NCTU_TaNoShiI

Contents

1 Basic

1.1 Default

```
#include<bits/stdc++.h>
#define mp(a,b) make_pair((a),(b))
#define pii pair<int,int>
#define pdd pair<double,double>
#define pll pair<LL,LL>
#define pb(x) push_back(x)
#define x first
#define y second
#define sqr(x) ((x)*(x))
#define EPS 1e-6
#define mii map<int,int>
#define MEM(x) memset(x,0,sizeof(x))
#define MEMS(x) memset(x,-1,sizeof(x))
#define pi 3.14159265359
//#define INF 0x7fffffff
#define IOS ios_base::sync_with_stdio(0); cin.tie(0)
#define N 300005
using namespace std;
typedef long long LL;
```

2 DataStructure

2.1 PersistentTreap

```
const int MEM = 16000004;
struct Treap {
    static Treap nil, mem[MEM], *pmem;
    Treap *l, *r;
    char val;
    int size;
    Treap () : l(&nil), r(&nil), size(0) {}
    Treap (char _val) :
    l(&nil), r(&nil), val(_val), size(1) {}
} Treap::nil, Treap::mem[MEM], *Treap::pmem = Treap::
mem;
int size(const Treap *t) { return t->size; }
void pull(Treap *t) {
    if (!size(t)) return;
    t->size = size(t->l) + size(t->r) + 1;
}
Treap* merge(Treap *a, Treap *b) {
    if (!size(a)) return b;
    if (!size(b)) return a;
    Treap *t;
    if (rand() % (size(a) + size(b)) < size(a)) {
        t = new (Treap::pmem++) Treap(*a);
        t->r = merge(a->r, b);
        } else {
        t = new (Treap::pmem++) Treap(*b);
        t->l = merge(a, b->l);
    }
    pull(t);
    return t;
}
void split(Treap *t, int k, Treap *&a, Treap *&b) {
    if (!size(t)) a = b = &Treap::nil;
    else if (size(t->l) + 1 <= k) {
        a = new (Treap::pmem++) Treap(*t);
        split(t->r, k - size(t->l) - 1, a->r, b);
        pull(a);
        } else {
        b = new (Treap::pmem++) Treap(*t);
        split(t->l, k, a, b->l);
        pull(b);
    }
}
int nv;
Treap *rt[50005];
void print(const Treap *t) {
    if (!size(t)) return;
    print(t->l);
    cout << t->val;
    print(t->r);
}
int main(int argc, char** argv) {
    IOS;
    rt[nv=0] = &Treap::nil;
    Treap::pmem = Treap::mem;
    int Q, cmd, p, c, v;
    string s;
    cin >> Q;
    while (Q--) {
        cin >> cmd;
        if (cmd == 1) {
            // insert string s after position p
            cin >> p >> s;
            Treap *tl, *tr;
            split(rt[nv], p, tl, tr);
            for (int i=0; i<s.size(); i++)
            tl = merge(tl, new (Treap::pmem++) Treap(s[i]))
            ;
            rt[++nv] = merge(tl, tr);
            } else if (cmd == 2) {
            // remove c characters starting at position
            Treap *tl, *tm, *tr;
            cin >> p >> c;
            split(rt[nv], p-1, tl, tm);
            split(tm, c, tm, tr);
            rt[++nv] = merge(tl, tr);
            } else if (cmd == 3) {
            // print c characters starting at position p, in version v
            Treap *tl, *tm, *tr;
            cin >> v >> p >> c;
            split(rt[v], p-1, tl, tm);
            split(tm, c, tm, tr);
            print(tm);
            cout << "n";
        }
    }
    return 0;
}
```

