

# 1 DataStructure

## 1.1 1d\_segTree

```

1 void buildst(int l, int r, int idx) //l, r是st的區間
2 {
3     if(l == r){
4         st[idx] = arr[l];
5         return;
6     }
7     int mid = (l+r)/2;
8     buildst(l, mid, idx*2);
9     buildst(mid+1, r, idx*2+1);
10    st[idx] = max(st[idx*2], st[idx*2+1]);
11 }
12
13 ll query(int l, int r, int idx, int L, int R) //L,R是操作的
    區間
14 {
15     if(r < L || R < l) return -INF;
16     if(L <= l && r <= R) return st[idx];
17     int mid = (l+r)/2;
18     return max(query(l, mid, idx*2, L, R), query(mid+1, r,
19         idx*2+1, L, R));
20 }
21
22 void modify(int l, int r, int idx, int x, int v)
23 {
24     if(r < x || x < l) return;
25     if(l == r){
26         st[idx] += v; return;
27     }
28     int mid = (l+r)/2;
29     modify(l, mid, idx*2, x, v);
30     modify(mid+1, r, idx*2+1, x, v);
31     st[idx] = max(st[idx*2], st[idx*2+1]);
32 }

```

## 1.2 2d\_st\_tag

```

1 //二維陣列單點查詢區間加值
2 class St1d
3 {
4 private:
5     ll st[4*N];
6
7 public:
8     void build();
9     void modify(int l, int r, int idx, int L, int R, ll v);
10    ll query(int l, int r, int idx, int x);
11    void down(int idx);
12 };
13
14 void St1d::build(){
15     memset(st, 0, sizeof(st));
16 }
17
18 void St1d::modify(int l, int r, int idx, int L, int R, ll v){
19     if(r < L || R < l) return;
20     if(L <= l && r <= R)

```

```

21     {
22         st[idx] += v;
23         return;
24     }
25     assert(l != r);
26     down(idx);
27     int mid = (l+r)/2;
28     modify(l, mid, idx*2, L, R, v);
29     modify(mid+1, r, idx*2+1, L, R, v);
30 }
31
32 ll St1d::query(int l, int r, int idx, int x){
33     if(x < l || r < x) return 0;
34     if(l == x && r == x) return st[idx];
35     down(idx);
36     int mid = (l+r)/2;
37     ll left = query(l, mid, idx*2, x);
38     ll right = query(mid+1, r, idx*2+1, x);
39     return left+right;
40 }
41
42 void St1d::down(int idx){
43     st[idx*2] += st[idx], st[idx*2+1] += st[idx];
44     st[idx] = 0;
45 }
46
47 ////////////////
48
49 class St2d
50 {
51 private:
52     St1d st[4*N];
53
54 public:
55     void build(int il, int ir, int idx);
56     void modify(int il, int ir, int jl, int jr, int idx, int
57         iL, int iR, int jL, int jR, ll v);
58     ll query(int il, int ir, int jl, int jr, int idx, int i,
59         int j);
60 };
61
62 void St2d::build(int il, int ir, int idx){
63     st[idx].build();
64     if(il == ir) return;
65     int mid = (il+ir)/2;
66     build(il, mid, idx*2);
67     build(mid+1, ir, idx*2+1);
68 }
69
70 void St2d::modify(int il, int ir, int jl, int jr, int idx,
71     int iL, int iR, int jL, int jR, ll v){
72     if(ir < iL || iR < iL) return;
73     if(iL <= il && ir <= iR){
74         st[idx].modify(jl, jr, 1, jL, jR, v); return;
75     }
76     int mid = (il+ir)/2;
77     modify(il, mid, jl, jr, idx*2, iL, iR, jL, jR, v);
78     modify(mid+1, ir, jl, jr, idx*2+1, iL, iR, jL, jR, v);
79 }
80
81 ll St2d::query(int il, int ir, int jl, int jr, int idx, int i
82     , int j){
83     ll tot = 0;
84     if(i < iL || iR < i) return 0;
85     if(iL <= i && i <= iR) tot += st[idx].query(jl, jr, 1, j)
86         ;
87 }

```

```

82     if(il == i && ir == i) return tot;
83     int mid = (il+ir)/2;
84     tot += query(il, mid, jl, jr, idx*2, i, j);
85     tot += query(mid+1, ir, jl, jr, idx*2+1, i, j);
86     return tot;
87 }

```

## 1.3 undo\_disjoint\_set

```

1 struct DisjointSet {
2     // save() is like recursive
3     // undo() is like return
4     int n, fa[MXN], sz[MXN];
5     vector<pair<int*,int*>> h;
6     vector<int> sp;
7     void init(int tn) {
8         n=tn;
9         for (int i=0; i<n; i++) sz[fa[i]=i]=1;
10        sp.clear(); h.clear();
11    }
12    void assign(int *k, int v) {
13        h.PB({k, *k});
14        *k=v;
15    }
16    void save() { sp.PB(SZ(h)); }
17    void undo() {
18        assert(!sp.empty());
19        int last=sp.back(); sp.pop_back();
20        while (SZ(h)!=last) {
21            auto x=h.back(); h.pop_back();
22            *x.F=x.S;
23        }
24    }
25    int f(int x) {
26        while (fa[x]!=x) x=fa[x];
27        return x;
28    }
29    void uni(int x, int y) {
30        x=f(x); y=f(y);
31        if (x==y) return;
32        if (sz[x]<sz[y]) swap(x, y);
33        assign(&sz[x], sz[x]+sz[y]);
34        assign(&fa[y], x);
35    }
36 }djs;

```

## 1.4 treap

```

1 struct Treap {
2     int pri, sz;
3     int rev;
4     ll data, tag; // tag: make-same
5     Treap *l, *r;
6     Treap(ll d):pri(rand()), sz(1), rev(0), data(d), tag(INF)
7         , l(NULL), r(NULL) {}
8     inline void up();
9     inline void down();
10 };
11
12 int size(Treap *t) { return t? t->sz:0; }

```

```

12 ll get_data(Treap *t) { return t? t->data:0; }
13
14 void Treap::up() {
15     if(l) l->down();
16     if(r) r->down();
17     sz = 1+size(l)+size(r);
18 }
19 void Treap::down() {
20     if(tag != INF) {
21         data = tag;
22         if(l) l->tag = tag;
23         if(r) r->tag = tag;
24         tag = INF;
25     }
26     if(rev) {
27         swap(l, r);
28         if(l) l->rev ^= 1;
29         if(r) r->rev ^= 1;
30         rev ^= 1;
31     }
32 }
33 void freeTreap(Treap *t) {
34     if(!t) return;
35     if(t->l) freeTreap(t->l);
36     if(t->r) freeTreap(t->r);
37     delete t;
38 }
39 Treap *merge(Treap *a, Treap *b) {
40     if(!a || !b) return (a? a:b);
41     if(a->pri < b->pri) {
42         a->down();
43         a->r = merge(a->r, b);
44         a->up();
45         return a;
46     } else {
47         b->down();
48         b->l = merge(a, b->l);
49         b->up();
50         return b;
51     }
52 }
53 void split(Treap *o, Treap *&a, Treap *&b, int k) {
54     if(!o) a = b = NULL;
55     else {
56         o->down();
57         if(k >= size(o->l)+1) {
58             a = o;
59             split(o->r, a->r, b, k-size(o->l)-1);
60         } else {
61             b = o;
62             split(o->l, a, b->l, k);
63         }
64         o->up();
65     }
66 }
67
68 Treap* buildTreap(vector<int> &arr) {
69     srand(7122+time(NULL));
70     Treap *tp = NULL;
71     for(auto x : arr)
72         tp = merge(tp, new Treap(x));
73     return tp;
74 }
75 void ins(Treap *&tp, int pos, int x) {
76     Treap *a, *b;
77     split(tp, a, b, pos);

```

```

78     tp = merge(a, merge(new Treap(x), b));
79 }
80 void del(Treap *&tp, int pos, int k) {
81     Treap *a, *b, *c;
82     split(tp, a, b, pos-1);
83     split(b, b, c, k);
84     freeTreap(b);
85     tp = merge(a, c);
86 }
87 void makeSame(Treap *tp, int pos, int k, int val) {
88     Treap *a, *b, *c;
89     split(tp, a, b, pos-1);
90     split(b, b, c, k);
91     b->tag = val;
92     tp = merge(a, merge(b, c));
93 }
94 void rev(Treap *&tp, int pos, int k) {
95     Treap *a, *b, *c;
96     split(tp, a, b, pos-1);
97     split(b, b, c, k);
98     b->rev ^= 1;
99     tp = merge(a, merge(b, c));
100 }

```

## 1.5 disjoint\_set

```

1 // path compression
2 int f[N];
3
4 int findrt(int x)
5 {
6     if(f[x] == x) return x;
7     else return f[x] = findrt(f[x]);
8 }
9
10 int same(int x, int y)
11 {
12     return findrt(x) == findrt(y);
13 }
14
15 void uni(int x, int y)
16 {
17     f[findrt(y)] = findrt(x);
18 }
19
20 void init()
21 {
22     for(int i = 0; i < N; i++) f[i] = i;
23 }
24
25 //union by rank
26 int f[N]; //disjoint set
27 int rk[N]; //union by rank
28
29 int findrt(int x)
30 {
31     if(f[x] == x) return x;
32     else return f[x] = findrt(f[x]);
33 }
34
35 bool same(int x, int y)
36 {
37     return findrt(x) == findrt(y);

```

```

38 }
39
40 void uni(int x, int y)
41 {
42     x = findrt(x), y = findrt(y);
43     if(x == y) return;
44     if(rk[x] < rk[y]) f[x] = y;
45     else if(rk[x] == rk[y]) f[x] = y, rk[y]++;
46     else f[y] = x;
47 }
48
49 void init()
50 {
51     for(int i = 0; i < N; i++) f[i] = i, rk[i] = 0;
52 }

```

## 1.6 Matrix

```

1 ll SZ,MOD;
2 const int MAXSZ=105;
3
4 struct Mat{
5     ll m[MAXSZ][MAXSZ];
6     Mat(){memset(m, 0, sizeof(m));}
7 };
8
9 Mat matMul(const Mat &A, const Mat &B){
10     Mat rtn;
11     for(int i = 0; i < SZ; i++)
12         for(int k = 0; k < SZ; k++){
13             if(A.m[i][k])for(int j=0; j<SZ; j++){
14                 rtn.m[i][j]+=(A.m[i][k]*B.m[k][j]);
15             }
16         }
17     return rtn;
18 }
19 //B is of size SZ
20 vector<ll> matMul(const Mat &A, const vector<ll> &B)
21 {
22     vector<ll> rtn(SZ,0);
23     for(int i = 0; i < SZ; i++)
24         for(int j = 0; j < SZ; j++){
25             rtn[i]=(rtn[i]+A.m[i][j]*B[j]);
26         }
27     return rtn;
28 }
29 Mat matPow(Mat& M, ll p){
30     if(p == 0){
31         Mat iden;
32         for(int i=0;i<SZ;i++)iden.m[i][i]=1;
33         return iden;
34     }
35     if(p == 1)return M;
36     Mat rtn = matPow(M, p/2);
37     if(p&1)return matMul(matMul(rtn, rtn), M);
38     else return matMul(rtn, rtn);
39 }

```

## 1.7 1d\_segTree\_tag

```

1 //線段樹懶人標記：一維陣列區間加值區間乘值區間查詢總和
2 struct Node //data = data*mul+add;
3 {
4     ll data, mul, add;
5 };
6
7 ll getval(int l, int r, int idx){
8     return (st[idx].data*st[idx].mul%MD+(r-l+1)*st[idx].add%
9         MD)%MD;
10 }
11 void up(int l, int r, int idx){
12     int mid = l+(r-l)/2;
13     st[idx].data = (getval(l, mid, idx*2)+getval(mid+1, r,
14         idx*2+1))%MD;
15 }
16 void down(int l, int r, int idx){
17     st[idx].data = getval(l, r, idx);
18     int lson = idx*2, rson = idx*2+1;
19     if(l != r){
20         st[lson].mul = st[lson].mul*st[idx].mul%MD;
21         st[lson].add = (st[lson].add*st[idx].mul+st[idx].add)
22             %MD;
23         st[rson].mul = st[rson].mul*st[idx].mul%MD;
24         st[rson].add = (st[rson].add*st[idx].mul+st[idx].add)
25             %MD;
26     }
27     st[idx].mul = 1, st[idx].add = 0;
28 }
29 void buildst(int l, int r, int idx){
30     st[idx].mul = 1, st[idx].add = 0;
31     if(l == r){
32         st[idx].data = arr[l];
33         return;
34     }
35     int mid = l+(r-l)/2;
36     buildst(l, mid, idx*2);
37     buildst(mid+1, r, idx*2+1);
38     up(l, r, idx);
39 }
40 void add(int l, int r, int idx, int L, int R, int v){ //操作L
41     ,R
42     if(r < L || R < l) return;
43     if(L <= l && r <= R){
44         st[idx].add = (st[idx].add+v)%MD;
45         return;
46     }
47     down(l, r, idx);
48     int mid = l+(r-l)/2;
49     add(l, mid, idx*2, L, R, v);
50     add(mid+1, r, idx*2+1, L, R, v);
51     up(l, r, idx);
52 }
53 void mul(int l, int r, int idx, int L, int R, int v){
54     if(r < L || R < l) return;
55     if(L <= l && r <= R){
56         st[idx].add = st[idx].add*v%MD;
57         st[idx].mul = st[idx].mul*v%MD;
58         return;
59     }
60     down(l, r, idx);

