tag = 0;
42        }
43    }
44    void add(int ql, int qr, int l, int r, int i, int val) {
45        if(!val) return;
46        int lid = i<<1, rid = (i<<1)|1;
47        if(l==r) lid = rid = 0;
48        push(l, r, i, lid, rid);
49        if(r<ql || qr<l) return;
50        if(ql<=l && r<=qr) {
51            nd[i].add_tag += val;
52        }
53        else {
54            int mid = (l+r)>>1;
55            add(ql, qr, l, mid, lid, val), add(ql, qr, mid+1, r, rid,
56                val);
57            nd[i].Max = max(nd[lid].Max + nd[lid].add_tag, nd
58                [rid].Max + nd[rid].add_tag);
59            nd[i].Min = min(nd[lid].Min + nd[lid].add_tag, nd
60                [rid].Min + nd[rid].add_tag);
61            nd[i].sum = nd[lid].sum + nd[lid].add_tag*(mid-l
62                +1) + nd[rid].sum + nd[rid].add_tag*(r-mid);
63        }
64    }
65    void chg(int ql, int qr, int l, int r, int i, int val) {
66        int lid = i<<1, rid = (i<<1)|1;
67        if(l==r) lid = rid = 0;
68        push(l, r, i, lid, rid);
69        if(r<ql || qr<l) return;
70        if(ql<=l && r<=qr) {
71            nd[i].chg_tag = val;
72        }
73        else {
74            int mid = (l+r)>>1;
75            chg(ql, qr, l, mid, lid, val), chg(ql, qr, mid+1, r, rid,
76                val);
77            nd[i].Max = max(nd[lid].chg_tag?nd[lid].chg_tag:
78                nd[lid].Max, nd[rid].chg_tag?nd[rid].chg_tag
79                :nd[rid].Max);
80            nd[i].Min = min(nd[lid].chg_tag?nd[lid].chg_tag:
81                nd[lid].Min, nd[rid].chg_tag?nd[rid].chg_tag
82                :nd[rid].Min);
83            nd[i].sum = (nd[lid].chg_tag?nd[lid].chg_tag*(mid
84                -l+1):nd[lid].sum) + (nd[rid].chg_tag?nd[rid]
85                .chg_tag*(r-mid):nd[rid].sum);
86        }
87    }
88    int query_min(int ql, int qr, int l, int r, int i) {

```

```

75     int lid = i<<1, rid = (i<<1)|1;
76     if(l==r) lid = rid = 0;
77     push(l,r,i,lid,rid);
78     if(r<ql || qr<l) return INF;
79     if(ql<=l && r<=qr) return nd[i].Min;
80     int mid = (l+r)>>1;
81     return min(query_min(ql,qr,l,mid,lid), query_min(ql,
82         qr,mid+1,r,rid));
83 }
84 int query_max(int ql,int qr,int l,int r,int i) {
85     int lid = i<<1, rid = (i<<1)|1;
86     if(l==r) lid = rid = 0;
87     push(l,r,i,lid,rid);
88     if(r<ql || qr<l) return -INF;
89     if(ql<=l && r<=qr) return nd[i].Max;
90     int mid = (l+r)>>1;
91     return max(query_max(ql,qr,l,mid,lid), query_max(ql,
92         qr,mid+1,r,rid));
93 }
94 int query_sum(int ql,int qr,int l,int r,int i) {
95     int lid = i<<1, rid = (i<<1)|1;
96     if(l==r) lid = rid = 0;
97     push(l,r,i,lid,rid);
98     if(r<ql || qr<l) return 0;
99     if(ql<=l && r<=qr) return nd[i].sum;
100     int mid = (l+r)>>1;
101     return query_sum(ql,qr,l,mid,lid) + query_sum(ql,qr,
    mid+1,r,rid);
}
};

```

1.5 持久化線段樹

```

1 //POJ 2104 //k-th number
2 #include<bits/stdc++.h>
3 #define maxn 100005
4 using namespace std;
5 int a[maxn],b[maxn],root[maxn];
6 int cnt;
7 struct node {
8     int sum, L_son, R_son;
9 } tree[maxn<<5];
10 int create(int _sum, int _L_son, int _R_son) {
11     int idx = ++cnt;
12     tree[idx].sum = _sum, tree[idx].L_son = _L_son, tree[idx]
13     ].R_son = _R_son;
14     return idx;
15 }
16 void Insert(int &root,int pre_rt,int pos,int L,int R) {
17     root = create(tree[pre_rt].sum+1,tree[pre_rt].L_son,tree[
18     pre_rt].R_son);
19     if(L==R) return;
20     int M = (L+R)>>1;
21     if(pos<=M) Insert(tree[root].L_son,tree[pre_rt].L_son,pos
22     ,L,M);
23     else Insert(tree[root].R_son,tree[pre_rt].R_son,pos,M+1,R
24     );
25 }
26 int query(int L_id,int R_id,int L,int R,int K) {
27     if(L==R) return L;
28     int M = (L+R)>>1;
29     int s = tree[tree[R_id].L_son].sum - tree[tree[L_id].
30     L_son].sum;
31     if(K<=s) return query(tree[L_id].L_son,tree[R_id].L_son,L
32     ,M,K);
33     return query(tree[L_id].R_son,tree[R_id].R_son,M+1,R,K-s)
34     ;
35 }
36 int main() {
37     int n,m; scanf("%d%d",&n,&m);
38     for(int i=1;i<=n;i++) {
39         scanf("%d",&a[i]);
40         b[i] = a[i];
41     }
42     sort(b+1,b+1+n); //離散化
43     int b_sz = unique(b+1,b+1+n)-(b+1);
44     cnt = root[0] = 0;
45     for(int i=1;i<=n;i++) {
46         int pos = lower_bound(b+1,b+1+b_sz,a[i])-b;
47         Insert(root[i],root[i-1],pos,1,b_sz);
48     }
49     while(m--) {
50         int l,r,k; scanf("%d%d%d",&l,&r,&k);
51         int pos = query(root[l-1],root[r],1,b_sz,k);
52         printf("%d\n",b[pos]);
53     } return 0;
54 }

```

1.6 Treap

```

1 //POJ 3580 區間反轉 + 加值 詢問min
2 #include<bits/stdc++.h>
3 using namespace std;
4 #define maxn 100005
5 #define INF 2147483647
6 struct Treap
7 {
8     Treap *L, *R;
9     int min_val, size, val;
10    bool rev_tag;
11    int add_tag;
12    Treap(int _val):L(NULL),R(NULL),min_val(_val),size(1),
13    rev_tag(false),add_tag(0),val(_val){}
14    void pull()
15    {
16        if(L) L->push();
17        if(R) R->push();
18        size = (L ? L->size : 0) + (R ? R->size : 0) + 1;
19        min_val = min( min( val, L ? L->min_val : INF), R ? R
20        ->min_val : INF);
21    }
22    void push()
23    {
24        val += add_tag;
25        min_val += add_tag;
26        if(L) L->add_tag += add_tag;
27        if(R) R->add_tag += add_tag;
28        add_tag = 0;
29        if(rev_tag)
30        {
31            swap(L,R);
32            if(L) L->rev_tag ^= 1;
33            if(R) R->rev_tag ^= 1;
34            rev_tag = false;
35        }
36    }
37 }

```

```

38 }
39 inline int Random()
40 {
41     static int x = 7122;
42     return (x = x * 0xdefaced + 1) & INF;
43 }
44 void split(Treap *p, Treap *a, Treap *b, int k)
45 {
46     int sz = p ? p->size : 0;
47     if(!p) a = b = NULL;
48     else if(k<=0) a = NULL, b = p;
49     else
50     {
51         p->push();
52         if(p->L && p->L->size >=k)
53         {
54             b = p;
55             split(p->L, a, b->L, k);
56         }
57         else
58         {
59             a = p;
60             split(p->R, a->R, b, k - (p->L ? p->L->size : 0)
61             -1 );
62         }
63         p->pull();
64     }
65 }
66 Treap* merge(Treap *a, Treap *b)
67 {
68     if(!a) return b;
69     if(!b) return a;
70     if(Random()%<a->size+b->size> < a->size)
71     {
72         a->push();
73         a->R = merge(a->R,b);
74         a->pull();
75         return a;
76     }
77     else
78     {
79         b->push();
80         b->L = merge(a,b->L);
81         b->pull();
82         return b;
83     }
84 }
85 void insert(Treap *&p, int x, int pos)
86 {
87     Treap *a, *b;
88     Treap *t = new Treap(x);
89     split(p,a,b,pos);
90     p = merge(merge(a,t),b);
91 }
92 void del(Treap *&p, int x)
93 {
94     Treap *a, *b, *c, *d;
95     split(p,a,d,x-1);
96     split(d,b,c,1);
97     p = merge(a,c);
98 }
99 void reverse(Treap *&p, int x,int y)
100 {
101     Treap *a, *b,*c;
102     split(p,a,c,y);
103     split(a,b,x-1);
104     reverse(b);
105     p = merge(a,b);
106 }

```

```

99     b->rev_tag ^= 1;
100     p = merge(merge(a,b),c);
101 }
102 void add(Treap *p, int x,int y,int v)
103 {
104     Treap *a, *b,*c;
105     split(p,a,c,y);
106     split(a,a,b,x-1);
107     b->add_tag += v;
108     p = merge(merge(a,b),c);
109 }
110 int Min(Treap *p, int x, int y)
111 {
112     Treap *a, *b,*c;
113     split(p,a,c,y);
114     split(a,a,b,x-1);
115     int ans = b->min_val + b->add_tag;
116     p = merge(merge(a,b),c);
117     return ans;
118 }
119 void revolve(Treap *p, int x, int y, int t)
120 {
121     Treap *a, *b,*c,*d,*e;
122     split(p,a,c,y);
123     split(a,a,b,x-1);
124     split(b,d,e,(y-x+1)-t%(y-x+1));
125     p = merge(merge(a,merge(e,d)),c);
126 }
127
128 int main()
129 {
130     int n,m;
131     scanf("%d",&n);
132     int a[maxn];
133     for(int i=1; i<=n; i++)
134         scanf("%d",&a[i]);
135     Treap* root = new Treap(a[1]);
136     for(int i=2; i<=n; i++)
137         insert(root,a[i],i);
138     scanf("%d",&m);
139     while(m--)
140     {
141         char s[10];
142         scanf("%s",s);
143         if(strcmp(s,"ADD")==0)
144         {
145             int x, y, d;
146             scanf("%d%d%d",&x,&y,&d);
147             add(root,x,y,d);
148         }
149         else if(strcmp(s,"REVERSE")==0)
150         {
151             int x, y;
152             scanf("%d%d",&x,&y);
153             reverse(root,x,y);
154         }
155         else if(strcmp(s,"REVOLVE")==0)
156         {
157             int x, y, t;
158             scanf("%d%d%d",&x,&y,&t);
159             revolve(root,x,y,t);
160         }
161         else if(strcmp(s,"INSERT")==0)
162         {
163             int x, p;
164             scanf("%d%d",&x,&p);

```

```

165         insert(root,p,x);
166     }
167     else if(strcmp(s,"DELETE")==0)
168     {
169         int x;
170         scanf("%d",&x);
171         del(root,x);
172     }
173     else if(strcmp(s,"MIN")==0)
174     {
175         int x, y;
176         scanf("%d%d",&x,&y);
177         printf("%d\n",Min(root,x,y));
178     }
179 }
180 return 0;
181 }

```

1.7 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 };