2.2 PQTree

```
struct PQ_tree{
    int fail,res,n,tot;
    vector<int> G[N << 2];
    int ty[N << 2],sz[N << 2],szc[N << 2];
    bool s[N];
    inline int getstate(int u){
        if(szc[u] == 0) return 0;
        if(szc[u] == sz[u]) return 2;
        return 1;
    }
    void addson(int x, int y) {if(y) G[x].pb(y);}
    void join(int x, int y) {for(auto v : G[y]) G[x].pb(v);}
    int mergeP(vector<int> &vec){
        if(vec.size() == 0) return 0;
        if(vec.size() == 1) return vec[0];
        G[++tot] = vec;
        return tot;
    }
    void init(int _n){
        n = _n;
        tot = n + 1;
        memset(ty, 0, sizeof(ty));
        for(int i = 1; i <= n; i++) G[n + 1].pb(i);
        fail = 0;
    }
    void dfs(int u){
        sz[u] = u <= n;
        szc[u] = u <= n && s[u];
        for(auto v : G[u]){
            dfs(v);
            sz[u] += sz[v];
            szc[u] += szc[v];
        }
    }
    int check(int u, int t){
        if(fail) return 0;

        vector<int> vec[3];
        for(auto v : G[u]) vec[getstate(v)].pb(v);

        if(vec[1].size() > 2 || (t && vec[1].size() > 1)) return fail = 1, 0;
        if(t == 0 && vec[1].size() == 1 && vec[2].size() == 0) return check(vec[1][0], 0);

        if(ty[u] == 0){
            int p2 = mergeP(vec[2]);
            if(t == 0){
                G[u] = vec[0];
                if(vec[1].size() == 0) addson(u, p2);
                else{
                    int tmp1 = check(vec[1][0], 2);
                    addson(tmp1, p2);
                    if(vec[1].size() == 2) join(tmp1, check(vec[1][1], 1));
                    addson(u, tmp1);
                }
                return u;
            }
            else{
                ty[u] = 1;
                G[u].clear();
                addson(u, p2);
                if(vec[1].size() == 1) join(u, check(vec[1][0], 1));
                addson(u, mergeP(vec[0]));

                if(t == 2) reverse(G[u].begin(), G[u].end());
            }
            return u;
        }
        else{
            if(getstate(G[u].front()) > getstate(G[u].back())) reverse(G[u].begin(), G[u].end());

            int flag = 0;

            vector<int> tG;
            for(auto v : G[u]){
                int sta = getstate(v);
                if(sta == 0){
                    if(flag == 1) flag = 2;
                    tG.pb(v);
                }
                else if(sta == 2){
                    if(flag == 0) flag = 1;
                    else if(flag == 2) return fail = 2, 0;
                    tG.pb(v);
                }
                else{
                    int p1;
                    if(flag == 0) flag = 1, p1 = check(v, 2);
                    else if(flag == 1) flag = 2, p1 = check(v, 1);
                    else return fail = 3, 0;
                    for(auto x : G[v]) tG.pb(x);
                }
            }
            if(t && flag == 2) return fail = 4, 0;
            if(t == 1) reverse(tG.begin(), tG.end());
            G[u] = tG;
            return u;
        }
    }
    void dfs_permutation(int u){
        if(u <= n) {
            return;
        }
        if(!ty[u]){
            res=(LL)res*fra[G[u].size()]%MOD;
        }else if(G[u].size()!=1){
            res=(LL)res*2%MOD;
        }
        for(auto v : G[u]){
            dfs_permutation(v);
        }
    }
    int get_permutation(){
        res=1; dfs_permutation(n + 1);
        return res;
    }
    void restrict(vector<int> res){
        for(int i = 1; i <= n; i++) s[i] = 0;
        for(auto x : res) s[x] = 1;
        dfs(n + 1); check(n + 1, 0);
    }
};
```