```

```

61     int mid = l+(r-l)/2;
62     mul(l, mid, idx*2, L, R, v);
63     mul(mid+1, r, idx*2+1, L, R, v);
64     up(l, r, idx);
65 }
66
67 ll query(int l, int r, int idx, int L, int R){
68     if(r < L || R < l) return 0;
69     if(L <= l && r <= R){
70         return getval(l, r, idx);
71     }
72     down(l, r, idx);
73     int mid = l+(r-l)/2;
74     return (query(l, mid, idx*2, L, R)+query(mid+1, r, idx
75         *2+1, L, R))%MD;

```

## 1.8 BIT

```

1 #define lowbit(x) x&-x
2
3 int arr[N]; //紀錄前綴和
4 int bit[N];
5
6 void conv(int a[], int n) //離散化
7 {
8     vector<int> tmp;
9     for(int i=1; i<=n; i++) tmp.push_back(a[i]);
10    sort(tmp.begin(), tmp.end());
11    for(int i=1; i<=n; i++) a[i] = lower_bound(tmp.begin(),
12        tmp.end(), a[i]) - tmp.begin() + 1;
13 }
14 void buildbit() //每個bit[x]紀錄[x-lowbit(x)+1, x]的總和
15 {
16     for(int i = 0; i < n; i++) bit[i] = arr[i]-arr[i-lowbit(i
17         )];
18 }
19 int sum(int x) //查詢[1,x]的總和
20 {
21     int rtn = 0;
22     for(;x>=lowbit(x);) rtn += bit[x];
23     return rtn;
24 }
25
26 void modify(int x, int d) //把位置x的東西加上d
27 {
28     for(;x<=n;x+=lowbit(x)) bit[x] += d;
29 }

```

## 2 Flow

### 2.1 dinic

```

1 template<typename T>
2 struct DINIC{

```

```

3     static const int MAXN=105;
4     static const T INF=INT_MAX;
5     int n, level[MAXN], cur[MAXN];
6     struct edge{
7         int v,pre;
8         T cap,flow,r;
9         edge(int v,int pre,T cap):v(v),pre(pre),cap(cap),flow(0),
10             r(cap){}
11     };
12     int g[MAXN];
13     vector<edge> e;
14     void init(int _n){
15         memset(g,-1,sizeof(int)*((n=_n)+1));
16         e.clear();
17     }
18     void add_edge(int u,int v,T cap,bool directed=false){
19         e.push_back(edge(v,g[u],cap));
20         g[u]=e.size()-1;
21         e.push_back(edge(u,g[v],directed?0:cap));
22         g[v]=e.size()-1;
23     }
24     int bfs(int s,int t){
25         memset(level,0,sizeof(int)*(n+1));
26         memcpy(cur,g,sizeof(int)*(n+1));
27         queue<int> q;
28         q.push(s);
29         level[s]=1;
30         while(q.size()){
31             int u=q.front();q.pop();
32             for(int i=g[u];~i;i=e[i].pre){
33                 if(!level[e[i].v]&&e[i].r){
34                     level[e[i].v]=level[u]+1;
35                     q.push(e[i].v);
36                     if(e[i].v==t)return 1;
37                 }
38             }
39         }
40         return 0;
41     }
42     T dfs(int u,int t,T cur_flow=INF){
43         if(u==t)return cur_flow;
44         T df;
45         for(int &i=cur[u];~i;i=e[i].pre){
46             if(level[e[i].v]==level[u]+1&&e[i].r){
47                 if(df=dfs(e[i].v,t,min(cur_flow,e[i].r))){
48                     e[i].flow+=df;
49                     e[i^1].flow-=df;
50                     e[i].r-=df;
51                     e[i^1].r+=df;
52                     return df;
53                 }
54             }
55         }
56         return level[u]=0;
57     }
58     T dinic(int s,int t,bool clean=true){
59         if(clean){
60             for(size_t i=0;i<e.size();++i){
61                 e[i].flow=0;
62                 e[i].r=e[i].cap;
63             }
64         }
65         T ans=0, mf=0;
66         while(bfs(s,t))while(mf=dfs(s,t))ans+=mf;
67         return ans;

```

68};

## 2.2 MaxDensitySubgraph

```

1 #include<stdio.h>
2 #include<string.h>
3 const int N=1500;
4 const double inf=0x3fffffff;
5 const double eps=1e-8;
6 int gap[N],dis[N],start,end,ans,sum,head[N],num,dep[N],n,m;
7 bool vis[N];
8 struct edge
9 {
10     int st,ed,next;
11     double flow;
12 }e[80*N];
13 struct node
14 {
15     int x,y;
16 }P[1100];
17 void addedge(int x,int y,double w)
18 {
19     e[num].st=x;e[num].ed=y;e[num].flow=w;e[num].next=head[x];
20     head[x]=num++;
21     e[num].st=y;e[num].ed=x;e[num].flow=0;e[num].next=head[y];
22     head[y]=num++;
23 }
24 void makemap(double g)
25 {
26     int i;
27     memset(head,-1,sizeof(head));
28     num=0;
29     for(i=1;i<=n;i++)
30         addedge(i,end,g);
31     for(i=0;i<m;i++)
32     {
33         addedge(n+i+1,P[i].y,inf);
34         addedge(n+i+1,P[i].x,inf);
35         addedge(start,n+i+1,1.0);
36     }
37 }
38 double dfs(int u,double minflow)
39 {
40     if(u==end)return minflow;
41     int i,v;
42     double f,flow=0.0;
43     for(i=head[u];i!=-1;i=e[i].next)
44     {
45         v=e[i].ed;
46         if(e[i].flow>0)
47         {
48             if(dis[v]+1==dis[u])
49             {
50                 f=dfs(v,e[i].flow>minflow-flow?minflow-flow:e[i].flow);
51                 flow+=f;
52                 e[i].flow-=f;
53                 e[i^1].flow+=f;
54                 if(minflow-flow<=1e-8)return flow;
55                 if(dis[start]>=ans)return flow;
56             }
57         }
58     }
59     return flow;
60 }

```

```

57     if(--gap[dis[u]]==0)
58         dis[start]=ans;
59     dis[u]++;
60     gap[dis[u]]++;
61     return flow;
62 }
63 double isap()
64 {
65     double maxflow=0.0;
66     memset(gap,0,sizeof(gap));
67     memset(dis,0,sizeof(dis));
68     gap[0]=ans;
69     while(dis[start]<ans)
70         maxflow+=dfs(start,inf);
71     return 1.0*m-maxflow;
72 }
73 void dfs1(int u)
74 {
75     vis[u]=true;
76     if(u>=1&&u<=n)
77         sum++;
78     for(int i=head[u];i!=-1;i=e[i].next)
79     {
80         int v=e[i].ed;
81         if(vis[v]==false&&e[i].flow>0)
82             dfs1(v);
83     }
84 }
85 int main()
86 {
87     int i;
88     double Left,Right,mid,flow;
89     while(scanf("%d%d",&n,&m)!=-1)
90     {
91         if(m==0){printf("1\n1\n");continue;}
92         start=0,end=n+m+1,ans=end+1;
93         for(i=0;i<m;i++)
94         {
95             scanf("%d%d",&P[i].x,&P[i].y);
96         }
97         Left=0;Right=m;
98         while(Right-Left>=1.0/n/n)//胡伯涛的论文给出了证明,不同解
99             之间误差的精度不超过1/(n*n)
100         {
101             mid=(Left+Right)/2;
102             makemap(mid);
103             flow=isap();//求出最大权值闭回路
104             if(flow<eps)//如果小于0, g值太大
105                 Right=mid;
106             else Left=mid;
107         }
108         makemap(Left);//最大密度建图
109         isap();
110         memset(vis,false,sizeof(vis));
111         sum=0;
112         dfs1(start);
113         printf("%d\n",sum);
114         for(i=1;i<=n;i++)
115             if(vis[i]==true)//残留网络中源点能到达的点
116                 printf("%d\n",i);
117     }
118     return 0;

```

## 2.3 MinCostMaxFlow

```

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),cost(
9             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=false)
21     {
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?r,-cost));
26         g[v]=e.size()-1;
27     }
28     TP augment(int u,TP CF){
29         if(u==T||CF)return ans+=PIS*CF,CF;
30         vis[u]=1;
31         TP r=CF,d;
32         for(int i=g[u];~i;i=e[i].pre){
33             if(e[i].r&&!e[i].cost&&vis[e[i].v]){
34                 d=augment(e[i].v,min(r,e[i].r));
35                 e[i].r-=d;
36                 e[i^1].r+=d;
37                 if(!(r-=d))break;
38             }
39         }
40         return CF-r;
41     }
42     bool modlabel(){
43         for(int u=0;u<=n;++u)dis[u]=INF;
44         static deque<int>q;
45         dis[T]=0,q.push_back(T);
46         while(q.size()){
47             int u=q.front();q.pop_front();
48             TP dt;
49             for(int i=g[u];~i;i=e[i].pre){
50                 if(e[i^1].r&&(dt=dis[u]-e[i].cost)<dis[e[i].v]){
51                     if((dis[e[i].v]=dt)<=dis[q.size()-1-q.front()]){
52                         q.push_front(e[i].v);
53                     }else q.push_back(e[i].v);
54                 }
55             }
56         }
57         for(int u=0;u<=n;++u)
58             for(int i=g[u];~i;i=e[i].pre)
59                 e[i].cost+=dis[e[i].v]-dis[u];
60         return PIS+=dis[S], dis[S]<INF;
61     }
62     TP mincost(int s,int t){
63         S=s,T=t;

```

```

62 PIS=ans=0;
63 while(modLabel()){
64     do memset(vis,0,sizeof(bool)*(n+1));
65     while(augment(S,INF));
66 }return ans;
67 }
68 };

```

## 3 Geometry

### 3.1 point

```

1 const double eps = 5e-8;
2 struct Point{
3     double x,y;
4     Point(){}
5     Point(double x,double y):x(x),y(y){}
6     Point operator+(Point b)const{
7         return Point(x+b.x,y+b.y);
8     }
9     Point operator-(Point b)const{
10        return Point(x-b.x,y-b.y);
11    }
12    Point operator*(double b)const{
13        return Point(x*b,y*b);
14    }
15    Point operator/(double b)const{
16        return Point(x/b,y/b);
17    }
18    bool operator==(Point b)const{
19        return (fabs(x-b.x)<=eps&&fabs(y-b.y)<=eps);
20    }
21    double dot(Point b)const{
22        return x*b.x+y*b.y;
23    }
24    double cross(Point b)const{
25        return x*b.y-y*b.x;
26    }
27    Point normal()const{
28        return Point(-y,x);
29    } // 求法向量
30    double abs2()const{
31        return dot(*this);
32    } // 向量長度的平方
33    double rad(const Point b)const{
34        return fabs(atan2(fabs(cross(b)),dot(b)));
35    } // 兩向量的弧度
36 };

```

### 3.2 intercircle

```

1 vector<Point> interCir(Point o1, double r1, Point o2, double
2     r2){
3     double d=sqrt((o1-o2).abs2());
4     double c=(r1*r1 + d*d - r2*r2)/2.0/r1/d;
5     double s=sqrt(1.0-c*c);
6     Point v=(o2-o1)*r1/d;

```

```

6 // case 0 intersections
7 if(d>r1+r2||d<fabs(r1-r2)) return{};
8 // case 1 intersection
9 if(d-eps<=r1+r2&&r1+r2<=d+eps) return{o1+v};
10 if(d-eps<=fabs(r1-r2)&&fabs(r1-r2)<=d+eps) return{o1-v};
11 // case 2 intersections
12 Point v_up=(Point){v.x*c-v.y*s,v.x*s+v.y*c};
13 Point v_down=(Point){v.x*c+v.y*s,-v.x*s+v.y*c};
14 return {o1+v_up,o1+v_down};
15 } // 求兩圓交點

```

### 3.3 SegmentGeometry

```

1 double EPS = 1e-10;
2
3 double add(double a, double b){
4     if(fabs(a+b)<EPS*(abs(a)+abs(b)))return 0;
5     else return a+b;
6 }
7
8 struct P//struct for 2d vector/point
9 {
10     double x,y;
11     P(){}
12     P(double x, double y):x(x),y(y){}
13     P operator+(P p){return P(add(x,p.x), add(y,p.y));}
14     P operator-(P p){return P(add(x,-p.x), add(y,-p.y));}
15     P operator*(double d){return P(x*d,y*d);}
16     double dot(P p){return add( x*p.x, y*p.y );}
17     double det(P p){return add( x*p.y, -y*p.x );}
18 };
19
20 //is point q on p1p2
21 bool on_seg(P p1, P p2, P q){return (p1-q).det(p2-q)==0&&(p1-
22     q).dot(p2-q)<=0;}
23
24 P intersection(P p1, P p2, P q1, P q2)//p and q Must not be
25     parallel
26 {return p1 + (p2-p1)*((q2-q1).det(q1-p1)/(q2-q1).det(p2-p1))
27     ;}
28
29 bool par(P p1, P p2, P p3, P p4){return (p2-p1).det(p4-p3)
30     ==0;}
31
32 bool operator<(const P& lhs, const P& rhs)
33 {return (lhs.x==rhs.x)?lhs.y<rhs.y:lhs.x<rhs.x;}
34
35 bool operator==(const P& lhs, const P& rhs)
36 {return lhs.x==rhs.x&&lhs.y==rhs.y;}
37
38 double len(P vec)
39 {return sqrt(add(vec.x*vec.x, vec.y*vec.y));}
40
41 double dis(P p1, P p2)
42 {return len(p2-p1);}
43
44 struct seg
45 {
46     seg(){}
47     seg(P _p1, P _p2)
48     {
49         p[0]=_p1;
50         p[1]=_p2;