```

1.8 Heavy Light

```

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 }

```

1.9 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，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是每個邊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 DP

2.1 Edit Distance

```

1 int edit_distance(string s, string t) {
2     int n = s.size(), m = t.size();
3     memset(dp, 0, sizeof(dp));
4     for (int i = 1; i <= n; i++) dp[i][0] = i;
5     for (int i = 1; i <= m; i++) dp[0][i] = i;
6     for (int i = 1; i <= n; i++)
7         for (int j = 1; j <= m; j++)
8             dp[i][j] = min(dp[i-1][j-1] + !(s[i-1] == t[j-1]), min(dp[i][j-1], dp[i-1][j]) + 1);
9     return dp[n][m];
10 }

```

2.2 LCIS

```

1 // Longest Common Increasing Subsequence
2 #include<bits/stdc++.h>
3 using namespace std;
4 #define LEN 505
5 int main()
6 {
7     int n,m;
8     scanf("%d%d",&n,&m);
9     int a[LEN],b[LEN];

```

```

10     for(int i=1; i<=n; i++)
11         scanf("%d", &a[i]);
12     for(int i=1; i<=m; i++)
13         scanf("%d", &b[i]);
14     int dp[LEN][LEN] = {}; // dp[i][j]:以b[j]結尾的LCIS長度
15     int pre[LEN][LEN] = {}; // 用來回溯
16     for(int i=1; i<=n; i++)
17     {
18         int p = 0;
19         for(int j=1; j<=m; j++)
20         {
21             if(a[i]!=b[j])
22             {
23                 dp[i][j] = dp[i-1][j];
24                 pre[i][j] = j;
25                 if( a[i]>b[j] && dp[i-1][j]>dp[i-1][p] )
26                     p = j;
27             }
28             else
29             {
30                 dp[i][j] = dp[i-1][p]+1;
31                 pre[i][j] = p;
32             }
33         }
34     }
35     int len = 0, p = 0;
36     for(int j=1; j<=m; j++)
37     {
38         if(dp[n][j]>len)
39         {
40             len = dp[n][j];
41             p = j;
42         }
43     }
44     printf("LEN = %d\n", len);
45     vector<int> ans;
46     for(int i=n; i>=1; i--)
47     {
48         if(a[i]==b[p])
49             ans.push_back(b[p]);
50         p = pre[i][p];
51     }
52     while(ans.size())
53     {
54         printf("%d ",ans.back());
55         ans.pop_back();
56     }
57     return 0;
58 }

```

2.3 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         {}
8         bool operator<(const BB &x) const {
9             return w * c < x.w * x.c;
10         }

```

```

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
20                 && k <= c; k++, tw += w, tv += v) {
21                 int dpv = dp[tw] - tv;
22                 while (l <= r && MQ[r][1] <= dpv) r--;
23                 r++;
24                 MQ[r][0] = k, MQ[r][1] = dpv;
25                 dp[tw] = max(dp[tw], MQ[l][1] + tv);
26             }
27             for (int k = c+1, tw = (c+1)*w+j, tv = (c+1)*
28                 v; tw <= sum; k++, tw += w, tv += v) {
29                 if (k - MQ[l][0] > c) l++;
30                 int dpv = dp[tw] - tv;
31                 while (l <= r && MQ[r][1] <= dpv) r--;
32                 r++;
33                 MQ[r][0] = k, MQ[r][1] = dpv;
34                 dp[tw] = max(dp[tw], MQ[l][1] + tv);
35             }
36         }
37     }
38     static int knapsack(int C[][3], int N, int W) { // O(WN)
39         vector<BB> A;
40         for (int i = 0; i < N; i++) {
41             int w = C[i][0], v = C[i][1], c = C[i][2];
42             A.push_back(BB(w, v, c));
43         }
44         assert(N < MAXN);
45         static int dp1[MAXW+1], dp2[MAXW+1];
46         BB Ar[2][MAXN];
47         int ArN[2] = {};
48         memset(dp1, 0, sizeof(dp1[0])*(W+1));
49         memset(dp2, 0, sizeof(dp2[0])*(W+1));
50         sort(A.begin(), A.end());
51         int sum[2] = {};
52         for (int i = 0; i < N; i++) {
53             int ch = sum[1] < sum[0];
54             Ar[ch][ArN[ch]] = A[i];
55             ArN[ch]++;
56             sum[ch] = min(sum[ch] + A[i].w*A[i].c, W);
57         }
58         run(Ar[0], dp1, W, ArN[0]);
59         run(Ar[1], dp2, W, ArN[1]);
60         int ret = 0;
61         for (int i = 0, j = W, mx = 0; i <= W; i++, j--) {
62             mx = max(mx, dp2[i]);
63             ret = max(ret, dp1[j] + mx);
64         }
65         return ret;
66     }
67 }
68 int main() {
69     int W, N;
70     assert(scanf("%d %d", &W, &N) == 2);
71     int C[MAXN][3];
72     for (int i = 0; i < N; i++)
73         assert(scanf("%d %d %d", &C[i][1], &C[i][0], &C[i]
74             [2]) == 3);

```

```

73     printf("%d\n", knapsack(C, N, W));
74     return 0;
75 }

```

2.4 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) <
23         f(stk[top].L, stk[top].pos) ) {
24         stk[top-1].R = stk[top].R;
25         top--;
26     }
27     int lo = stk[top].L, hi = stk[top].R, mid, pos = stk[top]
28         ].pos;
29     // if ( i >= lo ) lo = i + 1;
30     while (lo != hi) {
31         mid = lo + (hi - lo) / 2;
32         if ( f(mid, i) < f(mid, pos) ) hi = mid;
33         else lo = mid + 1;
34     }
35     if (hi < stk[top].R) {
36         stk[top+1] = (INV) { hi, stk[top].R, i };
37         stk[top++].R = hi;
38     }
39 }
40 int main() {
41     cin >> t;
42     while (t-- ) {
43         cin >> n >> L >> p;
44         dp[0] = sum[0] = 0;
45         for (int i = 1; i <= n; i++) {
46             cin >> s[i];
47             sum[i] = sum[i-1] + strlen(s[i]);
48             dp[i] = numeric_limits<long double>::max();
49         }
50         stk[top] = (INV) { 1, n + 1, 0 };
51         for (int i = 1; i <= n; i++) {
52             if (i >= stk[bot].R) bot++;
53             dp[i] = f(i, stk[bot].pos);
54             update(i);
55             // cout << (LL) f(i, stk[bot].pos) << endl;
56         }
57         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 }

```

3 Graph

3.1 Dijkstra

```

1 // Queries minimum path from src to dest in a graph where src
2 // and dest are connected.
3 // The n vertices are supposed to be marked [0, n); edge
4 // should have size n.
5 int dijkstra(int src, int dest, const vector<vector<pii>>&
6     edge) {
7     const int N = edge.size();
8     bool nvis[N] = {0};
9     // A comparator may be required.
10    priority_queue<pii, vector<pii>, greater<pii>> q;
11    q.emplace(0, src);
12    while (!q.empty()) {
13        int v = q.top().second;
14        int d = q.top().first;
15        q.pop();
16        if (v == dest) return d;
17        if (nvis[v]) continue;
18        nvis[v] = true;
19        for (auto& e : edge[v]) {
20            if (!nvis[e.second]) {
21                // Fit the comparator
22                q.emplace(d + e.first, e.second);
23            }
24        }
25    }
26    throw "src and dest are not connected.";
27 }
28 // Queries minimum path from src to all the other vertices
29 // in a graph where all vertices are connected.
30 // The n vertices are supposed to be marked [0, n); edge
31 // should have size n.
32 vector<int> dijkstra(int src, const vector<vector<pii>>& edge
33 ) {
34     const int N = edge.size();
35     vector<int> mindist(N, -1);
36     int nvis = 0;
37     // A comparator may be required.
38     priority_queue<pii, vector<pii>, greater<pii>> q;
39     q.emplace(0, src);
40     while (nvis < N) {
41         if (q.empty()) throw "Not all vertices connected.";
42         int v = q.top().second;
43         int d = q.top().first;
44         q.pop();
45         if (mindist[v] != -1) continue;
46         mindist[v] = d;
47         nvis++;
48         for (auto& e : edge[v]) {

```



```

44     if (mindist[e.second] == -1) {
45         // Fit the comparator
46         q.emplace(d + e.first, e.second);
47     }
48 }
49 }
50 return mindist;
51 }

```

3.2 Bellman Ford

```

1 #include<vector>
2 #include<iostream>
3 using namespace std;
4 #define maxn 2000
5 #define INF 100000000
6 typedef pair<int,int> pii;
7 vector<pii> G[maxn];
8 int dis[maxn];
9 bool BellmanFord(int n,int s)
10 {
11     for(int i=1; i<=n; i++)
12         dis[i] = INF;
13     dis[s] = 0;
14     bool relax;
15     for(int r=1; r<=n; r++) //O(VE)
16     {
17         relax = false;
18         for(int i=1; i<=n; i++)
19             for(pii e:G[i])
20                 if( dis[i] + e.second < dis[e.first] )
21                     dis[e.first] = dis[i] + e.second, relax = true;
22     }
23     return relax; //有負環
24 }
25 int main()
26 {
27     int n,m; cin >> n >> m;
28     for(int i=0; i<m; i++)
29     {
30         int a,b,w;
31         cin >> a >> b >> w;
32         G[a].push_back(pii(b,w));
33     }
34     int s,t;
35     cin >> s >> t;
36     if(BellmanFord(n,s)) cout << "There is a minus-cycle.\n";
37     else cout << "dis = " << dis[t] << endl;
38     return 0;
39 }

```

3.3 Floyd Warshall

```

1 #include<iostream>
2 #include<vector>
3 using namespace std;
4 #define maxn 505
5 #define INF 10000000

```

```

6 typedef pair<int,int> pii;
7 vector<pii> G[maxn];
8 int dist[maxn][maxn];
9 void FloydWarshall11(int n)
10 {
11     for(int i=1; i<=n; i++)
12         for(int j=1; j<=n; j++)
13             dist[i][j] = i==j?0:INF;
14     for(int i=1; i<=n; i++)
15         for(pii j:G[i])
16             dist[i][j.first] = j.second;
17     for(int k=1; k<=n; k++) //DP //O(n^3)
18         for(int i=1; i<=n; i++)
19             for(int j=1; j<=n; j++)
20                 dist[i][j] = min(dist[i][j],dist[i][k]+dist[k][j]);
21 }
22 int main()
23 {
24     int n,m,q;
25     cin >> n >> m >> q;
26     for(int i=0; i<m; i++)
27     {
28         int u,v,d;
29         cin >> u >> v >> d;
30         G[u].push_back(pii(v,d));
31     }
32     FloydWarshall11(n);
33     while(q--)
34     {
35         int u,v;
36         cin >> u >> v;
37         if(dist[u][v]==INF)printf("(%d,%d) not connected\n",u,v);
38         printf("Dist(%d,%d)=%d\n",u,v,dist[u][v]);
39     }
40     return 0;
41 }