```

```

47         if(p[1]<p[0])swap(p[0],p[1]);
48     }
49     P p[2];
50 };
51
52 bool par(seg& lhs, seg& rhs)
53 {return par(lhs.p[0],lhs.p[1],rhs.p[0],rhs.p[1]);}
54
55 P intersection(seg& lhs, seg& rhs)//p and q Must not be
56     parallel
57 {return intersection(lhs.p[0],lhs.p[1],rhs.p[0],rhs.p[1]);}
58
59 bool on_seg(seg& sg, P q)
60 {return on_seg(sg.p[0],sg.p[1],q);}
61
62 bool overlap(seg s1, seg s2){
63     return par(s1,s2)&&
64     ( on_seg(s1,s2.p[0])||on_seg(s1,s2.p[1])||
65     on_seg(s2,s1.p[0])||on_seg(s2,s1.p[1]) );
66 }
67
68 bool is_intersect(seg s1, seg s2){
69     if(par(s1,s2))return false;
70     P p0 = intersection(s1,s2);
71     return on_seg(s1,p0)&&on_seg(s2,p0);
72 }
73
74 //make sure the vec is not vertical
75 double interpolate(seg& vec, double X){
76     double y0=vec.p[0].y,y1=vec.p[1].y,
77     x0=vec.p[0].x,x1=vec.p[1].x;
78     return y0+(y1-y0)*(X-x0)/(x1-x0);
79 }
80
81 //pts in clockwise order, p[N]=p[0]
82 bool in_poly(P* pol,int N,P pt){
83     double X = pt.x,Y=pt.y;
84     int pas=0;
85     for(int i=0;i<N;i++){
86         if(pol[i].x==pol[i+1].x)continue;
87         seg s0(pol[i],pol[i+1]);
88         //up or down?
89         double Y1 = interpolate(s0,X);
90         if(Y1<Y-EPS)continue;
91         double x1=min(pol[i].x,pol[i+1].x),xr=max(pol[i].x,
92             pol[i+1].x);
93         if(x1<X-EPS&&xr>=X-EPS)pas++;
94     }
95     return pas&1;
96 }
97
98 double dpseg(P p, P p1, P p2)//p to p1p2, p1!=p2
99 {
100     P v=p2-p1, v1=p-p1, v2=p-p2;
101     if( v.dot(v1) < EPS )return dis(p,p1);
102     if( v.dot(v2) > EPS )return dis(p,p2);
103     return fabs((p-p1).det(v))/len(v);
104 }
105
106 double dpseg(P p, seg s1){
107     return dpseg(p,s1.p[0],s1.p[1]);
108 }
109
110 double dsegseg(P p1, P p2, P p3, P p4){
111     if( is_intersect( seg(p1,p2), seg(p3,p4) ) )return 0;

```

```

110 return min( min( dpseg(p1,p3,p4),dpseg(p2,p3,p4) ), min(
    dpseg(p3,p1,p2),dpseg(p4,p1,p2) ) );
111 }
112
113 double dsegseg(seg s1, seg s2)
114 {
115     return dsegseg( s1.p[0],s1.p[1],s2.p[0],s2.p[1] );
116 }

```

### 3.4 convexHullTrick

```

1 // usage ( for example )
2 // dp[i] = min(dp[j] - 2*a[i]*a[j] + a[j]^2) + m + a[i]^2
3 // insert into hull :
4 //      a      x      +      b
5 // (-2*a[j]) (a[i]) + (dp[j]+a[j]^2)
6 // get dp[i] :
7 // dp[i] = hull(a[i]) + m + a[i]^2
8
9 template<typename Ty = long long int>
10 class Linear{
11 private:
12     Ty a, b;
13 public:
14     Linear(Ty a, Ty b):a(a), b(b) {}
15     Ty operator()(Ty x) { return a*x+b; }
16     // get x of intersection of two lines (fraction)
17     tuple<Ty, Ty> inter(Linear &that){
18         ll up = that.b-this->b;
19         ll down = this->a-that.a;
20         if(down < 0) up *= -1, down *= -1;
21         return make_tuple(up, down);
22     }
23 };
24
25 template<typename Ty = long long int>
26 class ConvexHull{
27 private:
28     using L = Linear<Ty>;
29     vector<L> hull;
30 public:
31     void push_back(L h){
32         while(hull.size() >= 2){
33             auto &f = hull.end()[-2];
34             auto &g = hull.end()[-1]; // back
35             // x of inter(h,f) <= x of inter(f,g)
36             Ty a, b, c, d;
37             tie(a, b) = h.inter(f);
38             tie(c, d) = f.inter(g);
39             if(a*d <= b*c) hull.pop_back();
40             else break;
41         }
42         hull.push_back(h);
43     }
44     Ty operator() (Ty x){
45         static int idx = 0;
46         if(idx >= hull.size())
47             idx = max(0, (int)hull.size()-2);
48         while(idx+1 < hull.size())
49         {
50             if(hull[idx+1](x) <= hull[idx](x)) idx++;
51             else break;
52         }

```

```

53     return hull[idx](x);
54 }
55 };

```

### 3.5 Geometry

```

1 const double PI=atan2(0.0,-1.0);
2 template<typename T>
3 struct point{
4     T x,y;
5     point(){}
6     point(const T&x,const T&y):x(x),y(y){}
7     point operator+(const point &b)const{
8         return point(x+b.x,y+b.y); }
9     point operator-(const point &b)const{
10        return point(x-b.x,y-b.y); }
11     point operator*(const T &b)const{
12        return point(x*b,y*b); }
13     point operator/(const T &b)const{
14        return point(x/b,y/b); }
15     bool operator==(const point &b)const{
16        return x==b.x&&y==b.y; }
17     T dot(const point &b)const{
18        return x*b.x+y*b.y; }
19     T cross(const point &b)const{
20        return x*b.y-y*b.x; }
21     point normal()const{//求法向量
22        return point(-y,x); }
23     T abs2()const{//向量長度的平方
24        return dot(*this); }
25     T rad(const point &b)const{//兩向量的弧度
26     return fabs(atan2(fabs(cross(b)),dot(b))); }
27     T getA()const{//對x軸的弧度
28         T A=atan2(y,x);//超過180度會變負的
29         if(A<=-PI/2)A+=PI*2;
30         return A;
31     }
32 };
33 template<typename T>
34 struct line{
35     line(){}
36     point<T> p1,p2;
37     T a,b,c;//ax+by+c=0
38     line(const point<T>&x,const point<T>&y):p1(x),p2(y){}
39     void pton()const{//轉成一般式
40         a=p1.y-p2.y;
41         b=p2.x-p1.x;
42         c=-a*p1.x-b*p1.y;
43     }
44     T ori(const point<T> &p)const{//點和有向直線的關係, >0左
45         //邊、=0在線上<0右邊
46         return (p2-p1).cross(p-p1);
47     }
48     T btw(const point<T> &p)const{//點投影落在線段上<=0
49         return (p1-p).dot(p2-p);
50     }
51     bool point_on_segment(const point<T>&p)const{//點是否在線段
52         //上
53         return ori(p)==0&&btw(p)<=0;

```

```

53     T dis2(const point<T> &p,bool is_segment=0)const{//點跟直線
54         //線段的距離平方
55         point<T> v=p2-p1,v1=p-p1;
56         if(is_segment){
57             point<T> v2=p-p2;
58             if(v.dot(v1)<=0)return v1.abs2();
59             if(v.dot(v2)>=0)return v2.abs2();
60         }
61         T tmp=v.cross(v1);
62         return tmp*tmp/v.abs2();
63     }
64     T seg_dis2(const line<T> &l)const{//兩線段距離平方
65         return min({dis2(l.p1,1),dis2(l.p2,1),l.dis2(p1,1),l.dis2
66             (p2,1)});
67     }
68     point<T> projection(const point<T> &p)const{//點對直線的投
69         //影
70         point<T> n=(p2-p1).normal();
71         return p-n*(p-p1).dot(n)/n.abs2();
72     }
73     point<T> mirror(const point<T> &p)const{
74         //點對直線的鏡射, 要先呼叫pton轉成一般式
75         point<T> R;
76         T d=a*a+b*b;
77         R.x=(b*b*p.x-a*a*p.x-2*a*b*p.y-2*a*c)/d;
78         R.y=(a*a*p.y-b*b*p.y-2*a*b*p.x-2*b*c)/d;
79         return R;
80     }
81     bool equal(const line &l)const{//直線相等
82         return ori(l.p1)==0&&ori(l.p2)==0;
83     }
84     bool parallel(const line &l)const{
85         return (p1-p2).cross(l.p1-l.p2)==0;
86     }
87     bool cross_seg(const line &l)const{
88         return (p2-p1).cross(l.p1-p1)*(p2-p1).cross(l.p2-p1)<=0;
89         //直線是否交線段
90     }
91     int line_intersect(const line &l)const{//直線相交情況, -1無
92         //限多點、1交於一點、0不相交
93         return parallel(l)?(ori(l.p1)==0?-1:0):1;
94     }
95     int seg_intersect(const line &l)const{
96         T c1=ori(l.p1), c2=ori(l.p2);
97         T c3=l.ori(p1), c4=l.ori(p2);
98         if(c1==0&&c2==0){//共線
99             bool b1=btw(l.p1)>=0,b2=btw(l.p2)>=0;
100             T a3=l.btw(p1),a4=l.btw(p2);
101             if(b1&&b2&&a3==0&&a4>=0) return 2;
102             if(b1&&b2&&a3>=0&&a4==0) return 3;
103             if(b1&&b2&&a3>=0&&a4>=0) return 0;
104             return -1;//無限交點
105         }else if(c1*c2<=0&&c3*c4<=0)return 1;
106         return 0;//不相交
107     }
108     point<T> line_intersection(const line &l)const{/*直線交點*/
109         point<T> a=p2-p1,b=l.p2-l.p1,s=l.p1-p1;
110         //if(a.cross(b)==0)return INF;
111         return p1+a*(s.cross(b)/a.cross(b));
112     }
113     point<T> seg_intersection(const line &l)const{//線段交點
114         int res=seg_intersect(l);
115         if(res<=0) assert(0);