```

3.4 SPFA

```

1 #include<vector>
2 #include<queue>
3 #include<iostream>
4 using namespace std;
5 #define maxn 2000
6 #define INF 100000000
7 typedef pair<int,int> pii;
8 vector<pii> G[maxn];
9 int dis[maxn];
10 void SPFA(int n,int s) //O(kE) k~2.
11 {
12     for(int i=1; i<=n; i++)
13         dis[i] = INF;
14     dis[s] = 0;
15     queue<int> q;
16     q.push(s);
17     bool inque[maxn] = {};
18     while(!q.empty())
19     {
20         int u = q.front(); q.pop();
21         inque[u] = false;
22         for(pii e:G[u])

```

```

23     {
24         int v = e.first, w = e.second;
25         if( dis[u] + w < dis[v])
26         {
27             if(!inque[v]) q.push(v), inque[v] = true;
28             dis[v] = dis[u] + w;
29         }
30     }
31 }
32 }
33 int main()
34 {
35     int n,m; cin >> n >> m;
36     for(int i=0; i<m; i++)
37     {
38         int a,b,w;
39         cin >> a >> b >> w;
40         G[a].push_back(pii(b,w));
41     }
42     int s,t;
43     cin >> s >> t;
44     SPFA(n,s);
45     if(dis[t]==INF) cout << "dis = INF\n";
46     else cout << "dis = " << dis[t] << endl;
47     return 0;
48 }

```

3.5 Kruskal

```

1 #include<iostream>
2 #include<algorithm>
3 using namespace std;
4 typedef pair<int,int> pii;
5 typedef pair<int,pii> piii;
6 #define w first
7 #define x second.first
8 #define y second.second
9 #define maxm 100000
10 #define maxn 10000
11 struct UFT //Union-Find Tree
12 {
13     int sz;
14     int p[maxn];
15     UFT(int _sz) {
16         sz = _sz;
17         for(int i=1; i<=sz; i++)
18             p[i] = i;
19     }
20     inline int par(int a) { //parent
21         return p[a] = (a==p[a] ? a : par(p[a]));
22     }
23     inline bool same(int a, int b) { //check if a and b are
24         //in the same set
25         return par(a) == par(b);
26     }
27     inline void uni(int a, int b) { //union the sets of a and
28         //b
29         p[p[a]] = p[b];
30     }
31 };
32 int main()
33 {
34     piii e[maxn];

```

```

33 int n,m; cin >> n >> m;
34 for(int i=0; i<m; i++)
35     cin >> e[i].x >> e[i].y >> e[i].w;
36 UFT uft(n); sort(e,e+m); //sort the edges
37 int cnt = 0, cost = 0;
38 for(int i=0; i<m && cnt<n-1; i++)
39 {
40     if( uft.same(e[i].x,e[i].y) ) continue;
41     uft.uni(e[i].x,e[i].y);
42     cnt++; cost += e[i].w;
43 }
44 if(cnt<n-1) cout << "Not connected!\n";
45 else cout << "Cost = " << cost << endl;
46 return 0;
47 }

```

3.6 Prim

```

1 // Queries minimum path of spanning tree of a graph, where
  // all vertices are connected, using Prim's algorithm.
2 // The n vertices are supposed to be marked [0, n); edge
  // should have size n.
3 int minpath(const vector<vector<pii>>& edge) {
4     const int N = edge.size();
5     bool vis[N] = {0};
6     priority_queue<pii, vector<pii>, greater<pii>> q;
7
8     vis[0] = true;
9     int nvis = 1;
10    for (auto& e : edge[0]) q.push(e);
11
12    while (nvis < N) {
13        int d = q.top().first;
14        int v = q.top().second;
15        q.pop();
16        if (vis[v]) continue;
17        cout << d << endl;
18        vis[v] = true;
19        if (++nvis == N) return d;
20        for (auto& e : edge[v]) {
21            if (!vis[e.second]) q.emplace(d + e.first, e.second);
22        }
23    }
24    throw "Never reaches here.";
25 }

```

3.7 LCA

```

1 //LCA //Tarjan's algorithm
2 #define maxv 100
3 int LCA[maxv][maxv]; //Lowest common ancestor
4 vector<int> v[maxv]; //adjacency lists
5 int p[maxv]; //parent
6 bool visit[maxv]; //false
7 int n; //the number of vertex
8 void init() { for(int i=0; i<n; i++) p[i]=i; }
9 int parent(int x) { return (p[x]==x) ? (p[x]=parent(x)) : x; }
10 int dfs(int u) {

```

```

11     visit[u] = true;
12     for(int i=0; i<n; i++)
13         if(visit[i]) LCA[i][u] = LCA[u][i] = parent(i);
14     for(int i=v[u].size()-1; i>=0; i--) {
15         int uu = v[u][i];
16         if(!visit[uu]) { dfs(uu); p[uu] = u; }
17     }
18 }
19 void Tarjan() { init(); dfs(0); }

```

3.8 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
9 // 0 base
10 struct TarjanSCC{
11     static const int MAXN = 1000006;
12     int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
13     vector<int> G[MAXN];
14     stack<int> stk;
15     bool ins[MAXN];
16     void tarjan(int u) {
17         dfn[u] = low[u] = ++count;
18         stk.push(u);
19         ins[u] = true;
20         for(auto v:G[u]) {
21             if(!dfn[v]) {
22                 tarjan(v);
23                 low[u] = min(low[u], low[v]);
24             } else if(ins[v]) {
25                 low[u] = min(low[u], dfn[v]);
26             }
27         }
28         if(dfn[u] == low[u]) {
29             int v;
30             do {
31                 v = stk.top(); stk.pop();
32                 scc[v] = scn;
33                 ins[v] = false;
34             } while(v != u);
35             scn++;
36         }
37     }
38     void getSCC(){
39         memset(dfn,0,sizeof(dfn));
40         memset(low,0,sizeof(low));
41         memset(ins,0,sizeof(ins));
42         memset(scc,0,sizeof(scc));
43         count = scn = 0;
44         for(int i = 0; i < n; i++) {
45             if(!dfn[i]) tarjan(i);
46         }
47     } SCC;

```

3.9 Min Mean Cycle

```

1 #include<cstdio> //for DBL_MAX
2 int dp[MAXN][MAXN]; // 1-base,0(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;
10            tie(u,v,w) = e;
11            dp[t+1][v] = min(dp[t+1][v],dp[t][u]+w);
12        }
13    }
14    double res = DBL_MAX;
15    for(int u=1; u<=n; ++u) {
16        if(dp[n][u]==INF) continue;
17        double val = -DBL_MAX;
18        for(int t=0; t<n; ++t)
19            val = max(val, (dp[n][u]-dp[t][u])*1.0/(n-t));
20        res = min(res, val);
21    }
22    return res;
23 }

```

3.10 2-SAT

```

1 const int MAXN = 2020;
2 struct TwoSAT{
3     static const int MAXv = 2*MAXN;
4     vector<int> GO[MAXv],BK[MAXv],stk;
5     bool vis[MAXv];
6     int SC[MAXv];
7     void imply(int u,int v){ // u imply v
8         GO[u].push_back(v);
9         BK[v].push_back(u);
10    }
11    int dfs(int u,vector<int>*G,int sc){
12        vis[u]=1, SC[u]=sc;
13        for (int v:G[u])if (!vis[v])
14            dfs(v,G,sc);
15        if (G==GO) stk.push_back(u);
16    }
17    int scc(int n=MAXv){
18        memset(vis,0,sizeof(vis));
19        for (int i=0; i<n; i++)
20            if (!vis[i]) dfs(i,GO,-1);
21        memset(vis,0,sizeof(vis));
22        int sc=0;
23        while (!stk.empty()){
24            if (!vis[stk.back()])
25                dfs(stk.back(),BK,sc++);
26            stk.pop_back();
27        }
28    }
29 } SAT;
30 int main(){
31     SAT.scc(2*n);
32     bool ok = 1;
33     for (int i=0; i<n; i++){
34         if (SAT.SC[2*i]==SAT.SC[2*i+1]) ok = 0;

```



```

35     }
36     if (ok) {
37         for (int i=0; i<n; i++)
38             if (SAT.SC[2*i]>SAT.SC[2*i+1])
39                 cout << i << endl;
40     }
41     else puts("NO");
42 }
43 void warshall(){
44     bitset<2003> d[2003];
45     for (int k=0; k<n; k++)
46         for (int i=0; i<n; i++)
47             if (d[i][k]) d[i] |= d[k];
48 }

```

3.11 生成樹數量

```

1 // D : degree-matrix
2 // A : adjacent-matrix
3 // 無向圖
4 // (u,v)
5 // A[u][v]++, A[v][u]++
6 // D[u][u]++, D[v][v]++
7 // G = D-A
8 // abs(det(G去掉i-col和i-row))
9 // 生成樹的數量
10 // 有向圖
11 // A[u][v]++
12 // D[v][v]++ (in-deg)
13 // 以i為root的樹形圖數量
14 // 所有節點都能到達root

```

3.12 BCC_edge

```

1 邊雙連通
2 任意兩點間至少有兩條不重疊的路徑連接。找法：
3 1. 標記出所有的橋
4 2. 對全圖進行 DFS，不走橋，每一次 DFS 就是一個新的邊雙連通
5 // from BCCW
6 struct BccEdge {
7     static const int MXN = 100005;
8     struct Edge { int v, eid; };
9     int n, m, step, par[MXN], dfn[MXN], low[MXN];
10    vector<Edge> E[MXN];
11    DisjointSet djs;
12    void init(int _n) {
13        n = _n; m = 0;
14        for (int i=0; i<n; i++) E[i].clear();
15        djs.init(n);
16    }
17    void add_edge(int u, int v) {
18        E[u].PB({v, m});
19        E[v].PB({u, m});
20        m++;
21    }
22    void DFS(int u, int f, int f_eid) {
23        par[u] = f;
24        dfn[u] = low[u] = step++;

```

```

25    for (auto it:E[u]) {
26        if (it.eid == f_eid) continue;
27        int v = it.v;
28        if (dfn[v] == -1) {
29            DFS(v, u, it.eid);
30            low[u] = min(low[u], low[v]);
31        } else {
32            low[u] = min(low[u], dfn[v]);
33        }
34    }
35 }
36 void solve() {
37     step = 0;
38     memset(dfn, -1, sizeof(int)*n);
39     for (int i=0; i<n; i++) {
40         if (dfn[i] == -1) DFS(i, i, -1);
41     }
42     djs.init(n);
43     for (int i=0; i<n; i++) {
44         if (low[i] < dfn[i]) djs.uni(i, par[i]);
45     }
46 }
47 } graph;

```

4 Flow_Matching

4.1 Dinic

```

1 const int maxn = 1e5 + 10;
2 const int INF = 1e9;
3 const long long INF64 = 1e18;
4 struct edge{
5     int to, cap, rev;
6 };
7 vector<edge> G[maxn];
8 int n, m, s, t, a, b, c, iter[maxn], level[maxn];
9 void bfs(int s) {
10     memset(level, -1, sizeof(level));
11     queue<int> q;
12     level[s] = 0;
13     q.push(s);
14     while (q.size()) {
15         int u = q.front(); q.pop();
16         for (edge e: G[u]) {
17             if (e.cap > 0 && level[e.to] < 0) {
18                 level[e.to] = level[u] + 1;
19                 q.push(e.to);
20             }
21         }
22     }
23 }
24 int dfs(int v, int t, int f) {
25     if (v == t) return f;
26     for (int &i = iter[v]; i < G[v].size(); i++) {
27         edge &e = G[v][i];
28         if (e.cap > 0 && level[v] < level[e.to]) {
29             int d = dfs(e.to, t, min(f, e.cap));
30             if (d > 0) {
31                 e.cap -= d;
32                 G[e.to][e.rev].cap += d;
33                 return d;

```