```



```

111 if(res==2) return p1;
112 if(res==3) return p2;
113 return line_intersection(l);
114 }
115 };
116 template<typename T>
117 struct polygon{
118     polygon(){}
119     vector<point<T> > p; //逆時針順序
120     T area()const{//面積
121         T ans=0;
122         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++)
123             ans+=p[i].cross(p[j]);
124         return ans/2;
125     }
126     point<T> center_of_mass()const{//重心
127         T cx=0,cy=0,w=0;
128         for(int i=p.size()-1,j=0;j<(int)p.size();i=j++){
129             T a=p[i].cross(p[j]);
130             cx+=(p[i].x+p[j].x)*a;
131             cy+=(p[i].y+p[j].y)*a;
132             w+=a;
133         }
134         return point<T>(cx/3/w,cy/3/w);
135     }
136     char ahas(const point<T>& t)const{//點是否在簡單多邊形內，
137         是的話回傳1、在邊上回傳-1、否則回傳0
138         bool c=0;
139         for(int i=0,j=p.size()-1;i<p.size();j=i++){
140             if((line<T>(p[i],p[j]).point_on_segment(t))return -1;
141             else if((p[i].y>t.y)!=p[j].y>t.y)&&
142                 t.x<(p[j].x-p[i].x)*(t.y-p[i].y)/(p[j].y-p[i].y)+p[i].x)
143                 c=!c;
144             return c;
145         }
146         char point_in_convex(const point<T>&x)const{
147             int l=1,r=(int)p.size()-2;
148             while(l<r){//點是否在凸多邊形內，是的話回傳1、在邊上回傳
149                 -1、否則回傳0
150                 int mid=(l+r)/2;
151                 T a1=(p[mid]-p[0]).cross(x-p[0]);
152                 T a2=(p[mid+1]-p[0]).cross(x-p[0]);
153                 if(a1>=0&&a2<=0){
154                     T res=(p[mid+1]-p[mid]).cross(x-p[mid]);
155                     return res>0?1:(res>=0?-1:0);
156                 }else if(a1<0)r=mid-1;
157                 else l=mid+1;
158             }
159             return 0;
160         }
161         vector<T> getA()const{//凸包邊對x軸的夾角
162             vector<T>res; //一定是遞增的
163             for(size_t i=0;i<p.size();i++){
164                 res.push_back((p[(i+1)%p.size()]-p[i]).getA());
165             }
166             bool line_intersect(const vector<T>&A,const line<T> &l)
167                 const{//O(logN)
168                 int f1=upper_bound(A.begin(),A.end(),(l.p1-l.p2).getA())-
169                     A.begin();
170                 int f2=upper_bound(A.begin(),A.end(),(l.p2-l.p1).getA())-
171                     A.begin();
172                 return l.cross_seg(line<T>(p[f1],p[f2]));
173             }
174         }
175     }
176     polygon cut(const line<T> &l)const{//凸包對直線切割，得到直
177         線l左側的凸包
178         polygon ans;
179         for(int n=p.size(),i=n-1,j=0;j<n;i=j++){
180             if(l.ori(p[i])>=0){
181                 ans.p.push_back(p[i]);
182                 if(l.ori(p[j])<0)
183                     ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
184             }else if(l.ori(p[j])>0)
185                 ans.p.push_back(l.line_intersection(line<T>(p[i],p[j])));
186             }
187         return ans;
188     }
189     static bool graham_cmp(const point<T>& a,const point<T>& b){
190         //凸包排序函數
191         return (a.x<b.x)||a.x==b.x&&a.y<b.y;
192     }
193     void graham(vector<point<T> > &s){//凸包
194         sort(s.begin(),s.end(),graham_cmp);
195         p.resize(s.size()+1);
196         int m=0;
197         for(size_t i=0;i<s.size();i++){
198             while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
199             p[m++]=s[i];
200         }
201         for(int i=s.size()-2,t=m+1;i>=0;--i){
202             while(m>=2&&(p[m-1]-p[m-2]).cross(s[i]-p[m-2])<=0)--m;
203             p[m++]=s[i];
204         }
205         if(s.size()>1)--m;
206         p.resize(m);
207     }
208     T diam()const{//直徑
209         int n=p.size(),t=1;
210         T ans=0;p.push_back(p[0]);
211         for(int i=0;i<n;i++){
212             point<T> now=p[i+1]-p[i];
213             while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=t+1;
214             ans=max(ans,(p[i]-p[t]).abs2());
215         }
216         return p.pop_back(),ans;
217     }
218     T min_cover_rectangle()const{//最小覆蓋矩形
219         int n=p.size(),t=1,r=1,l;
220         if(n<3)return 0;//也可以做最小周長矩形
221         T ans=1e99;p.push_back(p[0]);
222         for(int i=0;i<n;i++){
223             point<T> now=p[i+1]-p[i];
224             while(now.cross(p[t+1]-p[i])>now.cross(p[t]-p[i]))t=t+1;
225             while(now.dot(p[r+1]-p[i])>now.dot(p[r]-p[i]))r=(r+1)%n;
226             if(!i)l=r;
227             while(now.dot(p[l+1]-p[i])<=now.dot(p[l]-p[i]))l=(l+1)%n;
228             T d=now.abs2();
229             T tmp=now.cross(p[t]-p[i])*(now.dot(p[r]-p[i])-now.dot(p[l]-p[i]))/d;
230             ans=min(ans,tmp);
231         }
232     }
233     return p.pop_back(),ans;
234 }
235 T dis2(polygon &pl){//凸包最近距離平方
236     vector<point<T> > &P=p,&Q=pl.p;
237     int n=P.size(),m=Q.size(),l=0,r=0;
238     for(int i=0;i<n;++i)if(P[i].y<P[l].y)l=i;
239     for(int i=0;i<m;++i)if(Q[i].y<Q[r].y)r=i;
240     P.push_back(P[0]),Q.push_back(Q[0]);
241     T ans=1e99;
242     for(int i=0;i<n;++i){
243         while((P[l]-P[l+1]).cross(Q[r+1]-Q[r])<0)r=(r+1)%m;
244         ans=min(ans,line<T>(P[l],P[l+1]).seg_dis2(line<T>(Q[r],Q[r+1])));
245         l=(l+1)%n;
246     }
247     return P.pop_back(),Q.pop_back(),ans;
248 }
249 static char sign(const point<T>&t){
250     return (t.y==0?t.x:t.y)<0;
251 }
252 static bool angle_cmp(const line<T>& A,const line<T>& B){
253     point<T> a=A.p2-A.p1,b=B.p2-B.p1;
254     return sign(a)<sign(b)||sign(a)==sign(b)&&a.cross(b)>0;
255 }
256 int halfplane_intersection(vector<line<T> > &s){//半平面交
257     sort(s.begin(),s.end(),angle_cmp); //線段左側為該線段半平面
258     int L,R,n=s.size();
259     vector<point<T> > px(n);
260     vector<line<T> > q(n);
261     q[L=R=0]=s[0];
262     for(int i=1;i<n;++i){
263         while(L<R&&s[i].ori(px[R-1])<=0)--R;
264         while(L<R&&s[i].ori(px[L])<=0)+L;
265         q[++R]=s[i];
266         if(q[R].parallel(q[R-1])){
267             --R;
268             if(q[R].ori(s[i].p1)>0)q[R]=s[i];
269             if(L<R)px[R-1]=q[R-1].line_intersection(q[R]);
270         }
271         while(L<R&&q[L].ori(px[R-1])<=0)--R;
272         p.clear();
273         if(R-L<=1)return 0;
274         px[R]=q[R].line_intersection(q[L]);
275         for(int i=L;i<=R;++i)p.push_back(px[i]);
276         return R-L+1;
277     }
278 }
279 template<typename T>
280 struct triangle{
281     point<T> a,b,c;
282     triangle(){}
283     triangle(const point<T> &a,const point<T> &b,const point<T> &c):a(a),b(b),c(c){}
284     T area()const{
285         T t=(b-a).cross(c-a)/2;
286         return t>0?t:-t;
287     }
288     point<T> barycenter()const{//重心
289         return (a+b+c)/3;
290     }
291     point<T> circumcenter()const{//外心
292         static line<T> u,v;
293         u.p1=(a+b)/2;

```

```

286 u.p2=point<T>(u.p1.x-a.y+b.y,u.p1.y+a.x-b.x);
287 v.p1=(a+c)/2;
288 v.p2=point<T>(v.p1.x-a.y+c.y,v.p1.y+a.x-c.x);
289 return u.line_intersection(v);
290 }
291 point<T> incenter()const{//內心
292 T A=sqrt((b-c).abs2()),B=sqrt((a-c).abs2()),C=sqrt((a-b).abs2());
293 return point<T>(A*a.x+B*b.x+C*c.x,A*a.y+B*b.y+C*c.y)/(A+B+C);
294 }
295 point<T> perpencenter()const{//垂心
296 return barycenter()*3-circumcenter()*2;
297 }
298 };
299 template<typename T>
300 struct point3D{
301 T x,y,z;
302 point3D(){}
303 point3D(const T&x,const T&y,const T&z):x(x),y(y),z(z){}
304 point3D operator+(const point3D &b)const{
305 return point3D(x+b.x,y+b.y,z+b.z);}
306 point3D operator-(const point3D &b)const{
307 return point3D(x-b.x,y-b.y,z-b.z);}
308 point3D operator*(const T &b)const{
309 return point3D(x*b.y*y*b.z,*b);}
310 point3D operator/(const T &b)const{
311 return point3D(x/b.y/b.z/b);}
312 bool operator==(const point3D &b)const{
313 return x==b.x&&y==b.y&&z==b.z;}
314 T dot(const point3D &b)const{
315 return x*b.x+y*b.y+z*b.z;}
316 point3D cross(const point3D &b)const{
317 return point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
318 T abs2()const{//向量長度的平方
319 return dot(*this);}
320 T area2(const point3D &b)const{//和b、原點圍成面積的平方
321 return cross(b).abs2()/4;}
322 };
323 template<typename T>
324 struct line3D{
325 point3D<T> p1,p2;
326 line3D(){}
327 line3D(const point3D<T> &p1,const point3D<T> &p2):p1(p1),p2(p2){}
328 T dis2(const point3D<T> &p,bool is_segment=0)const{//點跟直線/線段的距離平方
329 point3D<T> v=p2-p1,v1=p-p1;
330 if(is_segment){
331 point3D<T> v2=p-p2;
332 if(v.dot(v1)<=0)return v1.abs2();
333 if(v.dot(v2)>=0)return v2.abs2();
334 }
335 point3D<T> tmp=v.cross(v1);
336 return tmp.abs2()/v.abs2();
337 }
338 pair<point3D<T>,point3D<T>> closest_pair(const line3D<T> &l)const{
339 point3D<T> v1=(p1-p2),v2=(l.p1-l.p2);
340 point3D<T> N=v1.cross(v2),ab(p1-l.p1);
341 //if(N.abs2()==0)return NULL;平行或重合
342 T tmp=N.dot(ab),ans=tmp*tmp/N.abs2();{//最近點對距離
343 point3D<T> d1=p2-p1,d2=l.p2-l.p1,D=d1.cross(d2),G=l.p1-p1;
344 T t1=(G.cross(d2)).dot(D)/D.abs2();
345 T t2=(G.cross(d1)).dot(D)/D.abs2();
346 return make_pair(p1+d1*t1,l.p1+d2*t2);
347 }
348 bool same_side(const point3D<T> &a,const point3D<T> &b)const{
349 return (p2-p1).cross(a-p1).dot((p2-p1).cross(b-p1))>0;
350 }
351 };
352 template<typename T>
353 struct plane{
354 point3D<T> p0,n;//平面上的點和法向量
355 plane(){}
356 plane(const point3D<T> &p0,const point3D<T> &n):p0(p0),n(n){}
357 T dis2(const point3D<T> &p)const{//點到平面距離的平方
358 T tmp=(p-p0).dot(n);
359 return tmp*tmp/n.abs2();
360 }
361 point3D<T> projection(const point3D<T> &p)const{
362 return p-n*(p-p0).dot(n)/n.abs2();
363 }
364 point3D<T> line_intersection(const line3D<T> &l)const{
365 T tmp=n.dot(l.p2-l.p1);//等於0表示平行或重合該平面
366 return l.p1+(l.p2-l.p1)*(n.dot(p0-l.p1)/tmp);
367 }
368 line3D<T> plane_intersection(const plane &pl)const{
369 point3D<T> e=n.cross(pl.n),v=n.cross(e);
370 T tmp=pl.n.dot(v);//等於0表示平行或重合該平面
371 point3D<T> q=p0+(v*(pl.n.dot(pl.p0-p0))/tmp);
372 return line3D<T>(q,q+e);
373 }
374 };
375 template<typename T>
376 struct triangle3D{
377 point3D<T> a,b,c;
378 triangle3D(){}
379 triangle3D(const point3D<T> &a,const point3D<T> &b,const point3D<T> &c):a(a),b(b),c(c){}
380 bool point_in(const point3D<T> &p)const{//點在該平面上的投影在三角形中
381 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);
382 }
383 };
384 template<typename T>
385 struct tetrahedron{//四面體
386 point3D<T> a,b,c,d;
387 tetrahedron(){}
388 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){}
389 T volume6()const{//體積的六倍
390 return (d-a).dot((b-a).cross(c-a));
391 }
392 point3D<T> centroid()const{
393 return (a+b+c+d)/4;
394 }
395 bool point_in(const point3D<T> &p)const{
396 return triangle3D<T>(a,b,c).point_in(p)&&triangle3D<T>(c,d,a).point_in(p);
397 }
398 };
399 template<typename T>
400 struct convexhull3D{
401 static const int MAXN=1005;
402 struct face{
403 int a,b,c;
404 face(int a,int b,int c):a(a),b(b),c(c){}
405 };
406 vector<point3D<T>> pt;
407 vector<face> ans;
408 int fid[MAXN][MAXN];
409 void build(){
410 int n=pt.size();
411 ans.clear();
412 memset(fid,0,sizeof(fid));
413 ans.emplace_back(0,1,2);//注意不能共線
414 ans.emplace_back(2,1,0);
415 int ftop = 0;
416 for(int i=3, ftop=1; i<n; ++i,++ftop){
417 vector<face> next;
418 for(auto &f:ans){
419 T d=(pt[i]-pt[f.a]).dot((pt[f.b]-pt[f.a]).cross(pt[f.c]-pt[f.a]));
420 if(d==0) next.push_back(f);
421 int ff=0;
422 if(d>0) ff=ftop;
423 else if(d<0) ff=-ftop;
424 fid[f.a][f.b]=fid[f.b][f.c]=fid[f.c][f.a]=ff;
425 }
426 for(auto &f:ans){
427 if(fid[f.a][f.b]>0 && fid[f.a][f.b]!=fid[f.b][f.a])
428 next.emplace_back(f.a,f.b,i);
429 if(fid[f.b][f.c]>0 && fid[f.b][f.c]!=fid[f.c][f.b])
430 next.emplace_back(f.b,f.c,i);
431 if(fid[f.c][f.a]>0 && fid[f.c][f.a]!=fid[f.a][f.c])
432 next.emplace_back(f.c,f.a,i);
433 }
434 ans=next;
435 }
436 }
437 point3D<T> centroid()const{
438 point3D<T> res(0,0,0);
439 T vol=0;
440 for(auto &f:ans){
441 T tmp=pt[f.a].dot(pt[f.b].cross(pt[f.c]));
442 res=res+(pt[f.a]+pt[f.b]+pt[f.c])*tmp;
443 vol+=tmp;
444 }
445 return res/(vol*4);
446 }
447 };

```

### 3.6 nearestDist

```

1 bool cmp_y(P a, P b){
2 return a.y < b.y;
3 }
4
5 bool cmp_x(P a, P b){
6 return a.x < b.x;
7 }
8
9 double dc(P *arr, int n){
10 if(n == 1) return INF;
11 int mid = n/2;

```



```

12 double cx = arr[mid].x;
13 double dist = min( dc(arr, mid), dc(arr+mid, n-mid) );
14 inplace_merge(arr, arr+mid, arr+n, cmp_y);
15 static vector<P> brr; brr.clear();
16 for(int i = 0; i < n; i++){
17     if(fabs(arr[i].x)-cx >= dist) continue;
18     for(int j = brr.size()-1; j >= 0; j--){
19         double dx = brr[j].x-arr[i].x;
20         double dy = brr[j].y-arr[i].y;
21         if(fabs(dy) >= dist) break;
22         dist = min(dist, sqrt(dx*dx+dy*dy));
23     }
24     brr.push_back(arr[i]);
25 }
26 return dist;
27 }
28
29 double nearestDist(P *arr, int n){
30     sort(arr, arr+n, cmp_x);
31     return dc(arr, n);
32 }