```

34     }
35 }
36 return 0;
37 }
38 int dinic(int s, int t) {
39     int flow = 0;
40     while (true) {
41         bfs(s);
42         if (level[t] < 0) return flow;
43         memset(iter, 0, sizeof(iter));
44         int f;
45         while ((f = dfs(s, t, INF)) > 0)
46             flow += f;
47     }
48 }
49 void init(int n) {
50     for (int i = 0; i < n; i++) G[i].clear();
51 }
52 int main() {
53     cin >> n >> m >> s >> t;
54     init(n);
55     while (m--) {
56         cin >> a >> b >> c;
57         G[a].push_back((edge){b, c, (int)G[b].size()});
58         G[b].push_back((edge){a, 0, (int)G[a].size() - 1});
59     }
60     cout << dinic(s, t) << '\n';
61     return 0;
62 }
63 }

```

4.2 KM

```

1 #define MAXN 405
2 #define INF 0x3f3f3f3f3f3f3f3f
3 int n; // 1-base, 0表示沒有匹配
4 LL g[MAXN][MAXN]; //input graph
5 int My[MAXN], Mx[MAXN]; //output match
6 LL lx[MAXN], ly[MAXN], pa[MAXN], Sy[MAXN];
7 bool vx[MAXN], vy[MAXN];
8 void augment(int y) {
9     for (int x, z; y; y = z) {
10         x = pa[y], z = Mx[x];
11         My[y] = x, Mx[x] = y;
12     }
13 }
14 void bfs(int st) {
15     for (int i=1; i<=n; ++i)
16         Sy[i] = INF, vx[i]=vy[i]=0;
17     queue<int> q; q.push(st);
18     while(1) {
19         while(q.size()) {
20             int x=q.front(); q.pop();
21             vx[x]=1;
22             for (int y=1; y<=n; ++y) if (!vy[y]) {
23                 LL t = lx[x]+ly[y]-g[x][y];
24                 if (t==0) {
25                     pa[y]=x;
26                     if (!My[y]) {augment(y); return;}
27                     vy[y]=1, q.push(My[y]);
28                 }
29                 else if (Sy[y]>t) pa[y]=x, Sy[y]=t;
30             }

```

```

31     }
32     LL cut = INF;
33     for(int y=1; y<=n; ++y)
34         if(!vy[y]&&cut>Sy[y]) cut=Sy[y];
35     for(int j=1; j<=n; ++j) {
36         if(vx[j]) lx[j] -= cut;
37         if(vy[j]) ly[j] += cut;
38         else Sy[j] -= cut;
39     }
40     for(int y=1; y<=n; ++y) {
41         if(!vy[y]&&Sy[y]==0) {
42             if(!My[y]){augment(y);return;}
43             vy[y]=1, q.push(My[y]);
44         }
45     }
46 }
47 }
48 LL KM() {
49     memset(My,0,sizeof(int)*(n+1));
50     memset(Mx,0,sizeof(int)*(n+1));
51     memset(ly,0,sizeof(LL)*(n+1));
52     for(int x=1; x<=n; ++x){
53         lx[x] = -INF;
54         for(int y=1; y<=n; ++y)
55             lx[x] = max(lx[x],g[x][y]);
56     }
57     for(int x=1; x<=n; ++x) bfs(x);
58     LL ans = 0;
59     for(int y=1; y<=n; ++y) ans += g[My[y]][y];
60     return ans;
61 }

```

4.3 Ford Fulkerson

```

1 const int maxn = 1e5 + 10, INF = 1e9;
2 const long long INF64 = 1e18;
3 struct edge{
4     int to, cap, rev;
5 };
6 vector<edge> G[maxn];
7 int n, m, s, t, a, b, c;
8 bool vis[maxn];
9 int dfs(int v, int t, int f) {
10     cout << v << ' ' << t << ' ' << f << '\n';
11     if (v == t) return f;
12     vis[v] = true;
13     for (edge &e: G[v]) {
14         if (!vis[e.to] && e.cap > 0) {
15             int d = dfs(e.to, t, min(f, e.cap));
16             if (d > 0) {
17                 e.cap -= d, G[e.to][e.rev].cap += d;
18                 return d;
19             }
20         }
21     }
22     return 0;
23 }
24 int ford_fulkerson(int s, int t) {
25     int flow = 0, f;
26     for (int i = 0; i < n; i++) {
27         cout << i << " : ";
28         for (edge e: G[i])

```

```

29         cout << '(' << e.to << ', ' << e.cap << ')' << ' ' << '\n';
30     }
31 }
32 do {
33     memset(vis, false, sizeof(vis));
34     f = dfs(s, t, INF);
35     for (int i = 0; i < n; i++) {
36         cout << i << " : ";
37         for (edge e: G[i])
38             cout << '(' << e.to << ', ' << e.cap << ')' << '\n';
39     }
40     cout << f << '\n';
41     flow += f;
42 } while (f > 0);
43 return flow;
44 }
45 void init(int n) {
46     for (int i = 0; i < n; i++) G[i].clear();
47 }
48 int main() {
49     cin >> n >> m >> s >> t;
50     init(n);
51     while (m--) {
52         cin >> a >> b >> c;
53         G[a].push_back((edge){b, c, (int)G[b].size()});
54         G[b].push_back((edge){a, 0, (int)G[a].size() - 1});
55     }
56     cout << ford_fulkerson(s, t) << '\n';
57     return 0;
58 }
59 }

```

4.4 Min Cost Max Flow

```

1 template<typename TP>
2 struct MCMF{
3     static const int MAXN=440;
4     static const TP INF=999999999;
5     struct edge{
6         int v,pre;
7         TP r,cost;
8         edge(int v,int pre,TP r,TP cost):v(v),pre(pre),r(r),
9             cost(cost){}
10    };
11    int n,S,T;
12    TP dis[MAXN],PIS,ans;
13    bool vis[MAXN];
14    vector<edge> e;
15    int g[MAXN];
16    void init(int _n){
17        memset(g,-1,sizeof(int)*((n=_n)+1));
18        e.clear();
19    }
20    void add_edge(int u,int v,TP r,TP cost,bool directed=
21        false){
22        e.push_back(edge(v,g[u],r,cost));
23        g[u]=e.size()-1;
24        e.push_back(
25            edge(u,g[v],directed?0:r,-cost));
26        g[v]=e.size()-1;
27    }
28 }

```

```

TP augment(int u,TP CF){
    if(u==T||!CF)return ans+=PIS*CF,CF;
    vis[u]=1;
    TP r=CF,d;
    for(int i=g[u];~i;i=e[i].pre){
        if(e[i].r&&!e[i].cost&&vis[e[i].v]){
            d=augment(e[i].v,min(r,e[i].r));
            e[i].r-=d;
            e[i^1].r+=d;
            if(!(r-=d))break;
        }
    }
    return CF-r;
}
bool modlabel(){
    for(int u=0;u<=n;++u)dis[u]=INF;
    static deque<int>q;
    dis[T]=0,q.push_back(T);
    while(q.size()){
        int u=q.front();q.pop_front();
        TP dt;
        for(int i=g[u];~i;i=e[i].pre){
            if(e[i^1].r&&(dt=dis[u]-e[i].cost)<dis[e[i].v]){
                if((dis[e[i].v]=dt)<=dis[q.size()?q.front():S]){
                    q.push_front(e[i].v);
                }else q.push_back(e[i].v);
            }
        }
    }
    for(int u=0;u<=n;++u)
        for(int i=g[u];~i;i=e[i].pre)
            e[i].cost+=dis[e[i].v]-dis[u];
    return PIS+=dis[S], dis[S]<INF;
}
TP mincost(int s,int t){
    S=s,T=t;
    PIS=ans=0;
    while(modlabel()){
        do memset(vis,0,sizeof(bool)*(n+1));
        while(augment(S,INF));
    }return ans;
}
};

```

4.5 Hopcroft Karp

```

1 // https://github.com/voidrank/acm-icpc-Library/blob/master/
2 // code/hopcroft-karp.cpp
3 int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
4 vector<int> edge[maxn]; // for Left
5 bool dfs(int u) {
6     vis[u] = true;
7     for (vector<int>::iterator it = edge[u].begin(); it !=
8         edge[u].end(); ++it) {
9         int v = pr2[*it];
10        if (v == -1 || (!vis[v] && level[u] < level[v] && dfs
11            (v))) {
12            pr[u] = *it, pr2[*it] = u;
13            return true;
14        }
15    }
16    return false;
17 }

```

```

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

```

4.6 SW-MinCut

```

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

```

```

30     for (int i=0,x,y; i<n-1; i++) {
31         search(x,y);
32         res = min(res,wei[y]);
33         del[y] = 1;
34         for (int j=0; j<n; j++)
35             edge[x][j] = (edge[j][x] += edge[y][j]);
36     }
37     return res;
38 }
39 } graph;

```

4.7 Stable Marriage

```

1 // 演算法筆記
2 1. N位男士各自向自己最喜愛的女士求婚。
3 2. N位女士各自從自己的求婚者中，挑最喜愛的那位男士訂婚，但是
   往後可背約。
4 沒有求婚者的女士，就只好等等。
5 3. 失敗的男士們，只好各自向自己次喜愛的女士求婚。
6 4. N位女士各自從自己的求婚者中，挑最喜歡的那位男士訂婚，但是
   往後可背約。
7 已訂婚卻有更喜愛的女士求婚的女士，就毀約，改為與此男士訂
   婚。
8 沒有求婚者的女士，就只好再等等。
9 5. 重複3. 4.直到形成N對伴侶為止。
10 // Jinkela
11 queue<int> Q;
12 for ( i : 所有考生 ) {
13     設定在第0志願;
14     Q.push(考生i);
15 }
16 while(Q.size()){
17     當前考生=Q.front();Q.pop();
18     while ( 此考生未分發 ) {
19         指標移到下一志願;
20         if ( 已經沒有志願 or 超出志願總數 ) break;
21         計算該考生在該科系加權後的總分;
22         if ( 不符合科系需求 ) continue;
23         if ( 目前科系有餘額 ) {
24             依加權後分數高低順序將考生id加入科系錄取名單中;
25             break;
26         }
27         if ( 目前科系已額滿 ) {
28             if ( 此考生成績比最低分數還高 ) {
29                 依加權後分數高低順序將考生id加入科系錄取名單;
30                 Q.push(被踢出的考生);
31             }
32         }
33     }
34 }

```

5 Math

5.1 Matrix