```

### 3.7 convexHull

```

1 struct ConvexHull {
2     vector<Point> h; // hull
3     static bool cmp(const Point &lhs, const Point &rhs) {
4         if(lhs.x == rhs.x) return lhs.y < rhs.y;
5         else return lhs.x < rhs.x;
6     }
7     // p : points, h : return hull
8     void buildHull(vector<Point> &p) {
9         int n = p.size(), m = 0;
10        sort(p.begin(), p.end(), cmp);
11        h.resize(n+1);
12        for(int i = 0; i < n; ++i){
13            while(m>=2&&(h[m-1]-h[m-2]).cross(p[i]-h[m-2])<=0)--m;
14            h[m++]=p[i];
15        }
16        for(int i = n-2, t = m+1; i >= 0; --i) {
17            while(m>=t&&(h[m-1]-h[m-2]).cross(p[i]-h[m-2])<=0)--m;
18            h[m++]=p[i];
19        }
20        if(h.size(>1)--m;
21        h.resize(m);
22    }
23 };

```

## 4 Graph

### 4.1 SCC

```

1 /*
2  * SCC in an directed graph
3  * usage : init(), addEdge(), run()
4  * 0-base graph
5  */
6 class DirectedTarjan

```

```

7 {
8 private:
9     int vn;
10    int ts; // dfs timestamp
11    int sccIdx;
12    vector<vector<int>> graph;
13    vector<int> low;
14    vector<int> dep;
15    stack<int> stk;
16    vector<bool> inStk;
17    vector<int> scc; // scc[v] = id of scc
18    void reportSCC(int v){
19        int x;
20        do{
21            x = stk.top(); stk.pop();
22            inStk[x] = false;
23            scc[x] = sccIdx;
24        } while(x != v);
25        sccIdx++;
26    }
27    void dfs(int v) {
28        low[v] = dep[v] = ++ts;
29        stk.push(v); inStk[v] = true;
30        for(auto c : graph[v]){
31            if(dep[c] == 0){// not visited
32                dfs(c);
33                low[v] = min(low[v], low[c]);
34            }
35            if(inStk[c]) low[v] = min(low[v], dep[c]);
36        }
37        if(low[v] == dep[v]) reportSCC(v);
38    }
39 public:
40    void init(int v){
41        vn = v, ts = 0, sccIdx = 0;
42        graph.resize(v);
43        low.resize(v, 0);
44        dep.resize(v, 0);
45        scc.resize(v, 0);
46        inStk.resize(v, false);
47    }
48    void addEdge(int u, int v){
49        graph[u].emplace_back(v);
50    }
51    void run(){
52        for(int v = 0; v < vn; v++)
53            if(dep[v] == 0) dfs(v);
54    }
55    int getSCCId(int v) { return scc[v]; }
56 };

```

### 4.2 lca

```

1 const int MAXN=100000; // 1-base
2 const int MLG=17; //log2(MAXN)+1;
3 int pa[MLG+2][MAXN+5];
4 int dep[MAXN+5];
5 vector<int> G[MAXN+5];
6 void dfs(int x, int p=0){//dfs(root);
7     pa[0][x]=p;
8     for(int i=0; i<=MLG; ++i)
9         pa[i+1][x]=pa[i][pa[i][x]];
10    for(auto &i:G[x]){

```

```

11        if(i==p)continue;
12        dep[i]=dep[x]+1;
13        dfs(i,x);
14    }
15 }
16 inline int jump(int x, int d){
17     for(int i=0; i<=MLG; ++i)
18         if((d>>i)&1) x=pa[i][x];
19     return x;
20 }
21 inline int find_lca(int a, int b){
22     if(dep[a]>dep[b])swap(a,b);
23     b=jump(b, dep[b]-dep[a]);
24     if(a==b)return a;
25     for(int i=MLG; i>=0; --i){
26         if(pa[i][a]!=pa[i][b]){
27             a=pa[i][a];
28             b=pa[i][b];
29         }
30     }
31     return pa[0][a];
32 }

```

### 4.3 bellman\_Ford

```

1 struct edge{ int from, to, cost; };
2 #define INF 2147483647
3
4 edge es[100];
5
6 int d[100]; //min distance
7 int V, E, s, f;
8
9 bool bellman_ford() // return true if there is negative loop
10 {
11     for(int i = 0; i < V; i++) d[i] = INF;
12     d[s] = 0;
13
14     for(int i = 0; i < V; i++)
15     {
16         for(int j = 0; j < E; j++)
17         {
18             edge e = es[j];
19             if(d[e.from] != INF && d[e.to] > d[e.from] + e.cost)
20             {
21                 d[e.to] = d[e.from] + e.cost;
22                 if(i == V - 1) return true; //got neg loop
23             }
24             if(d[e.to] != INF && d[e.from] > d[e.to] + e.cost)
25             {
26                 d[e.from] = d[e.to] + e.cost;
27                 if(i == V - 1) return true; //got neg loop
28             }
29         }
30     }
31     return false;
32 }

```

## 4.4 MaxMatching

```

1 #define FZ(x) memset(x,0,sizeof(x))
2 struct GenMatch // 1-base
3 {
4     static const int MAXN = 250;
5     int V;
6     bool el[MAXN][MAXN];
7     int pr[MAXN];
8     bool inq[MAXN], inp[MAXN], inb[MAXN];
9     queue<int> qe;
10    int st, ed;
11    int nb;
12    int bk[MAXN], djs[MAXN];
13    int ans;
14    void init(int _V){
15        V = _V;
16        FZ(el);
17        FZ(pr);
18        FZ(inq);
19        FZ(inp);
20        FZ(inb);
21        FZ(bk);
22        FZ(djs);
23        ans = 0;
24    }
25    void add_edge(int u, int v){
26        el[u][v] = el[v][u] = 1;
27    }
28    int lca(int u, int v){
29        memset(inp, 0, sizeof(inp));
30        while(1){
31            u = djs[u];
32            inp[u] = true;
33            if(u == st) break;
34            u = bk[pr[u]];
35        }
36        while(1){
37            v = djs[v];
38            if(inp[v]) return v;
39            v = bk[pr[v]];
40        }
41        return v;
42    }
43    void upd(int u){
44        int v;
45        while(djs[u] != nb){
46            v = pr[u];
47            inb[djs[u]] = inb[djs[v]] = true;
48            u = bk[v];
49            if(djs[u] != nb) bk[u] = v;
50        }
51    }
52    void blo(int u, int v){
53        nb = lca(u, v);
54        memset(inb, 0, sizeof(inb));
55        upd(u);
56        upd(v);
57        if(djs[u] != nb) bk[u] = v;
58        if(djs[v] != nb) bk[v] = u;
59        for(int tu = 1; tu <= V; tu++){
60            if(inb[djs[tu]]){
61                djs[tu] = nb;
62                if(!inq[tu]){
63                    qe.push(tu);

```

```

64                inq[tu] = 1;
65            }
66        }
67    }
68    void flow(){
69        memset(inq, false, sizeof(inq));
70        memset(bk, 0, sizeof(bk));
71        for(int i = 1; i <= V; i++){
72            djs[i] = i;
73        }
74        while(qe.size()) qe.pop();
75        qe.push(st);
76        inq[st] = 1;
77        ed = 0;
78        while(qe.size()){
79            int u = qe.front();
80            qe.pop();
81            for(int v = 1; v <= V; v++){
82                if(el[u][v] && (djs[u] != djs[v]) && (pr[u] != v)){
83                    if((v == st) || ((pr[v] > 0) && bk[pr[v]] > 0)){
84                        blo(u, v);
85                    }
86                    else if(bk[v] == 0){
87                        bk[v] = u;
88                        if(pr[v] > 0){
89                            if(!inq[pr[v]]) qe.push(pr[v]);
90                        }
91                        else{
92                            ed = v;
93                            return;
94                        }
95                    }
96                }
97            }
98        }
99        void aug(){
100            int u, v, w;
101            u = ed;
102            while(u > 0){
103                v = bk[u];
104                w = pr[v];
105                pr[v] = u;
106                pr[u] = v;
107                u = w;
108            }
109        }
110        int solve(){
111            memset(pr, 0, sizeof(pr));
112            for(int u = 1; u <= V; u++){
113                if(pr[u] == 0){
114                    st = u;
115                    flow();
116                    if(ed > 0){
117                        aug();
118                        ans ++;
119                    }
120                }
121            }
122            return ans;
123        }
124    } gm;

```

## 4.5 MinimumMeanCycle

```

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        }
21        res = min(res, val);
22    }
23    return res;

```

## 4.6 MaxBiMatching

```

1 //注意：變數V
2 #define MAXV 505
3 int V; // # of vertex
4 vector<int> G[MAXV];
5 int match[MAXV];
6 int used[MAXV];
7
8 void add_edge(int u, int v){
9     G[u].pb(v);
10    G[v].pb(u);
11}
12
13 bool dfs(int u){
14     used[u] = true;
15     for(int i = 0; i < G[u].size(); i++){
16         int v = G[u][i], w = match[v];
17         if(w < 0 || !used[w] && dfs(w)){
18             match[u] = v;
19             match[v] = u;
20             return true;
21         }
22     }
23     return false;
24 }
25
26 int bip_match(){
27     int res = 0;
28     memset(match, -1, sizeof(match));
29     for(int v = 0; v < V; v++){
30         if(match[v] < 0){
31             memset(used, 0, sizeof(used));
32             if(dfs(v)) res ++;
33         }
34     }

```

```

35     return res;
36 }

```

## 4.7 MaximalClique

```

1  #define MAXN 32
2  int n, m, Max;
3  ll v[MAXN], deg[MAXN]; //neighbors
4
5  void update_maximum(ll R){
6      int Size = 0;
7      while(R){
8          if(R&1)Size++;
9          R>>=1;
10     }
11     Max = max(Size, Max);
12 }
13
14 int pickPivot(ll P){
15     int pivot = -1, Max = -1;
16     memset(deg, 0, sizeof(deg));
17     for(int i = 0; i < n; i++){
18         if(P&(1LL<<i))//i is in P
19             if(pivot == -1){//i = default pivot
20                 pivot = i;
21                 Max = deg[i];
22             }
23         for(int j = 0; j < i; j++){
24             if((P&(1LL<<j))&&(v[i]&(1LL<<j))){
25                 deg[i]++;
26                 if(deg[i] > Max){
27                     Max = deg[i];
28                     pivot = i;
29                 }
30             }
31             deg[j]++;
32             if(deg[j] > Max){
33                 Max = deg[j];
34                 pivot = j;
35             }
36         }
37     }
38     return pivot;
39 }
40
41 void BronKerbosch(ll R, ll P, ll X){
42     if(!P){//P is empty, no candidates left
43         if(!X){
44             //clique
45             update_maximum(R);
46         }
47         return;
48     }
49     int u = pickPivot(P|X);
50     for(int i = 0; i <= n-1; i++){
51         if(P&(~v[u])&(1LL<<i))//vi is in P
52             BronKerbosch( R|(1LL<<i), P&v[i], X&v[i] );
53         P&=~(1LL<<i);
54         X|=(1LL<<i);
55     }
56 }
57 }
58 }

```

```

59 int main(){
60     ios::sync_with_stdio(false);
61     cin.tie(0);
62     while(cin >> n){
63         cin >> m;
64
65         Max = 0;
66         FOR(i,0,n-1)v[i] = 0;
67
68         int a, b;
69         FOR(i,1,m){
70             cin >> a >> b;
71             v[a]|=(1LL<<b);
72             v[b]|=(1LL<<a);
73         }
74         BronKerbosch(0, (1LL<<n)-1, 0);
75         cout << Max << '\n';
76     }
77     return 0;
78 }
79 }

```

## 4.8 MaxWeightPerfectMatch

```

1  struct Graph {
2      // Minimum General Weighted Matching (Perfect Match) 0-base
3      static const int MXN = 105;
4      int n, edge[MXN][MXN];
5      int match[MXN], dis[MXN], onstk[MXN];
6      vector<int> stk;
7      void init(int _n) {
8          n = _n;
9          for (int i=0; i<n; i++)
10             for (int j=0; j<n; j++)
11                 edge[i][j] = 0;
12     }
13     void add_edge(int u, int v, int w) {
14         edge[u][v] = edge[v][u] = w;
15     }
16     bool SPFA(int u){
17         if (onstk[u]) return true;
18         stk.push_back(u);
19         onstk[u] = 1;
20         for (int v=0; v<n; v++){
21             if (u != v && match[u] != v && !onstk[v]){
22                 int m = match[v];
23                 if (dis[m] > dis[u] - edge[v][m] + edge[u][v]){
24                     dis[m] = dis[u] - edge[v][m] + edge[u][v];
25                     onstk[v] = 1;
26                     stk.push_back(v);
27                     if (SPFA(m)) return true;
28                     stk.pop_back();
29                     onstk[v] = 0;
30                 }
31             }
32         }
33         onstk[u] = 0;
34         stk.pop_back();
35         return false;
36     }
37 }
38 int solve() {
39     // find a match
40     for (int i=0; i<n; i+=2){

```

```

40         match[i] = i+1, match[i+1] = i;
41     }
42     for(;;){
43         int found = 0;
44         for (int i=0; i<n; i++) dis[i] = onstk[i] = 0;
45         for (int i=0; i<n; i++){
46             stk.clear();
47             if (!onstk[i] && SPFA(i)){
48                 found = 1;
49                 while (stk.size()>=2){
50                     int u = stk.back(); stk.pop_back();
51                     int v = stk.back(); stk.pop_back();
52                     match[u] = v;
53                     match[v] = u;
54                 }
55             }
56         }
57         if (!found) break;
58     }
59     int ret = 0;
60     for (int i=0; i<n; i++)
61         ret += edge[i][match[i]];
62     ret /= 2;
63     return ret;
64 }
65 }graph;

```

## 4.9 HeavyLightDecomposition

```

1  // 以樹重切樹鏈
2  struct HLD {
3      static int MAX_N = 1e6+6;
4      vector<int> G[MAX_N];
5      int root=1; // set root!
6      int pa[MAX_N], son[MAX_N], d[MAX_N], w[MAX_N], link_top[
7          MAX_N];
8      void add_edge(int x, int y) {
9          G[x].push_back(y);
10         G[y].push_back(x);
11     }
12     void build_dfs(int now, int P) {
13         w[now]=1;
14         son[now]=-1;
15         pa[now]=P;
16         for(auto &i:G[now]){
17             if(i==P) continue;
18             d[i]=d[now]+1;
19             build_dfs(i,now);
20             if(~son[now]||w[son[now]]<w[i]) son[now]=i;
21             w[now]+=w[i];
22         }
23     }
24     void build_top(int now, int top) {
25         link_top[now]=top;
26         if(~son[now]) return;
27         build_top(son[now],top);
28         for(auto &i:G[now]){
29             if(i==pa[now]||i==son[now]) continue;
30             build_top(i,i);
31         }
32     }
33     void build() { // HLD build function
34         d[root]=1;

```

```

34 son[root]=pa[root]=-1;
35 build_dfs(root,-1);
36 build_top(root,root);
37 }
38 void find_lca(int x, int y) {
39     int tx=link_top[x], ty=link_top[y];
40     while(tx!=ty) {
41         if(d[tx]<d[ty]) {
42             swap(tx,ty);
43             swap(x,y);
44         }
45         tx=link_top[x=pa[x]];
46     }
47     return d[x]<d[y]?x:y;
48 }
49 };

```

## 4.10 匹配問題轉換

### 4.10.1 一般圖匹配問題轉換

1. 最大匹配邊數  $|+|$  最小邊涵蓋  $|=|V|$  (無孤立點)
2. 最大獨立集  $|+|$  最小點涵蓋  $|=|V|$
3. 最大權匹配  $\rightarrow$  最大權完美匹配: 用 0 邊補成完全圖
4. 最大權最大匹配  $\rightarrow$  最大權匹配: 先把所有邊加上 | 最負邊權重  $+1$ , 得到新的圖  $G'$  上沒有任何負邊, 然後所有邊再加上  $G'$  上所有邊權重和, 這樣最大權匹配就會 = 最大權最大匹配.

## 4.11 TarjanUndirected

```

1  /*
2   * AP, Bridge, BiCC in an undirected graph
3   * usage : init(), addEdge(), run()
4   * 0-base graph
5   */
6  class UndirectedTarjan
7  {
8  private:
9      int vn;
10     int ts;
11     vector<vector<int>> graph;
12     vector<int> low;
13     vector<int> dep;
14     stack<int> biccStk;
15     stack<int> bccStk;
16     vector<int> AP;
17     vector<vector<int>> BiCC;
18     vector<pair<int,int>> Bridge;
19     vector<vector<int>> BCC;
20     void reportAP(int ap) { AP.emplace_back(ap); }
21     void reportBiCC(int v){
22         vector<int> block(1, v);
23         while(biccStk.top() != v) {
24             block.emplace_back(biccStk.top()); biccStk.pop();
25         }
26         BiCC.emplace_back(block);
27     }
28     void reportBridge(int u, int v) { Bridge.emplace_back(u, v); }
29     void reportBCC(int v){
30         vector<int> bcc;

```

```

31     int x;
32     do{
33         x = bccStk.top(); bccStk.pop();
34         bcc.emplace_back(x);
35     } while(x != v);
36     BCC.emplace_back(bcc);
37 }
38 void dfs(int v, int p){
39     int childNum = 0;
40     bool maybeAP = false;
41     low[v] = dep[v] = ++ts;
42     biccStk.push(v), bccStk.push(v);
43     for(auto c : graph[v]){
44         if(c == p) continue;
45         if(dep[c] == 0) { // not visited
46             childNum++;
47             dfs(c, v);
48             low[v] = min(low[v], low[c]);
49             if(dep[v] <= low[c]) maybeAP = true;
50             if(dep[v] <= low[c]) reportBiCC(v);
51             if(dep[v] < low[c]) reportBridge(v, c);
52         }
53         low[v] = min(low[v], dep[c]);
54     }
55     if(dep[v] == low[v]) reportBCC(v);
56     if(v == p && childNum >= 2) reportAP(v);
57     if(v != p && maybeAP) reportAP(v);
58 }
59 public:
60     void init(int v){
61         vn = v, ts = 0;
62         graph.resize(v);
63         low.resize(v, 0);
64         dep.resize(v, 0);
65     }
66     void addEdge(int u, int v){
67         graph[u].emplace_back(v);
68         graph[v].emplace_back(u);
69     }
70     void run(){
71         for(int i = 0; i < vn; i++){
72             if(dep[i] == 0) dfs(i, i);
73         }
74         vector<int> getAP() { return AP; }
75         vector<vector<int>> getBiCC() { return BiCC; }
76         vector<pair<int,int>> getBridge() { return Bridge; }
77         vector<vector<int>> getBCC() { return BCC; }
78     };

```

## 4.12 spfa

```

1  typedef pair<int, ll> P;
2  #define idx first
3  #define w second
4  int vn, en;
5  vector<P> graph[N];
6  ll dist[N];
7
8  bool spfa() { // return true if neg cycle
9      for(int i = 0; i < vn; i++) dist[i] = INF; dist[0] = 0;
10     int cnt[N] = {0};
11     bool inq[N] = {false};
12     queue<int> q; q.push(0); inq[0] = true;

```

```

13 while(!q.empty()){
14     int s = q.front(); q.pop();
15     inq[s] = false;
16     for(auto e:graph[s]){
17         if(dist[e.idx] > dist[s]+e.w){
18             dist[e.idx] = dist[s]+e.w;
19             if(++cnt[e.idx] >= vn) return true;
20             if(!inq[e.idx]){
21                 inq[e.idx] = true;
22                 q.push(e.idx);
23             }
24         }
25     }
26 }
27 return false;
28 }

```

## 4.13 dijkstra

```

1  struct edge{int to, cost;};
2  typedef pair<int, int> P; //first = min distance, second = v
3  #define id
4  #define f first
5  #define s second
6  #define INF 2147483647
7
8  int V, E, S, F;
9  vector<edge> G[100];
10 int d[100];
11
12 void dijkstra()
13 {
14     priority_queue<P, vector<P>, greater<P>> q;
15     fill(d, d + V, INF);
16     d[S] = 0;
17     q.push(P(0, S));
18
19     while(!q.empty())
20     {
21         P p = q.top(); q.pop();
22         int v = p.s;
23         if(d[v] < p.f) continue;
24         for(int i = 0; i < G[v].size(); i++)
25         {
26             edge e = G[v][i];
27             if(d[e.to] > d[v] + e.cost)
28             {
29                 d[e.to] = d[v] + e.cost;
30                 q.push(P(d[e.to], e.to));
31             }
32         }
33     }
34 }

```

## 4.14 MaxWeightPerfectBiMatch

```

1  const int maxn = 500 + 3, INF = 0x3f3f3f3f;
2  int n, W[maxn][maxn];
3  int mat[maxn];

```

```

4 int Lx[maxn], Ly[maxn], slack[maxn];
5 bool S[maxn], T[maxn];
6
7 inline void tension(int &a, const int b) {
8     if(b < a) a = b;
9 }
10
11 inline bool match(int u) {
12     S[u] = true;
13     for(int v = 0; v < n; ++v) {
14         if(T[v]) continue;
15         int t = Lx[u] + Ly[v] - W[u][v];
16         if(!t) {
17             T[v] = true;
18             if(mat[v] == -1 || match(mat[v])) {
19                 mat[v] = u;
20                 return true;
21             }
22         } else tension(slack[v], t);
23     }
24     return false;
25 }
26
27 inline void update() {
28     int d = INF;
29     for(int i = 0; i < n; ++i)
30         if(!T[i]) tension(d, slack[i]);
31     for(int i = 0; i < n; ++i) {
32         if(S[i]) Lx[i] -= d;
33         if(T[i]) Ly[i] += d;
34     }
35 }
36
37 inline void KM() {
38     for(int i = 0; i < n; ++i) {
39         Lx[i] = Ly[i] = 0; mat[i] = -1;
40         for(int j = 0; j < n; ++j) Lx[i] = max(Lx[i], W[i][j]);
41     }
42     for(int i = 0; i < n; ++i) {
43         fill(slack, slack + n, INF);
44         while(true) {
45             for(int j = 0; j < n; ++j) S[j] = T[j] = false;
46             if(match(i)) break;
47             else update();
48         }
49     }
50 }

```

## 5 Math

### 5.1 extgcd

```

1 int extgcd(int a, int b, int &x, int &y){
2     int gcd = a;
3     if(b != 0)
4         gcd = extgcd(b, a%b, y, x), y -= (a/b)*x;
5     else x = 1, y = 0;
6     return gcd;
7 }
8 //維護  $a*x+b*y=gcd(a, b)$ 

```

### 5.2 NTT

```

1 typedef long long ll;
2
3 const ll P = (479<<21)+1;
4 const ll G = 3;
5 inline ll fpw(ll x, ll y, ll m){
6     ll rtn = 1;
7     for(x=(x>=m?x%m:x);y;y>=>1){
8         if(y&1) rtn = rtn*x%m;
9         x = x*x%m;
10    }
11    return rtn;
12 }
13 inline vector<ll> ntt(vector<ll> rtn, int Rev = 1){
14     int ntt_n = rtn.size();
15     for(int i=0,j=0;i<ntt_n;i++){
16         if(i>j) swap(rtn[i],rtn[j]);
17         for(int k=(ntt_n>>1);(j^=k)<k;k>=>1);
18     }
19     for(int i=2,m=1;i<=ntt_n;i<=1,m++){
20         ll w = 1, wn = fpw(G,(P-1)>>m,P), u, t;
21         int mh = i>>1;
22         for(int j=0;j<mh;j++){
23             for(int k=j;k<ntt_n;k+=i){
24                 u = rtn[k], t = w*rtn[k+mh]%P;
25                 rtn[k] = (u+t)%P;
26                 rtn[k+mh] = (u-t+P)%P;
27             }
28             w = w*wn%P;
29         }
30     }
31     if(!~Rev){
32         for(int i=1;i<ntt_n/2;i++) swap(rtn[i],rtn[ntt_n-i]);
33         ll Revn = fpw(ntt_n,P-2,P);
34         for(int i=0;i<ntt_n;i++) rtn[i] = rtn[i]*Revn%P;
35     }
36     return rtn;
37 }
38 // 把原多項式包成 long long 的 vector(poly), 並把項次拓展到  $2^i$ .
39 // 用 ntt(poly) 即可得到轉換後的結果.
40 // Rev 為 1 時為 NTT, 為 -1 時為 InvNTT.

```

### 5.3 GaussianJordan

```

1 const double EPS = 1e-8;
2 typedef vector<double> vec;
3 typedef vector<vec> mat;
4
5 //solve Ax=b
6 //if no sol/inf sol, return vec of size 0
7 vec gauss_jordan(const mat& A, const vec& b){
8     int n = A.size();
9     mat B(n, vec(n+1));
10    for(int i=0;i<n;i++)for(int j=0;j<n;j++)B[i][j]=A[i][j];
11
12    for(int i=0;i<n;i++)B[i][n]=b[i];
13
14    for(int i=0;i<n;i++){
15        int pivot=i;
16        for(int j=i;j<n;j++){

```

```

17            if(abs(B[j][i])>abs(B[pivot][i]))pivot=j;
18        }
19        swap(B[i],B[pivot]);
20        if(abs(B[i][i])<EPS)return vec();//no/inf sol
21
22        for(int j=i+1;j<=n;j++)B[i][j]/=B[i][i];
23        for(int j=0;j<n;j++){
24            if(i!=j){
25                for(int k=i+1;k<=n;k++)
26                    B[j][k]-=B[i][i]*B[i][k];
27            }
28        }
29    }
30    vec x(n);
31    for(int i=0;i<n;i++)x[i]=B[i][n];
32    return x;
33 }

```

### 5.4 EulerPhi

```

1 //find in  $O(\sqrt{N})$ 
2
3 int euler_phi(int N)
4 {
5     int res=N;
6     for(int i=2;i*i<=N;i++)
7     {
8         if(N%i==0)
9         {
10             res=res/(i-1);
11             for(;N%i==0;N/=i);
12         }
13     }
14     if(N!=1)res=res/N*(N-1);//self=prime
15     return res;
16 }
17
18 //tabulate in  $O(\text{MAXN})$ 
19
20 int euler[MAXN];
21
22 void euler_phi2()
23 {
24     for(int i=0;i<MAXN;i++)euler[i]=i;
25     for(int i=2;i<MAXN;i++){
26         if(euler[i]==i)
27         {
28             for(int j=i;j<MAXN;j+=i)
29             {
30                 euler[j]=euler[j]/i*(i-1);
31             }
32         }
33     }
34 }
35 }

```

## 5.5 FFT

```

1 const double PI = acos(-1.0);
2 struct Complex
3 {
4     double x,y;
5     Complex(){}
6     Complex(double a):x(a),y(0){}
7     Complex(double a, double b):x(a),y(b){}
8     Complex operator+ (const Complex &a){ return Complex(x+a.x,
9         y+a.y); }
10    Complex operator- (const Complex &a){ return Complex(x-a.x,
11        y-a.y); }
12    Complex operator* (const Complex &a){ return Complex(x*a.x-
13        y*a.y,x*a.y+y*a.x); }
14};
15inline vector<Complex> fft(vector<Complex> rtn, int Rev = 1)
16{
17    int fft_n = rtn.size();
18    for(int i=0,j=0;i<fft_n;i++)
19    {
20        if(i>j) swap(rtn[i],rtn[j]);
21        for(int k=(fft_n>>1);(j^=k)<k;k>>=1);
22    }
23    for(int i=2,m;i<=fft_n;i<=1)
24    {
25        m = i>>1;
26        for(int j=0;j<fft_n;j+=i)
27        {
28            for(int k=0;k<m;k++)
29            {
30                Complex y = rtn[j+k+m]*Complex(cos(2*PI/i*k), Rev*sin
31                    (2*PI/i*k));
32                rtn[j+k+m] = rtn[j+k]-y;
33                rtn[j+k] = rtn[j+k]+y;
34            }
35        }
36    }
37    for(int i=0;!--Rev&&i<fft_n;i++)
38        rtn[i].x = rtn[i].x/fft_n;
39    return rtn;
40}
41// Complex的x為實部, y為虛部.
42// 把原多項式包成Complex的vector(poly), 並把項次拓展到2^i, 用
43//    fft(poly)即可得到轉換後的結果.
44// Rev為1時為FFT, 為-1時為InvFFT.