```

1 template<typename T>
2 struct Matrix{
3     using rt = std::vector<T>;
4     using mt = std::vector<rt>;
5     using matrix = Matrix<T>;
6     int r,c;
7     mt m;
8     Matrix(int r,int c):r(r),c(c),m(r,rt(c)){
9         rt& operator[](int i){return m[i];}
10    matrix operator+(const matrix &a){
11        matrix rev(r,c);
12        for(int i=0; i<r; ++i)
13            for(int j=0; j<c; ++j)
14                rev[i][j] = m[i][j]+a.m[i][j];
15        return rev;
16    }
17    matrix operator-(const matrix &a){
18        matrix rev(r,c);
19        for(int i=0; i<r; ++i)
20            for(int j=0; j<c; ++j)
21                rev[i][j] = m[i][j]-a.m[i][j];
22        return rev;
23    }
24    matrix operator*(const matrix &a){
25        matrix rev(r,a.c);
26        matrix tmp(a.c,a.r);
27        for(int i=0; i<a.r; ++i)
28            for(int j=0; j<a.c; ++j)
29                tmp[j][i] = a.m[i][j];
30        for(int i=0; i<r; ++i)
31            for(int j=0; j<a.c; ++j)
32                for(int k=0; k<c; ++k)
33                    rev.m[i][j] += m[i][k]*tmp[j][k];
34        return rev;
35    }
36    bool inverse(){
37        Matrix t(r,r+c);
38        for(int y=0; y<r; y++){
39            t.m[y][c+y] = 1;
40            for(int x=0; x<c; ++x)
41                t.m[y][x] = m[y][x];
42        }
43        if( !t.gas() )
44            return false;
45        for(int y=0; y<r; y++){
46            for(int x=0; x<c; ++x)
47                m[y][x] = t.m[y][c+x]/t.m[y][y];
48            return true;
49        }
50    }
51    T gas(){
52        vector<T> lazy(r,1);
53        bool sign = false;
54        for(int i=0; i<r; ++i){
55            if( m[i][i]==0 ){
56                int j=i+1;
57                while(j<r&&!m[j][i])j++;
58                if(j==r)continue;
59                m[i].swap(m[j]);
60                sign=!sign;
61            }
62            for(int j=0;j<r;j++){
63                if(i==j)continue;
64                lazy[j] = lazy[j]*m[i][i];
65                T mx = m[j][i];
66                for(int k=0;k<c;k++){
67                    m[j][k] = m[j][k]*m[i][i]-m[i][k]*mx;
68                }
69            }
70        }
71    }
}

```

```

67     }
68 }
69 T det = sign ? -1 : 1;
70 for(int i=0; i<r; ++i){
71     det = det*m[i][i];
72     det = det/lazy[i];
73     for(auto &j:m[i]) j/=lazy[i];
74 }
75 return det;
76 }
77 };

```

5.2 模逆元

```

1 // Queries value x such that (ax == 1) mod m.
2 ll modinv(ll a, ll m) {
3     if (m == 1) return 0;
4     ll m0 = m, y = 0, x = 1;
5     while (a > 1) {
6         ll q = a / m;
7         ll t = m;
8         m = a % m, a = t;
9         t = y;
10        y = x - q * y;
11        x = t;
12    }
13    if (x < 0) x += m0;
14    return x;
15 }

```

5.3 Euler Function

```

1 // Queries phi(x) value.
2 int phi(int x) {
3     int ret = x;
4     for (int p = 2; p * p <= x; p++) {
5         if (x % p == 0) {
6             while (x % p == 0) x /= p;
7             ret -= ret / p;
8         }
9     }
10    if (x > 1) ret -= ret / x;
11    return ret;
12 }
13 // Queries all phi(x) values where x in [0, n).
14 vector<int> phi_in(int n) {
15     vector<bool> prime(n, 1);
16     vector<int> ret(n);
17     prime[0] = prime[1] = false;
18     for (int i = 0; i < n; i++) ret[i] = i;
19     for (int i = 2; i < n; i++) {
20         if (!prime[i]) continue;
21         ret[i]--;
22         for (int j = i * 2; j < n; j += i) {
23             prime[j] = false;
24             ret[j] = ret[j] / i * (i - 1);
25         }
26     }
27     ret[1] = 0;
28 }

```

```

29     return ret;
30 }

```

5.4 Miller Rabin

```

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 {
14     T ans = 1;
15     while(b)
16     {
17         if(b&1) ans = mul(ans, a, mod);
18         a = mul(a, a, mod);
19         b >>= 1;
20     }
21     return ans;
22 }
23 template<typename T> bool isprime(T n, int num) //num = 3,7
24 {
25     int sprp[3] = {2,7,61}; //int範圍可解
26     //int llsprp[7] =
27     //    {2,325,9375,28178,450775,9780504,1795265022}; //至少
28     //    unsigned long long範圍
29     if(n==2) return true;
30     if(n<2 || n%2==0) return false;
31     //n-1 = u * 2^t
32     int t = 0;
33     T u = n-1;
34     while(u%2==0) u >>= 1, t++;
35     for(int i=0; i<num; i++)
36     {
37         T a = sprp[i]%n;
38         if(a==0 || a==1 || a==n-1) continue;
39         T x = pow(a, u, n);
40         if(x==1 || x==n-1) continue;
41         for(int j=1; j<t; j++)
42         {
43             x = mul(x, x, n);
44             if(x==1) return false;
45             if(x==n-1) break;
46         }
47         if(x!=n-1) return false;
48     }
49     return true;
50 }

```

5.5 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
5 LL pollorrho(const LL n, const int c) { //循環節長度
6     LL a=1, b=1;
7     a=func(a, n, c)%n;
8     b=func(b, n, c)%n; b=func(b, n, c)%n;
9     while(gcd(abs(a-b), n) != 1) {
10        a=func(a, n, c)%n;
11        b=func(b, n, c)%n; b=func(b, n, c)%n;
12    }
13    return gcd(abs(a-b), n);
14 }
15
16 void prefactor(LL &n, vector<LL> &v) {
17     for(int i=0; i<12; ++i) {
18         while(n%prime[i]==0) {
19             v.push_back(prime[i]);
20             n/=prime[i];
21         }
22     }
23 }
24
25 void smallfactor(LL n, vector<LL> &v) {
26     if(n<MAXPRIME) {
27         while(isp[(int)n]) {
28             v.push_back(isp[(int)n]);
29             n/=isp[(int)n];
30         }
31         v.push_back(n);
32     } else {
33         for(int i=0; i<primecnt&&prime[i]*prime[i]<=n; ++i) {
34             while(n%prime[i]==0) {
35                 v.push_back(prime[i]);
36                 n/=prime[i];
37             }
38         }
39         if(n!=1) v.push_back(n);
40     }
41 }
42
43 void comfactor(const LL &n, vector<LL> &v) {
44     if(n<1e9) {
45         smallfactor(n, v);
46         return;
47     }
48     if(Isprime(n)) {
49         v.push_back(n);
50         return;
51     }
52     LL d;
53     for(int c=3; ++c) {
54         d = pollorrho(n, c);
55         if(d!=n) break;
56     }
57     comfactor(d, v);
58     comfactor(n/d, v);
59 }
60
61 void Factor(const LL &x, vector<LL> &v) {
62     LL n = x;
63     if(n==1) { puts("Factor 1"); return; }
64     prefactor(n, v);
65     if(n==1) return;

```

```

66 comfactor(n,v);
67 sort(v.begin(),v.end());
68 }
69
70 void AllFactor(const LL &n,vector<LL> &v) {
71     vector<LL> tmp;
72     Factor(n,tmp);
73     v.clear();
74     v.push_back(1);
75     int len;
76     LL now=1;
77     for(int i=0;i<tmp.size();++i) {
78         if(i==0 || tmp[i]!=tmp[i-1]) {
79             len = v.size();
80             now = 1;
81         }
82         now*=tmp[i];
83         for(int j=0;j<len;++j)
84             v.push_back(v[j]*now);
85     }
86 }

```

5.6 快速冪

```

1 // Queries  $a^p$ .
2 ll fastpow(ll a, int p) {
3     ll ret = 1;
4     while (p) {
5         if (p & 1) ret *= a;
6         a *= a, p >>= 1;
7     }
8     return ret;
9 }
10
11 // Queries  $(a^p) \bmod m$ .
12 ll fastpow(ll a, ll p, ll m) {
13     ll ret = 1;
14     while (p) {
15         if (p & 1) ret = ret * a % m;
16         a = a * a % m, p >>= 1;
17     }
18     return ret;
19 }

```

5.7 實根

```

1 //  $an^x + \dots + a_1x + a_0 = 0$ ;
2 int sign(double x){
3     return x < -eps ? -1 : x > eps;
4 }
5
6 double get(const vector<double>&coef, double x){
7     double e = 1, s = 0;
8     for(auto i : coef) s += i*e, e *= x;
9     return s;
10 }
11
12 double find(const vector<double>&coef, int n, double lo,
13             double hi){
14     double sign_lo, sign_hi;

```

```

14 if( !(sign_lo = sign(get(coef,lo))) ) return lo;
15 if( !(sign_hi = sign(get(coef,hi))) ) return hi;
16 if(sign_lo * sign_hi > 0) return INF;
17 for(int stp = 0; stp < 100 && hi - lo > eps; ++stp){
18     double m = (lo+hi)/2.0;
19     int sign_mid = sign(get(coef,m));
20     if(!sign_mid) return m;
21     if(sign_lo*sign_mid < 0) hi = m;
22     else lo = m;
23 }
24 return (lo+hi)/2.0;
25 }
26
27 vector<double> cal(vector<double>coef, int n){
28     vector<double>res;
29     if(n == 1){
30         if(sign(coef[1])) res.pb(-coef[0]/coef[1]);
31         return res;
32     }
33     vector<double>dcoef(n);
34     for(int i = 0; i < n; ++i) dcoef[i] = coef[i+1]*(i+1);
35     vector<double>droot = cal(dcoef, n-1);
36     droot.insert(droot.begin(), -INF);
37     droot.pb(INF);
38     for(int i = 0; i+1 < droot.size(); ++i){
39         double tmp = find(coef, n, droot[i], droot[i+1]);
40         if(tmp < INF) res.pb(tmp);
41     }
42     return res;
43 }
44
45 int main () {
46     vector<double>ve;
47     vector<double>ans = cal(ve, n);
48     // 視情況把答案 +eps · 避免 -0
49 }

```

5.8 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)
31             for B in S])
32         else state[A] = mex(set(SG(B) for B in next_states(A)))
33     } return state[A]
34 }

```

5.9 外星模運算

```

1 //  $a[0]^a[1]^a[2]^{\dots}$ 
2 #define maxn 1000000
3 int euler[maxn+5];
4 bool is_prime[maxn+5];
5 void init_euler(){
6     is_prime[1]=1; // 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){ //  $a^b \bmod$ 
19     LL ans=1;
20     for(;b;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];
42 int t,mod;
43 int main(){
44     init_euler();
45     scanf("%d",&t);

```

```

46 #define n 4
47 while(t--){
48     for(int i=0;i<n;++i)scanf("%Lld",&a[i]);
49     scanf("%d",&mod);
50     printf("%Lld\n",high_pow(a,n,mod));
51 }
52 return 0;
53 }

```

5.10 FFT

```

1 template<typename T,typename VT=vector<complex<T> > >
2 struct FFT{
3     const T pi;
4     FFT(const T pi=acos((T)-1)):pi(pi){}
5     unsigned bit_reverse(unsigned 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&0xFF0F0F0FU)>>4);
9         a=((a&0x00FF00FFU)<<8)|((a&0xFFFF00FFU)>>8);
10        a=((a&0x0000FFFFU)<<16)|((a&0xFFFF0000U)>>16);
11        return a>>(32-len);
12    }
13    void fft(bool is_inv,VT &in,VT &out,int N){
14        int bitlen=__lg(N),num=is_inv?-1:1;
15        for(int i=0;i<N;++i) out[bit_reverse(i,bitlen)]=in[i];
16        for(int step=2;step<=N;step<=1){
17            const int mh=step>>1;
18            for(int i=0;i<mh;++i){
19                complex<T> wi=exp(complex<T>(0,i*num*pi/mh));
20                for(int j=i;j<N;j+=step){
21                    int k=j+mh;
22                    complex<T> u=out[j], t=wi*out[k];
23                    out[j]=u+t;
24                    out[k]=u-t;
25                }
26            }
27        }
28        if(is_inv) for(int i=0;i<N;++i) out[i]/=N;
29    }
30 };

```

6 String

6.1 Rolling Hash