```

## 5.6 BigInt

```

1 #define MAX_N 1000
2 #define MAX 100000
3 #define MAX_LOG 5
4 class BigInt{
5 public:
6     int sign;
7     long long m[MAX_N];
8     int l;
9     long long operator [](int i) const { return m[i]; }
10    long long operator [](int i) { return m[i]; }
11    BigInt(){ l=1, m[0]=0; sign=1; }
12    BigInt(int x){ (*this)=x; }

```

```

13    BigInt(const char *s){ (*this)=s; }
14    BigInt operator =(int x){
15        if(x<0) x=-x, sign=-1;
16        else sign=1;
17        for(l=1, m[l-1]=x%MAX, x/=MAX; x; m[l++]=x%MAX, x/=MAX)
18            ;
19        if(sign==1&&l==1&&m[0]==0) sign=1;
20        return *this;
21    }
22    BigInt operator =(const char *t){
23        int i, j, len;
24        const char *s;
25        if(t[0]=='-') sign=-1, s=t+1;
26        else sign=1, s=t;
27        for(len=0; s[len]>='0' && s[len]<='9'; len++);
28        for(l=(len+MAX_LOG-1)/MAX_LOG, i=0; i<l; i++)
29            for(m[i]=0, j=0; j<MAX_LOG; j++)
30                if(len-i*MAX_LOG-MAX_LOG+j>=0)
31                    m[i]=m[i]*10+s[len-i*MAX_LOG-MAX_LOG+j]- '0';
32        if(sign==1&&l==1&&m[0]==0) sign=1;
33        return *this;
34    }
35    bool scan(){
36        char s[MAX_N*MAX_LOG+10];
37        if(scanf("%s", s)==EOF) return 0;
38        else { *this=s; return 1; }
39    }
40    void print(){
41        int i;
42        char s[8];
43        if(sign==1) printf("-");
44        for(sprintf(s, "%0%dld", MAX_LOG), printf("%lld", m[l-1]), i=l-2; i>=0; printf(s, m[i]), i--);
45    }
46};
47bool operator <(const BigInt &x, const BigInt &y){
48    if(x.sign!=y.sign) return x.sign<y.sign;
49    int i;
50    if(x.l!=y.l) return (x.l<y.l) ^ (x.sign==1);
51    for(i=x.l-1; i>=0 && x[i]==y[i]; i--);
52    return (i>=0 && x[i]<y[i]) ^ (x.sign==1);
53}
54bool operator ==(const BigInt &x, const BigInt &y){
55    if(x.sign!=y.sign) return false;
56    int i;
57    if(x.l!=y.l) return 0;
58    for(i=x.l-1; i>=0 && x[i]==y[i]; i--);
59    return i<0;
60}
61BigInt operator +(BigInt x, const BigInt &y){
62    int i;
63    long long h;
64    for(h=0, i=0; i<x.l || i<y.l || h; i++){
65        h+=(i<x.l)*x[i]*x.sign+(i<y.l)*y[i]*y.sign;
66        x[i]=h%MAX;
67        h/=MAX;
68    }
69    x.l=i;
70    for(; x.l>1 && !x[x.l-1]; x.l--);
71    x.sign=(x[x.l-1]>0?1:-1);
72    if(x[x.l-1]>0){ for(i=0; i<x.l; i++) if(x[i]<0) x[i+1]--, x[i]
73        +=MAX; }
74    else for(i=0; i<x.l; i++) if(x[i]>0) x[i+1]++, x[i]-=MAX;
75    for(i=0; i<x.l; i++) x[i]*=x.sign;
76    if(x.sign==1&&x.l==1&&x[0]==0) x.sign=1;
77    return x;

```

```

76    }
77    BigInt operator -(BigInt x, const BigInt &y){
78        int i;
79        long long h;
80        for(h=0, i=0; i<x.l || i<y.l || h; i++){
81            h+=(i<x.l)*x[i]*x.sign-(i<y.l)*y[i]*y.sign;
82            x[i]=h%MAX;
83            h/=MAX;
84        }
85        x.l=i;
86        for(; x.l>1 && !x[x.l-1]; x.l--);
87        x.sign=(x[x.l-1]>0?1:-1);
88        if(x[x.l-1]>0){ for(i=0; i<x.l; i++) if(x[i]<0) x[i+1]--, x
89            [i]+=MAX; }
90        else for(i=0; i<x.l; i++) if(x[i]>0) x[i+1]++, x[i]-=MAX;
91        for(; x.l>1 && !x[x.l-1]; x.l--);
92        for(i=0; i<x.l; i++) x[i]*=x.sign;
93        if(x.sign==1&&x.l==1&&x[0]==0) x.sign=1;
94        return x;
95    }
96    BigInt operator *(BigInt x, int y){
97        int i, sign=1;
98        long long h;
99        if(y<0) y=-y, sign=-1;
100        for(h=0, i=0; i<x.l || h; i++){
101            h+=(i<x.l)*x[i]*y;
102            x[i]=h%MAX;
103            h/=MAX;
104        }
105        for(x.l=i; x.l>1 && !x[x.l-1]; x.l--);
106        x.sign=(x.sign==sign?1:-1);
107        if(x.sign==1&&x.l==1&&x[0]==0) x.sign=1;
108        return x;
109    }
110    BigInt operator /(BigInt x, int y){
111        int i, sign=1;
112        long long h;
113        if(y<0) y=-y, sign=-1;
114        for(h=0, i=x.l-1; i>=0; i--){
115            h=h*MAX+x[i];
116            x[i]=h/y;
117            h%=y;
118        }
119        for(; x.l>1 && !x[x.l-1]; x.l--);
120        x.sign=(x.sign==sign?1:-1);
121        if(x.sign==1&&x.l==1&&x[0]==0) x.sign=1;
122        return x;
123    }
124    int operator %(BigInt x, int y){
125        int i;
126        long long h;
127        for(h=0, i=x.l-1; i>=0; i--){
128            h=h*MAX+x[i];
129            h%=y;
130        }
131        if(x.sign==1) h=-h;
132        return h;
133    }
134    long long fl(double x) { return x<0?x-0.5:x+0.5; }
135    BigInt operator *(BigInt x, const BigInt &y){
136        if(y.l==1) return x*(y[0]*y.sign);
137        int i, N;
138        long long t;
139        vector<Complex> a, b;
140        for(i=0; i<x.l; i++) a.emplace_back(x[i]);
141        for(i=0; i<y.l; i++) b.emplace_back(y[i]);

```



```

141 for(N=1; N<x.l+y.l; N<=1);
142 while(N!=(int)(a.size())) a.emplace_back(0);
143 while(N!=(int)(b.size())) b.emplace_back(0);
144 a=fft(a), b=fft(b);
145 for(i=0; i<N; i++) a[i]=a[i]*b[i];
146 a=fft(a,-1);
147 for(i=0, t=0, x.l=0; i<N; i++){
148     t+=fl(a[i].x);
149     x[x.l++] = t*MAX;
150     t/=MAX;
151 } x[x.l++] = t;
152 for(; x.l>1 && !x[x.l-1]; x.l--);
153 x.sign=(x.sign==y.sign?-1);
154 if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
155 return x;
156 }
157 BigInt operator /(BigInt x, const BigInt &y){
158     if(y.l==1) return x/(y[0]*y.sign);
159     int i;
160     BigInt h;
161     for(h=0, i=x.l-1; i>=0; i--){
162         h=h*MAX+x[i];
163         if(h.l>y.l) x[i]=(h[h.l-1]*MAX*MAX+h[h.l-2]*MAX+h[h.l-3])
164             ;
165         if(h.l==y.l) x[i]=(h[h.l-1]*MAX+h[h.l-2]);
166         x[i]/=(y[y.l-1]*MAX+y[y.l-2]);
167         for(; x[i] && h<y*(x[i]*y.sign); x[i]--);
168         h=h-(y*(x[i]*y.sign));
169     }
170     for(; x.l>1 && !x[x.l-1]; x.l--);
171     x.sign=(x.sign==y.sign?-1);
172     if(x.sign==-1&&x.l==1&&x[0]==0) x.sign=1;
173     return x;
174 }
175 BigInt operator %(BigInt x, BigInt y){
176     if(y.l==1) return x%(y[0]*y.sign);
177     return x-(x/y)*y;
178 }

```

## 5.7 mobius

```

1 /* Mobius Function
2  * m(x) = 0, x has repeated factors
3  * m(x) = 1, x = 1
4  * m(x) = (-1)^k, x is the product of k distinct primes
5  * f(n) = sum(g(d)) for d|n,
6  * g(n) = sum( mu(n/d)f(d) ) = sum( mu(d)f(n/d) )
7 */
8 const int M = 100005;
9 ll sp[M], mobius[M];
10 void sieve(){
11     for(ll i = 0; i < M; i++) sp[i] = i;
12     for(ll i = 2; i < M; i++) if(sp[i] == i)
13     {
14         for(ll j = i*i; j < M; j += i)
15             if(sp[j] == j) sp[j] = i;
16     }
17 }
18 void makeMobius(){
19     for(ll i = 0; i < M; i++) mobius[i] = 1;
20     mobius[0] = 0;
21     for(ll i = 2; i < M; i++) if(sp[i] == i){
22         for(ll j = i; j < M; j += i) mobius[j] = -mobius[j];

```

```

23         for(ll j = i*i; j < M; j += i*i) mobius[j] = 0;
24     }
25 }

```

## 5.8 modeq

```

1 ll solve(ll *a, ll *b, ll *m, int N)
2 {
3     ll k = 0, h = 1, gcd, n, t, ar;
4     for (ll i = 0; i < N; i++)
5     {
6         gcd = extgcd(a[i] * h, m[i], ar, t);
7         t = (b[i] - a[i] * k) / gcd, n = abs(m[i] / gcd);
8         if (t * gcd != b[i] - a[i] * k) return -1;
9         t = ((ar * t) % n + n) % n;
10        k += h * t, h *= n, k %= h;
11    }
12    return (k % h + h) % h;
13 }
14 // 解n組a*x=b%m, 回傳x.
15 // 回傳的是最小非負整數解。無解回傳-1.

```

## 6 String

### 6.1 BWT

```

1 // use with suffix array
2 int pivot;
3 // BWT array size must be double of the data size
4 inline void BWT(char *tmp, char *in, char *out, int *SA, int
    *Rank){
5     int len=strlen(in);
6     for(int i=0;i<len;i++) tmp[i]=tmp[i+len]=in[i];
7     tmp[len*2]='|0';
8     SA_build(SA,Rank,tmp);
9     for(int i=0, j=0;i<2*len;i++){
10        if(SA[i]==len) pivot=j;
11        if(SA[i]<len)
12            out[j++]=in[(SA[i]+len-1)%len];
13    }
14    out[len]='|0';
15 }
16
17 inline void IBWT(char *in, char *out, int *tmp){
18     int len=strlen(in);
19     vector<int> idx[256];
20     for(int i=0;i<len;i++)
21         idx[in[i]].emplace_back(i);
22     for(int i=0,k=0;i<256;i++)
23         for(int j=0;j<(int)(idx[i].size());j++){
24             tmp[k++]=idx[i][j];
25         }
26     int p=pivot;
27     for(int i=0;i<len;i++)
28         out[i]=in[p=tmp[p]];
29     out[len]='|0';
30 }

```

## 6.2 SuffixArray

```

1 void SA_radix_sort(int *s, int *e, int *Rank, int rankcnt){
2     int box[MAX_N], tmp[MAX_N], len=e-s;
3     memset(box,0,sizeof(int)*rankcnt);
4     for(int i=0;i<len;i++) box[Rank[i]]++;
5     for(int i=1;i<rankcnt;i++) box[i]=box[i]+box[i-1];
6     for(int i=len-1;i>=0;i--) tmp[--box[Rank[s[i]]]]=s[i];
7     for(int i=0;i<len;i++) s[i]=tmp[i];
8 }
9 #define equal(a,b,c) c[a]!=c[b]||a+k>=len||c[a+k]!=c[b+k]
10 void SA_build(int *SA, int *Rank, char *S){
11     int ranktmp[MAX_N], len=strlen(S), rankcnt='z'+1;
12     for(int i=0;i<len;i++) Rank[i]=S[i];
13     for(int k=1;rankcnt!=len;k*=2){
14         for(int i=0;i<len;i++) SA[i]=(i+len-k)%len;
15         SA_radix_sort(SA+k, SA+len, Rank+k, rankcnt);
16         SA_radix_sort(SA, SA+len, Rank, rankcnt);
17         ranktmp[SA[0]]=0, rankcnt=0;
18         for(int i=1;i<len;i++){
19             ranktmp[SA[i]]=rankcnt+equal(SA[i-1], SA[i], Rank);
20             rankcnt++;
21             for(int i=0;i<len;i++) Rank[i]=ranktmp[i];
22         }
23     }
24 #undef equal

```

## 6.3 AC-Automation

```

1 #define SZ 25000
2 int nx[SZ][26], spt;
3 int fl[SZ], efl[SZ], ed[SZ];
4 int newnode(){
5     for(int i=0;i<26;i++) nx[spt][i]=0;
6     ed[spt]=0;
7     return spt++;
8 }
9 int add(char *s, int sptnow){
10     for(int i=0;s[i];i++){
11         int tmp=s[i]-'a';
12         if(nx[sptnow][tmp]==0) nx[sptnow][tmp]=newnode();
13         sptnow=nx[sptnow][tmp];
14     }
15     ed[sptnow]=1;
16     return sptnow;
17 }
18 int bfsq[SZ], qs, qe;
19 void make_fl(int root){
20     fl[root]=efl[root]=qs=qe=0;
21     bfsq[qe++]=root;
22     while(qs!=qe){
23         int p=bfsq[qs++];
24         for(int i=0;i<26;i++){
25             int t=nx[p][i];
26             if(t==0) continue;
27             int tmp=fl[p];
28             for(; tmp&&nx[tmp][i]==0; tmp=fl[tmp]);
29             fl[t]=tmp?nx[tmp][i]:root;
30             efl[t]=ed[fl[t]]?fl[t]:efl[fl[t]];
31             bfsq[qe++]=t;
32         }
33     }

```

```
34 }
```

## 6.4 LCP

```
1 //build query in O(nlogn), query LCP(i,j) in O(1)
2 int dp_height[MAX_N][20];
3 void height_build(int *SA, int *Rank, char *S, int *Height){
4     int len=strlen(S), k=0;
5     for(int i=0;i<len;i++){
6         if(Rank[i]==0) continue;
7         while(S[i+k] == S[SA[Rank[i]-1]+k]) k++;
8         Height[Rank[i]]=k;
9         if(k) k--;
10    } Height[0]=0;
11    for(int i=0;i<len;i++) dp_height[i][0]=Height[i];
12    for(int i=0;i<len;i++) for(int j=1;i+(1<<j)<len;j++)
13        dp_height[i][j]=min(dp_height[i][j-1], dp_height[i+(1<<(j-1))][j-1]);
14 }
15 int height_query(int x, int y){
16     int k=0;
17     while((1<<(k+1))<=y-x) k++;
18     return min(dp_height[x+1][k], dp_height[y-(1<<k)+1][k]);
19 }
```

## 6.5 Z-value

```
1 void Z_build(const char *S, int *Z){
2     Z[0]=0;
3     int b=0;
4     for(int i=1;S[i];i++){
5         if(Z[b]+b<i) Z[i]=0;
6         else Z[i]=min(Z[b]+b-i,Z[i-b]);
7         while(S[i+Z[i]]&&S[Z[i]]==S[i+Z[i]]) Z[i]++;
8         if(Z[i]+i>Z[b]+b) b=i;
9     }
10 }
```

## 6.6 KMP

```
1 void failure_build(const char *p, int *fail){
2     for(int i=1, j=fail[0]=-1; p[i]; i++){
3         while(j>=0&&p[j+1]!=p[i]) j=fail[j];
4         if(p[j+1]==p[i]) j++;
5         fail[i]=j;
6     }
7 }
8 int KMP(const char *T, const char *P, int *fail){
9     failure_build(P, fail);
10    for(int i=0, j=-1; T[i]; i++){
11        while(j>=0&&P[j+1]!=T[i]) j=fail[j];
12        if(P[j+1]==T[i]) j++;
13        if(!P[j+1]) return i-j;
14    }
15    return -1;
16 }
```

```
17
18 //使用方法: KMP(主字串, 待匹配字串, failure array)
19 //回傳: 第一個完全匹配的位置
```

## 7 other

### 7.1 2sat

```
1 const int N = 10; // 變數數量
2 bool adj[20][20]; // adjacency matrix
3 int visit[20]; // DFS visit record
4 int sat[20]; // 解

5
6 int not(int a) {return a<N ? a+N : a-N;}

7
8 // 另外一種方式
9 /*
10 int not(int a) {return a&1 ? a : a+1;}
11 int not(int a) {return a^1;}
12 */

13
14 bool dfs_try(int i){
15     if (visit[i] == 1 || sat[i] == 1) return true;
16     if (visit[i] == 2 || sat[i] == 2) return false;
17     visit[i] = 1;
18     visit[not(i)] = 2;
19     for (int j=0; j<N+N; ++j)
20         if (adj[i][j] && !dfs_try(j))
21             return false;
22     return true;
23 }

24
25 void dfs_mark(int i){
26     if (sat[i] == 1) return;
27     sat[i] = 1;
28     sat[not(i)] = 2;
29     for (int j=0; j<N+N; ++j)
30         if (adj[i][j])
31             dfs_mark(j);
32 }

33
34 void two_satisfiability(){
35     // 一次輸入一個括號
36     memset(adj, false, sizeof(adj));
37     int a, b;
38     while (cin >> a >> b){
39         map[not(a)][b] = true;
40         map[not(b)][a] = true;
41     }

42
43     // 找出一組解
44     for (int i=0; i<N; ++i){
45         memset(visit, 0, sizeof(visit));
46         if (dfs_try(i)) {dfs_mark(i); continue;}

47
48         memset(visit, 0, sizeof(visit));
49         if (dfs_try(not(i))) {dfs_mark(not(i)); continue;}

50
51         // 無解則立即結束。
52         return;
```

```
53     }
54
55     // 印出一組解。
56     for (int i=1; i<N; ++i)
57         if (sat[i] == 1)
58             cout << i;
59         else /*if (sat[i] == 2)*/
60             cout << "not" << i;
61 }
```

### 7.2 definss

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define pb push_back
4 #define pii pair<int,int>
5 #define pll pair<ll, ll>
6 #define pil pair<int,ll>
7 #define pli pair<ll,int>
8 #define ppi pair<pii,int>
9 #define pip pair<int,pii>
10 #define pdd pair<double, double>
11 #define f first
12 #define s second
13 #define MOD 1000000007
14 #define mkp make_pair
15 #define M_PI 3.14159265358979323846
16 #define FOR(i,l,r) for (int i=l;i<=r;i++)
17 #define LOR(i,l,r) for (ll i=l;i<=r;i++)
18 #define FORD(i,r,l) for (int i=r;i>=l;i--)
19 #define LORD(i,r,l) for (ll i=r;i>=l;i--)
20 #define INF 1000000000
21 #define CL(x) memset(x,0,sizeof(x))
22 typedef long long ll;

23
24 int main()
25 {
26     ios::sync_with_stdio(false);
27     cin.tie(0);

28
29     return 0;
30 }
```

### 7.3 PojTree

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 typedef long long int ll;
5 typedef pair<int, ll> P;
6 #define idx first
7 #define w second

8
9 const int N = 10004;
10 const ll INF = (1ll << 60);
11
12 int vn;
13 ll k;
14 vector<P> graph[N];
```

```

15 vector<int> dist;
16 ll subtreeSz[N];
17 bool isCentroid[N];
18
19 void init(){
20     for(int i = 1; i <= vn; i++){
21         graph[i].clear(), isCentroid[i] = false;
22     }
23
24 void buildTree(){
25     for(int i = 1; i < vn; i++){
26         int u, v, l; scanf("%d %d %d", &u, &v, &l);
27         graph[u].push_back(P(v, l));
28         graph[v].push_back(P(u, l));
29     }
30 }
31
32 ll calSubsz(int v, int p){
33     subtreeSz[v] = 1;
34     for(auto c:graph[v]){
35         if(isCentroid[c.idx] || c.idx == p) continue;
36         subtreeSz[v] += calSubsz(c.idx, v);
37     }
38     return subtreeSz[v];
39 }
40
41
42 P getCentroid(int v, int p, ll subsz){
43     P cen(-1, INF);
44     ll mxsonSz = -1;
45     for(auto c:graph[v]){
46         if(c.idx == p || isCentroid[c.idx]) continue;
47         P res = getCentroid(c.idx, v, subsz);
48         if(res.w < cen.w) cen = res;
49         mxsonSz = max(mxsonSz, subtreeSz[c.idx]);
50     }
51     mxsonSz = max(mxsonSz, subsz - subtreeSz[v]);
52     if(mxsonSz < cen.w) cen = P(v, mxsonSz);
53     return cen;
54 }
55
56 void getDist(int v, int p, ll w){
57     if(w > k) return;
58     dist.push_back(w);
59     for(auto c:graph[v]){
60         if(c.idx == p || isCentroid[c.idx]) continue;
61         getDist(c.idx, v, w+c.w);
62     }
63 }
64
65 ll calValidPair(int idx, ll w){
66     dist.clear();
67     getDist(idx, -1, w);
68     sort(dist.begin(), dist.end());
69     ll sum = 0;
70     for(int l = 0, r = dist.size()-1; l < r; ){
71         if(dist[r]+dist[l] <= k) sum += r-l, l++;
72         else r--;
73     }
74     return sum;
75 }
76
77 ll treedc(int v){
78     ll sum = 0;
79     // find centroid
80     calSubsz(v, v);
81
82     int cen = getCentroid(v, v, subtreeSz[v]).idx;
83     isCentroid[cen] = true;
84
85     sum += calValidPair(cen, 0);
86     for(auto c:graph[cen])
87     {
88         if(isCentroid[c.idx]) continue;
89         sum -= calValidPair(c.idx, c.w);
90         sum += treedc(c.idx);
91     }
92     return sum;
93 }
94
95 int main(){
96     while(scanf("%d %lld", &vn, &k) && vn && k)
97     {
98         init();
99         buildTree();
100         printf("%lld\n", treedc(1));
101     }
102     return 0;

```

# ACM ICPC TEAM

## REFERENCE -

### NTHU\_PILLARMEN

#### Contents

<b>1</b>	<b>DataStructure</b>	<b>1</b>
1.1	1d_segTree . . . . .	1
1.2	2d_st_tag . . . . .	1
1.3	undo_disjoint_set . . . . .	1
1.4	treap . . . . .	1
1.5	disjoint_set . . . . .	2
1.6	Matrix . . . . .	2
1.7	1d_segTree_tag . . . . .	2
1.8	BIT . . . . .	3
<b>2</b>	<b>Flow</b>	<b>3</b>
2.1	dinic . . . . .	3
2.2	MaxDensitySubgraph . . . . .	4

2.3	MinCostMaxFlow . . . . .	4
<b>3</b>	<b>Geometry</b>	<b>5</b>
3.1	point . . . . .	5
3.2	intercircle . . . . .	5
3.3	SegmentGeometry . . . . .	5
3.4	convexHullTrick . . . . .	6
3.5	Geometry . . . . .	6
3.6	nearestDist . . . . .	8
3.7	convexHull . . . . .	9
<b>4</b>	<b>Graph</b>	<b>9</b>
4.1	SCC . . . . .	9
4.2	lca . . . . .	9
4.3	bellman_Ford . . . . .	9
4.4	MaxMatching . . . . .	10
4.5	MinimumMeanCycle . . . . .	10
4.6	MaxBiMatching . . . . .	10
4.7	MaximalClique . . . . .	11
4.8	MaxWeightPerfectMatch . . . . .	11
4.9	HeavyLightDecomposition . . . . .	11
4.10	匹配問題轉換 . . . . .	12
4.10.1	一般圖匹配問題轉換 . . . . .	12
4.11	TarjanUndirected . . . . .	12
4.12	spfa . . . . .	12

4.13	dijkstra . . . . .	12
4.14	MaxWeightPerfectBiMatch . . . . .	12
<b>5</b>	<b>Math</b>	<b>13</b>
5.1	extgcd . . . . .	13
5.2	NTT . . . . .	13
5.3	GaussianJordan . . . . .	13
5.4	EulerPhi . . . . .	13
5.5	FFT . . . . .	14
5.6	BigInt . . . . .	14
5.7	mobius . . . . .	15
5.8	modeq . . . . .	15
<b>6</b>	<b>String</b>	<b>15</b>
6.1	BWT . . . . .	15
6.2	SuffixArray . . . . .	15
6.3	AC-Automation . . . . .	15
6.4	LCP . . . . .	16
6.5	Z-value . . . . .	16
6.6	KMP . . . . .	16
<b>7</b>	<b>other</b>	<b>16</b>
7.1	2sat . . . . .	16
7.2	definesss . . . . .	16
7.3	PojTree . . . . .	16