```

1 // Queires the first index where sub appears in str; -1
   indicates no matching.
2 int rollhash(string& str, string& sub) {
3     const int X = 1e6 + 99, MOD = 1e9 + 9;
4     ll xx = 1;
5     for (int i = 0; i < sub.size(); i++) xx = xx * X % MOD;
6     ll subhash = 0;
7     for (char c : sub) subhash = (subhash * X + c) % MOD;
8
9     vector<ll> hash = {0};
10    for (int i = 0; i < str.size(); i++) {

```

```

11        hash.push_back((hash.back() * X + str[i]) % MOD);
12        if (i >= sub.size()) {
13            ll h = (hash.back() - hash[i - sub.size() + 1] *
14                xx % MOD + MOD) % MOD;
15            if (h == subhash) return i - sub.size() + 1;
16        }
17        return -1;
18    }

```

6.2 Trie

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 class Trie
4 {
5 private:
6     struct Node
7     {
8         int cnt = 0;
9         int sum = 0;
10        Node *tr[128] = {};
11        ~Node()
12        {
13            for(int i=0;i<128;i++)
14                if(tr[i])delete tr[i];
15        }
16    };
17    Node* root;
18 public:
19    void insert(char *s)
20    {
21        Node* ptr = root;
22        for(;;s;s++)
23        {
24            if(!ptr->tr[*s])
25                ptr->tr[*s] = new Node();
26            ptr = ptr->tr[*s];
27            ptr->sum++;
28        }
29        ptr->cnt++;
30    }
31    inline int count(char *s)
32    {
33        Node *ptr = find(s);
34        return ptr ? ptr->cnt : 0;
35    }
36    Node* find(char *s)
37    {
38        Node* ptr = root;
39        for(;;s;s++)
40        {
41            if(!ptr->tr[*s])return 0;
42            ptr = ptr->tr[*s];
43        }
44        return ptr;
45    }
46    bool erase(char *s)
47    {
48        Node *ptr = find(s);
49        if(!ptr)return false;
50        int num = ptr->cnt;
51        if(!num)return false;

```

```

52        ptr = root;
53        for(;;s;s++)
54        {
55            Node *tmp = ptr;
56            ptr = ptr->tr[*s];
57            ptr->sum -= num;
58            if(!ptr->sum)
59            {
60                delete ptr;
61                tmp->tr[*s] = 0;
62                return true;
63            }
64        }
65    }
66    Trie(){root = new Node();}
67    ~Trie(){delete root;}
68 };
69
70 int main()
71 {
72     Trie *trie = new Trie();
73     string op;
74     char s[20];
75     while(cin>>op)
76     {
77         if(op=="END")
78         {
79             delete trie;
80             break;
81         }
82         else if(op=="insert")
83         {
84             cin >> s;
85             trie->insert(s);
86         }
87         else if(op=="erase")
88         {
89             cin >> s;
90             if(trie->erase(s))printf("Success erase");
91             else printf("Fail erase");
92         }
93         else if(op=="count")
94         {
95             cin >> s;
96             printf("%d\n",trie->count(s));
97         }
98     }
99     return 0;
100 }

```

6.3 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]
46                 ].efl;
47             q.push_back(t);
48             ++qe;
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
52     次數O(N*M)*/
53     int match_0(const char *s){
54         int ans = 0, id, p = 0, i;
55         for(i=0; s[i]; i++){
56             id = s[i]-L;
57             while(!S[p].next[id] && p) p = S[p].fail;
58             if(!S[p].next[id]) continue;
59             p = S[p].next[id];
60             ++S[p].cnt_dp; /*匹配成功則它所有後綴都可以被匹配(DP計算)*/
61         }
62         for(i=qe-1; i>=0; --i){
63             ans += S[q[i]].cnt_dp * S[q[i]].ed;
64             if(~S[q[i]].fail) S[S[q[i]].fail].cnt_dp += S[q[i]].cnt_dp;
65         }
66         return ans;
67     }
68     /*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)*/
69     int match_1(const char *s) const{
70         int ans = 0, id, p = 0, t;
71         for(int i=0; s[i]; i++){
72             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nA)的所有相異字串各恰一次並傳回次數O(N*M
83     ^{1/3})*/
84 int match_2(const char *s){
85     int ans=0, id, p=0, t;
86     ++vt;
87     /*把標記vt+=1. 只要vt沒溢位. 所有S[p].vis==vt就會變成
88     false
89     這種利用vt的方法可以O(1)歸零vis陣列*/
90     for(int i=0; s[i]; i++){
91         id = s[i]-L;
92         while(!S[p].next[id]&&p) p = S[p].fail;
93         if(!S[p].next[id]) continue;
94         p = S[p].next[id];
95         if(S[p].ed && S[p].vis!=vt){
96             S[p].vis = vt;
97             ans += S[p].ed;
98         }
99         for(t=S[p].efl; ~t && S[t].vis!=vt; t=S[t].efl){
100             S[t].vis = vt;
101             ans += S[t].ed; /*因為都走efl邊所以保證匹配成功*/
102         }
103     }
104     return ans;
105 }
106 /*把AC自動機變成真的自動機*/
107 void evolution(){
108     for(qs=1; qs!=qe;){
109         int p = q[qs++];
110         for(int i=0; i<=R-L; i++){
111             if(S[p].next[i]==0) S[p].next[i] = S[S[p].fail].next[i];
112         }
113     }
114 }

```

6.4 KMP

```

1 // KMP fail function.
2 int* kmp_fail(string& s) {
3     int* f = new int[s.size()];
4     int p = f[0] = -1;
5     for (int i = 1; s[i]; i++) {
6         while (p != -1 && s[p + 1] != s[i]) p = f[p];
7         if (s[p + 1] == s[i]) p++;
8         f[i] = p;
9     }
10    return f;
11 }
12
13 // Queries the counts sub appears in str.
14 int kmp_count(string& str, string& sub) {
15     int* fail = kmp_fail(sub);

```

```

16 int p = -1, ret = 0;
17 for (int i = 0; i < str.size(); i++) {
18     while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
19     if (sub[p + 1] == str[i]) p++;
20     if (p == sub.size() - 1) p = fail[p], ret++;
21 }
22 delete[] fail;
23 return ret;
24 }
25
26 // Queries the first index where sub appears in str; -1
27 // indicates no matching.
28 int kmp(string& str, string& sub) {
29     int* fail = kmp_fail(sub);
30     int i, j = 0;
31     while (i < str.size() && j < sub.size()) {
32         if (sub[j] == str[i])
33             i++, j++;
34         else if (j == 0)
35             i++;
36         else
37             j = fail[j - 1] + 1;
38     }
39     delete[] fail;
40     return j == sub.size() ? (i - j) : -1;
41 }

```

6.5 Z

```

1 //資芽2018
2 #include<iostream>
3 #include<string>
4 using namespace std;
5 void z_build(string& s, int *z)
6 {
7     int bst = z[0] = 0;
8     for(int i=1; s[i]; i++){
9         if( z[bst]+bst<i ) z[i] = 0;
10        else z[i] = min( z[bst]+bst-i , z[i-bst] );
11        while( s[z[i]] == s[i+z[i]] ) z[i]++;
12        if( z[i]+i > z[bst]+bst ) bst = i;
13    }
14 }
15
16 //s在t中出現幾次
17 int z_match(string& s, string& t)
18 {
19     int ans = 0;
20     int lens = s.length(), lent = t.length();
21     int* z = new int[lens+lent+5];
22     string st = s+"$"+t;
23     z_build(st, z);
24     for(int i=lens+1; i<=lens+lent; i++){
25         if(z[i]==lens) ans++;
26     }
27     return ans;
28 }
29 int main()
30 {
31     string s, t;
32     cin >> s >> t;
33     cout << z_match(s, t) << endl;
34     return 0;
35 }

```

6.6 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 strcmp(s+(int*)i, s+(int*)j, N);
7 }
8 // 此處便宜行事，採用  $O(N^2 \log N)$  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 }

```

6.7 Suffix_Array_LCP

```

1 #define radix_sort(x,y){\
2     for(i=0;i<A;++i)c[i]=0;\
3     for(i=0;i<n;++i)c[x[y[i]]]++;\
4     for(i=1;i<A;++i)c[i]+=c[i-1];\
5     for(i=n-1;~i;--i)sa[--c[x[y[i]]]]=y[i];\
6 }
7 #define AC(r,a,b)\
8     r[a]!=r[b]||a+k>n||r[a+k]!=r[b+k]
9 void suffix_array(const char *s,int n,int *sa,int *rank,int *
    tmp,int *c){
10     int A='z'+1,i,k,id=0;
11     for(i=0; i<n; ++i)rank[tmp[i]=i]=s[i];
12     radix_sort(rank,tmp);
13     for(k=1; id<n-1; k<=1){
14         for(id=0,i=n-k; i<n; ++i) tmp[id++]=i;
15         for(i=0; i<n; ++i)
16             if(sa[i]>=k) tmp[id++]=sa[i]-k;
17         radix_sort(rank,tmp);

```

```

18         swap(rank,tmp);
19         for(rank[sa[0]]=id=0,i=1; i<n; ++i)
20             rank[sa[i]] = id+=AC(tmp,sa[i-1],sa[i]);
21         A = id+1;
22     }
23 }
24 //h:高度數組 sa:後綴數組 rank:排名
25 void suffix_array_lcp(const char *s,int len,int *h,int *sa,
    int *rank){
26     for(int i=0; i<len; ++i)rank[sa[i]]=i;
27     for(int i=0,k=0; i<len; ++i){
28         if(rank[i]==0)continue;
29         if(k)--k;
30         while(s[i+k]==s[sa[rank[i]-1]+k])++k;
31         h[rank[i]]=k;
32     }
33     h[0]=0; // h[k]=Lcp(sa[k],sa[k-1]);
34 }

```

6.8 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 {
7     int i = 0;
8     while (a-i>=0 && b+i<N && s[a-i] == s[b+i]) i++;
9 }
10 void longest_palindromic_substring()
11 {
12     int N = strlen(t);
13     // t穿插特殊字元，存放到s。
14     // (實際上不會這麼做，都是細算索引值。)
15     memset(s, '.', N*2+1);
16     for (int i=0; i<N; ++i) s[i*2+1] = t[i];
17     N = N*2+1;
18     // s[N] = '\0'; // 可做可不做
19     // Manacher's Algorithm
20     z[0] = 1;
21     L = R = 0;
22     for (int i=1; i<N; ++i) {
23         int ii = L - (i - L); // i的映射位置
24         int n = R + 1 - i;
25         if (i > R) {
26             z[i] = extend(i, i);
27             L = i;
28             R = i + z[i] - 1;
29         }
30         else if (z[ii] == n) {
31             z[i] = n + extend(i-n, i+n);
32             L = i;
33             R = i + z[i] - 1;
34         }
35         else z[i] = min(z[ii], n);
36     }
37 }
38 // 尋找最長迴文子串的長度。
39 int n = 0, p = 0;
40 for (int i=0; i<N; ++i)

```

```

41     if (z[i] > n) n = z[p = i];
42     // 記得去掉特殊字元。
43     cout << "最長迴文子串的長度是" << (n-1) / 2;
44     // 印出最長迴文子串，記得別印特殊字元。
45     for (int i=p-z[p]+1; i<=p+z[p]-1; ++i)
46         if (i & 1) cout << s[i];
47 }

```

7 Geometry

7.1 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()const{//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
46         //邊，=0在線上<0右邊
47         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
52     上
53     return ori(p)==0 && btw(p)<=0;
54 }
55 T dis2(const point<T> &p, bool is_segment=0) const { // 點跟直線
56     /線段的距離平方
57     point<T> v=p2-p1, v1=p-p1;
58     if(is_segment){
59         point<T> v2=p-p2;
60         if(v.dot(v1)<=0) return v1.abs2();
61         if(v.dot(v2)>=0) return v2.abs2();
62     }
63     T tmp=v.cross(v1);
64     return tmp*tmp/v.abs2();
65 }
66 T seg_dis2(const line<T> &l) const { // 兩線段距離平方
67     return min({dis2(l.p1,1), dis2(l.p2,1), l.dis2(p1,1), l.dis2
68         (p2,1)});
69 }
70 point<T> projection(const point<T> &p) const { // 點對直線的投
71     影
72     point<T> n=(p2-p1).normal();
73     return p-n*(p-p1).dot(n)/n.abs2();
74 }
75 point<T> mirror(const point<T> &p) const {
76     // 點對直線的鏡射，要先呼叫 pton 轉成一般式
77     point<T> R;
78     T d=a*b+b*b;
79     R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
80     R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
81     return R;
82 }
83 bool equal(const line &l) const { // 直線相等
84     return ori(l.p1)==0 && ori(l.p2)==0;
85 }
86 bool parallel(const line &l) const {
87     return (p1-p2).cross(l.p1-l.p2)==0;
88 }
89 bool cross_seg(const line &l) const {
90     return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;
91     // 直線是否交線段
92 }
93 int line_intersect(const line &l) const { // 直線相交情況，-1無
94     限多點，1交於一點，0不相交
95     return parallel(l)?(ori(l.p1)==0?-1:0):1;
96 }
97 int seg_intersect(const line &l) const {
98     T c1=ori(l.p1), c2=ori(l.p2);
99     T c3=l.ori(p1), c4=l.ori(p2);
100     if(c1==0 && c2==0) { // 共線
101         bool b1=btw(l.p1)>=0, b2=btw(l.p2)>=0;
102         T a3=l.btw(p1), a4=l.btw(p2);
103         if(b1 && b2 && a3==0 && a4==0) return 2;
104         if(b1 && b2 && a3>0 && a4==0) return 3;
105         if(b1 && b2 && a3>0 && a4>0) return 0;
106         return -1; // 無限交點
107     } else if(c1*c2<=0 && c3*c4<=0) return 1;
108     return 0; // 不相交
109 }
110
111 point<T> line_intersection(const line &l) const { // 直線交點
112     point<T> a=p2-p1, b=l.p2-l.p1, s=l.p1-p1;
113     // if(a.cross(b)==0) return INF;
114     return p1+a*(s.cross(b)/a.cross(b));
115 }
116 point<T> seg_intersection(const line &l) const { // 線段交點
117     int res=seg_intersect(l);
118     if(res<=0) assert(0);
119     if(res==2) return p1;
120     if(res==3) return p2;
121     return line_intersection(l);
122 }
123
124 template<typename T>
125 struct polygon {
126     polygon() {}
127     vector<point<T> > p; // 逆時針順序
128     T area() const { // 面積
129         T ans=0;
130         for(int i=p.size()-1, j=0; j<(int)p.size(); i=j++)
131             ans+=p[i].cross(p[j]);
132         return ans/2;
133     }
134     point<T> center_of_mass() const { // 重心
135         T cx=0, cy=0, w=0;
136         for(int i=p.size()-1, j=0; j<(int)p.size(); i=j++){
137             T a=p[i].cross(p[j]);
138             cx+=(p[i].x+p[j].x)*a;
139             cy+=(p[i].y+p[j].y)*a;
140             w+=a;
141         }
142         return point<T>(cx/3/w, cy/3/w);
143     }
144     char ahas(const point<T> &t) const { // 點是否在簡單多邊形內，
145         是的話回傳1，在邊上回傳-1，否則回傳0
146         bool c=0;
147         for(int i=0, j=p.size()-1; i<p.size(); j=i++){
148             if(line<T>(p[i], p[j]).point_on_segment(t)) return -1;
149             else if((p[i].y>t.y) != (p[j].y>t.y) &&
150                 t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x
151                 )
152                 c=!c;
153             return c;
154         }
155     }
156     char point_in_convex(const point<T> &x) const {
157         int l=1, r=(int)p.size()-2;
158         while(l<r) { // 點是否在凸多邊形內，是的話回傳1，在邊上回傳
159             -1，否則回傳0
160             int mid=(l+r)/2;
161             T a1=(p[mid]-p[l]).cross(x-p[l]);
162             T a2=(p[mid+1]-p[l]).cross(x-p[l]);
163             if(a1>=0 && a2<=0) {
164                 T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
165                 return res>0?1:(res>=0?-1:0);
166             } else if(a1<0) r=mid-1;
167             else l=mid+1;
168         }
169         return 0;
170     }
171     vector<T> getA() const { // 凸包邊對x軸的夾角
172         vector<T> res; // 一定是遞增的
173         for(size_t i=0; i<p.size(); ++i)
174             res.push_back((p[(i+1)%p.size()]-p[i]).getA());
175         return res;
176     }
177 }
178
179 bool line_intersect(const vector<T> &A, const line<T> &l)
180     const { // O(LogN)
181     int f1=upper_bound(A.begin(), A.end(), (l.p1-l.p2).getA())-
182         A.begin();
183     int f2=upper_bound(A.begin(), A.end(), (l.p2-l.p1).getA())-
184         A.begin();
185     return l.cross_seg(line<T>(p[f1], p[f2]));
186 }
187
188 polygon cut(const line<T> &l) const { // 凸包對直線切割，得到直
189     線L左側的凸包
190     polygon ans;
191     for(int n=p.size(), i=n-1, j=0; j<n; i=j++){
192         if(l.ori(p[i])>=0){
193             ans.p.push_back(p[i]);
194             if(l.ori(p[j])<0)
195                 ans.p.push_back(l.line_intersection(line<T>(p[i], p[
196                     j])));
197         } else if(l.ori(p[j])>0)
198             ans.p.push_back(l.line_intersection(line<T>(p[i], p[j
199                 ])));
200     }
201     return ans;
202 }
203
204 static bool graham_cmp(const point<T> &a, const point<T> &b)
205     { // 凸包排序函數
206     return (a.x<b.x) || (a.x==b.x && a.y<b.y);
207 }
208
209 void graham(vector<point<T> > &s) { // 凸包
210     sort(s.begin(), s.end(), graham_cmp);
211     p.resize(s.size()+1);
212     int m=0;
213     for(size_t i=0; i<s.size(); ++i){
214         while(m>=2 && (p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0) --m;
215         p[m++]=s[i];
216     }
217     for(int i=s.size()-2, t=m+1; i>=0; --i){
218         while(m>=t && (p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0) --m;
219         p[m++]=s[i];
220     }
221     if(s.size()>1) --m;
222     p.resize(m);
223 }
224
225 T diam() { // 直徑
226     int n=p.size(), t=1;
227     T ans=0; p.push_back(p[0]);
228     for(int i=0; i<n; ++i){
229         point<T> now=p[i+1]-p[i];
230         while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i])) t=(t
231             +1)%n;
232         ans=max(ans, (p[i]-p[t]).abs2());
233     }
234     return p.pop_back(), ans;
235 }
236
237 T min_cover_rectangle() { // 最小覆蓋矩形
238     int n=p.size(), t=1, r=1, l;
239     if(n<3) return 0; // 也可以做最小周長矩形
240     T ans=1e99; p.push_back(p[0]);
241     for(int i=0; i<n; ++i){
242         point<T> now=p[i+1]-p[i];
243         while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i])) t=(t
244             +1)%n;
245         while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i])) r=(r+1)%n
246             ;
247     }

```

```

219     if(!l)l=r;
220     while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%280;
221     n;
222     T d=now.abs2();
223     T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(p[l]-p[i]))/d;
224     ans=min(ans,tmp);
225     return p.pop_back(),ans;
226 }
227 T dis2(polygon &p1){//凸包最近距離平方
228     vector<point<T> > &P=p,&Q=p1.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){
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> perpcenter()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 };
340 pair<point3D<T>,point3D<T> > closest_pair(const line3D<T> &l)const{
341     point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
342     point3D<T> N=v1.cross(v2),ab(p1-l.p1);
343     //if(N.abs2()==0)return NULL; 平行或重合
344     T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();//最近點對距離
345     point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1;
346     T t1=(G.cross(d2)).dot(D)/D.abs2();
347     T t2=(G.cross(d1)).dot(D)/D.abs2();
348     return make_pair(p1+d1*t1,l.p1+d2*t2);
349 }
350 bool same_side(const point3D<T> &a,const point3D<T> &b)const{
351     return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
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 &p1)const{
370         point3D<T> e=n.cross(p1.n),v=n.cross(e);
371         T tmp=p1.n.dot(v);//等於0表示平行或重合該平面
372         point3D<T> q=p0+(v*(p1.n.dot(p1.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,
398         d,a).point_in(p);
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.
421                     c]-pt[f.a]));
422                 if(d<=0) next.push_back(f);
423                 int ff=0;
424                 if(d>0) ff=ftop;
425                 else if(d<0) ff=-ftop;
426                 fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
427             }
428             for(auto &f:ans) {
429                 if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
430                     next.emplace_back(f.a,f.b,i);
431                 if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
432                     next.emplace_back(f.b,f.c,i);
433                 if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
434                     next.emplace_back(f.c,f.a,i);
435             }
436             ans=next;
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     };

```

7.2 旋轉卡尺

```

1 typedef pair<long long, long long> pt;
2 const int maxn = 1e6 + 10;
3 pt operator-(const pt& p1, const pt& p2) {

```

```

4     return pt(p1.x - p2.x, p1.y - p2.y);
5 }
6 long long cross(const pt& p1, const pt& p2) {
7     return p1.x * p2.y - p1.y * p2.x;
8 }
9 long long dis(pt a, pt b) {
10    return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b
11        .y);
12 }
13 vector<pt> ch;
14 pt p[maxn];
15 double shoelace_formula(vector<pt> &v) {
16     int n = v.size();
17     double ans = 0;
18     for (int i = 0; i < n; i++)
19         ans += (v[i].x * v[(i + 1) % n].y);
20     for (int i = 0; i < n; i++)
21         ans -= (v[i].y * v[(i + 1) % n].x);
22     return abs(ans / 2);
23 }
24 double farthest_dis(vector<pt> &v) {
25     int k = 1, n = v.size();
26     long long ans = 0;
27     if (n == 2) return dis(v[0], v[1]);
28     for (int i = 0; i < n; i++) {
29         while (llabs(cross(v[i] - v[(k + 1) % n], v[(i + 1) %
30             n] - v[(k + 1) % n])) >= llabs(cross(v[i] - v[k]
31                 ], v[(i + 1) % n] - v[k])))
32             k = (k + 1) % n;
33         ans = max(ans, max(dis(v[i], v[k]), dis(v[(i + 1) % n
34             ], v[k])));
35     }
36     return sqrt(ans);
37 }
38 int n;
39 int main() {
40     cin >> n;
41     for (int i = 0; i < n; i++) cin >> p[i].x >> p[i].y;
42     sort(p, p + n);
43     for (int i = 0; i < n; i++) {
44         while (ch.size() >= 2 && cross(ch[ch.size() - 1] - ch
45             [ch.size() - 2], p[i] - ch[ch.size() - 2]) <= 0)
46             ch.pop_back();
47         ch.push_back(p[i]);
48     }
49     for (int i = n - 2, t = ch.size() + 1; i >= 0; i--) {
50         while (ch.size() >= t && cross(ch[ch.size() - 1] - ch
51             [ch.size() - 2], p[i] - ch[ch.size() - 2]) <= 0)
52             ch.pop_back();
53         ch.push_back(p[i]);
54     }
55     if (n >= 2) ch.pop_back();
56     cout << setprecision(6) << fixed << shoelace_formula(ch)
57         << '\n';
58     cout << setprecision(6) << fixed << farthest_dis(ch) << '
59         \n';
60     return 0;
61 }

```

7.3 最近點對

```

1 template<typename T> struct Point
2 {
3     T x,y;

```

```

4     Point(){}
5     Point(const T &x, const T &y):x(x),y(y){}
6     inline T dist(Point b) {
7         return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y));
8     }
9     static bool cmpx(const Point &a, const Point &b) {
10        if(a.x==b.x) return a.y<b.y;
11        return a.x<b.x;
12    }
13    static bool cmpy(const Point &a, const Point &b) {
14        if(a.y==b.y) return a.x<b.x;
15        return a.y<b.y;
16    }
17 }
18 template<typename T, typename _IT = Point<T>* >
19 void DC(T &d, _IT p, _IT t, int L, int R) //Divide and
20     Conquer //NLgN
21 {
22     if(L>=R) return;
23     int mid = (L+R)>>1;
24     DC(d,p,t,L,mid);
25     DC(d,p,t,mid+1,R);
26     int N = 0;
27     for(int i=mid; i>=L && p[mid].x-p[i].x<d; i--) t[N++] = p
28         [i];
29     for(int i=mid+1; i<=R && p[i].x-p[mid].x<d; i++) t[N++] =
30         p[i];
31     sort(t,t+N,t->cmpy);
32     for(int i=0; i<N-1; i++)
33         for(int j=1; j<=3 && i+j<N; j++)
34             d = min(d,t[i].dist(t[i+j]));
35 }
36 template<typename T, typename _IT = Point<T>* >
37 void closest_pair(T &d, _IT p, _IT t, int n) {
38     sort(p,p+n,p->cmpx); DC(d,p,t,0,n-1);
39 }
40 int main() {
41     Point<double> p[maxn], t[maxn];
42     int n; scanf("%d",&n);
43     for(int i=0; i<n; i++) scanf("%lf%lf",&p[i].x,&p[i].y);
44     double d = INF; closest_pair(d,p,t,n);
45     printf("distance = %lf\n",d);
46     return 0;
47 }

```

7.4 最小覆蓋圓

```

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             {
15                 c=p[i]; r2=0;
16                 for(int j=0; j<i; j++)

```



```

17     if( (p[j]-c).abs2() > r2)
18     {
19         c.x = (p[i].x+p[j].x)/2;
20         c.y = (p[i].y+p[j].y)/2;
21         r2 = (p[j]-c).abs2();
22         for(int k=0; k<j; k++)
23             if( (p[k]-c).abs2() > r2)
24             {
25                 c = circumcenter(p[i],p[j],p[k]);
26                 r2 = (p[i]-c).abs2();
27             }
28     }
29 }
30 }

```

7.5 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.
38             begin();
39         r[i].r = lower_bound(x.begin(), x.end(), r[i].r) - x.
40             begin();
41         r[i].b = lower_bound(y.begin(), y.end(), r[i].b) - y.
42             begin();
43         r[i].t = lower_bound(y.begin(), y.end(), r[i].t) - y.
44             begin();

```

```

41         v.emplace_back(make_pair(r[i].l, 1), make_pair(r[i].b
42             , r[i].t));
43         v.emplace_back(make_pair(r[i].r, -1), make_pair(r[i].
44             b, r[i].t));
45     }
46     sort(v.begin(), v.end(), [](pair<pair<int, int>, pair<int
47         , int>> a, pair<pair<int, int>, pair<int, int>> b){
48         if (a.first.first != b.first.first) return a.first.
49             first < b.first.first;
50         return a.first.second > b.first.second;
51     });
52     for (int i = 0; i < v.size(); i++) {
53         if (i) ans += (x[v[i].first.first] - x[v[i - 1].first
54             .first]) * st[1];
55         modify(1, 0, y.size(), v[i].second.first, v[i].second
56             .second, v[i].first.second);
57     }
58     cout << ans << '\n';
59     return 0;
60 }

```

8 Other

8.1 BuiltIn

```

1 //gcc專用
2 //unsigned int ffs
3 //unsigned long ffsll
4 //unsigned long long ffslll
5 #include<stdio.h>
6 int main()
7 {
8     unsigned int x;
9     while(scanf("%u",&x)==1)
10     {
11         printf("右起第一個1的位置");
12         printf("%d\n",__builtin_ffs(x));
13         printf("左起第一個1之前0的個數:");
14         printf("%d\n",__builtin_clz(x));
15         printf("右起第一個1之後0的個數:");
16         printf("%d\n",__builtin_ctz(x));
17         printf("1的個數:");
18         printf("%d\n",__builtin_popcount(x));
19         printf("1的個數的奇偶性:");
20         printf("%d\n",__builtin_parity(x));
21     }
22     return 0;
23 }

```

8.2 莫隊算法-區間眾數

```

1 using namespace std;
2 const int maxn = 1e6 + 10;
3 struct query {
4     int id, bk, l, r;
5 };
6 int arr[maxn], cnt[maxn], d[maxn], n, m, bk, mx;

```

```

7 pair<int,int> ans[maxn];
8 vector<query> q;
9 bool cmp(query x,query y) {
10     return (x.bk < y.bk || (x.bk == y.bk) && x.r < y.r);
11 }
12 void add(int pos) {
13     d[cnt[arr[pos]]]--;
14     cnt[arr[pos]]++;
15     d[cnt[arr[pos]]]++;
16     if(d[mx + 1] > 0) mx++;
17 }
18 void del(int pos) {
19     d[cnt[arr[pos]]]--;
20     cnt[arr[pos]]--;
21     d[cnt[arr[pos]]]++;
22     if(d[mx] == 0) mx--;
23 }
24 void mo(int n, int m) {
25     sort(q.begin(), q.end(), cmp);
26     for(int i = 0, cl = 1, cr = 0; i < m; i++) {
27         while(cr < q[i].r) add(++cr);
28         while(cl > q[i].l) add(--cl);
29         while(cr > q[i].r) del(cr--);
30         while(cl < q[i].l) del(cl--);
31         ans[q[i].id] = make_pair(mx, d[mx]);
32     }
33 }
34 int main(){
35     cin >> n >> m;
36     bk = (int)sqrt(n + 0.5);
37     for(int i = 1; i <= n; i++)
38         cin >> arr[i];
39     q.resize(m);
40     for(int i = 0; i < m; i++) {
41         cin >> q[i].l >> q[i].r;
42         q[i].id = i, q[i].bk = (q[i].l - 1) / bk;
43     }
44     mo(n, m);
45     for(int i = 0; i < m; i++)
46         cout << ans[i].first << ' ' << ans[i].second << '\n';
47     return 0;
48 }

```

8.3 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 }
33 vector<long long> dp[MAXN][MAXN];
34 vector<bool> neg_INF[MAXN][MAXN];//如果花費是負的可能會有無限
35 小的情形
36 void relax(int l,int r,const CNF &c,long long cost,bool neg_c
37 =0){
38     if(!neg_INF[l][r][c.s]&&(neg_INF[l][r][c.x]||cost<dp[l][r][
39     c.s])){
40         if(neg_c||neg_INF[l][r][c.x]){
41             dp[l][r][c.s]=0;
42             neg_INF[l][r][c.s]=true;
43         }else dp[l][r][c.s]=cost;
44     }
45 }
46 void bellman(int l,int r,int n){
47     for(int k=1;k<=state;++k)
48         for(auto c:cnf)
49             if(c.y==-1)relax(l,r,c,dp[l][r][c.x]+c.cost,k==n);
50 }
51 void cyk(const vector<int> &tok){
52     for(int i=0;i<(int)tok.size();++i){
53         for(int j=0;j<(int)tok.size();++j){
54             dp[i][j]=vector<long long>(state+1,INT_MAX);
55             neg_INF[i][j]=vector<bool>(state+1,false);
56         }
57         dp[i][i][tok[i]]=0;
58         bellman(i,i,tok.size());
59     }
60     for(int r=1;r<(int)tok.size();++r){
61         for(int l=r-1;l>=0;--l){
62             for(int k=1;k<r;++k)
63                 for(auto c:cnf)
64                     if(~c.y)relax(l,r,c,dp[l][k][c.x]+dp[k+1][r][c.y]+c
65                     .cost);
66             bellman(l,r,tok.size());
67         }
68     }
69 }

```

NCTU-PUSHEEN

CODEBOOK

Contents

1 Data_Structure	1	3 Graph	6	5.5 質因數分解	12
1.1 Sparse Table	1	3.1 Dijkstra	6	5.6 快速冪	13
1.2 Binary Indexed Tree	1	3.2 Bellman Ford	7	5.7 實根	13
1.3 Fenwick Tree	1	3.3 Floyd Warshall	7	5.8 SG	13
1.4 線段樹	1	3.4 SPFA	7	5.9 外星模運算	13
1.5 持久化線段樹	2	3.5 Kruskal	7	5.10 FFT	14
1.6 Treap	2	3.6 Prim	8	6 String	14
1.7 Dynamic_KD_tree	3	3.7 LCA	8	6.1 Rolling Hash	14
1.8 Heavy Light	4	3.8 Tarjan	8	6.2 Trie	14
1.9 Link Cut Tree	4	3.9 Min Mean Cycle	8	6.3 AC 自動機	14
2 DP	5	3.10 2-SAT	8	6.4 KMP	15
2.1 Edit Distance	5	3.11 生成樹數量	9	6.5 Z	15
2.2 LCIS	5	3.12 BCC_edge	9	6.6 BWT	16
2.3 Bounded_Knapsack	5	4 Flow_Matching	9	6.7 Suffix_Array_LCP	16
2.4 1D1D	6	4.1 Dinic	9	6.8 LPS	16
		4.2 KM	9	7 Geometry	16
		4.3 Ford Fulkerson	10	7.1 Geometry	16
		4.4 Min Cost Max Flow	10	7.2 旋轉卡尺	19
		4.5 Hopcroft Karp	10	7.3 最近點對	19
		4.6 SW-MinCut	11	7.4 最小覆蓋圖	19
		4.7 Stable Marriage	11	7.5 Rectangle Union Area	20
		5 Math	11	8 Other	20
		5.1 Matrix	11	8.1 BuiltIn	20
		5.2 模逆元	12	8.2 莫隊算法-區間眾數	20
		5.3 Euler Function	12	8.3 CNF	20
		5.4 Miller Rabin	12		