2.3 KDtree

```
struct KDTree {
    struct Node {
        int x,y,x1,y1,x2,y2;
        int id,f;
        Node *L, *R;
    }tree[MXN];
    int n;
    Node *root;
    long long dis2(int x1, int y1, int x2, int y2) {
        long long dx = x1-x2;
        long long dy = y1-y2;
        return dx*dx+dy*dy;
    }
    static bool cmpx(Node& a, Node& b){ return a.x<b.x; }
    static bool cmpy(Node& a, Node& b){ return a.y<b.y; }
    void init(vector<pair<int,int>> ip) {
        n = ip.size();
        for (int i=0; i<n; i++) {
            tree[i].id = i;
            tree[i].x = ip[i].first;
            tree[i].y = ip[i].second;
        }
        root = build_tree(0, n-1, 0);
    }
    Node* build_tree(int L, int R, int dep) {
        if (L>R) return nullptr;
        int M = (L+R)/2;
        tree[M].f = dep%2;
        nth_element(tree+L, tree+M, tree+R+1, tree[M].f ?
        cmpy : cmpx);
        tree[M].x1 = tree[M].x2 = tree[M].x;
        tree[M].y1 = tree[M].y2 = tree[M].y;
        tree[M].L = build_tree(L, M-1, dep+1);
        if (tree[M].L) {
            tree[M].x1 = min(tree[M].x1, tree[M].L->x1);
            tree[M].x2 = max(tree[M].x2, tree[M].L->x2);
            tree[M].y1 = min(tree[M].y1, tree[M].L->y1);
            tree[M].y2 = max(tree[M].y2, tree[M].L->y2);
        }
        tree[M].R = build_tree(M+1, R, dep+1);
        if (tree[M].R) {
            tree[M].x1 = min(tree[M].x1, tree[M].R->x1);
            tree[M].x2 = max(tree[M].x2, tree[M].R->x2);
            tree[M].y1 = min(tree[M].y1, tree[M].R->y1);
            tree[M].y2 = max(tree[M].y2, tree[M].R->y2);
        }
        return tree+M;
    }
    int touch(Node* r, int x, int y, long long d2){
        long long dis = sqrt(d2)+1;
        if (x<r->x1-dis || x>r->x2+dis || y<r->y1-dis || y>
        r->y2+dis)
        return 0;
        return 1;
    }
    void nearest(Node* r, int x, int y, int &mID, long
    long &md2) {
        if (!r || !touch(r, x, y, md2)) return;
        long long d2 = dis2(r->x, r->y, x, y);
        if (d2 < md2 || (d2 == md2 && mID < r->id)) {
            mID = r->id;
            md2 = d2;
        }
        // search order depends on split dim
        if ((r->f == 0 && x < r->x) ||
        (r->f == 1 && y < r->y)) {
            nearest(r->L, x, y, mID, md2);
            nearest(r->R, x, y, mID, md2);
            } else {
            nearest(r->R, x, y, mID, md2);
            nearest(r->L, x, y, mID, md2);
        }
    }
    int query(int x, int y) {
        int id = 1029384756;
        long long d2 = 102938475612345678LL;
        nearest(root, x, y, id, d2);
        return id;
    }
}tree;
```

2.4 LCT

```
const int MXN = 100005;
const int MEM = 100005;

struct Splay {
  static Splay nil, mem[MEM], *pmem;
  Splay *ch[2], *f;
  int val, rev, size;
  Splay () : val(-1), rev(0), size(0) {
    f = ch[0] = ch[1] = &nil;
  }
  Splay (int _val) : val(_val), rev(0), size(1) {
    f = ch[0] = ch[1] = &nil;
  }
  bool isr() {
    return f->ch[0] != this && f->ch[1] != this;
  }
  int dir() {
    return f->ch[0] == this ? 0 : 1;
  }
  void setCh(Splay *c, int d) {
    ch[d] = c;
    if (c != &nil) c->f = this;
    pull();
  }
  void push() {
    if (rev) {
      swap(ch[0], ch[1]);
      if (ch[0] != &nil) ch[0]->rev ^= 1;
      if (ch[1] != &nil) ch[1]->rev ^= 1;
      rev=0;
    }
  }
  void pull() {
    size = ch[0]->size + ch[1]->size + 1;
    if (ch[0] != &nil) ch[0]->f = this;
    if (ch[1] != &nil) ch[1]->f = this;
  }
} Splay::nil, Splay::mem[MEM], *Splay::pmem = Splay::mem;
Splay *nil = &Splay::nil;

void rotate(Splay *x) {
  Splay *p = x->f;
  int d = x->dir();
  if (!p->isr()) p->f->setCh(x, p->dir());
  else x->f = p->f;
  p->setCh(x->ch[!d], d);
  x->setCh(p, !d);
  p->pull(); x->pull();
}

vector<Splay*> splayVec;
void splay(Splay *x) {
  splayVec.clear();
  for (Splay *q=x;; q=q->f) {
    splayVec.push_back(q);
    if (q->isr()) break;
  }
  reverse(begin(splayVec), end(splayVec));
  for (auto it : splayVec) it->push();
  while (!x->isr()) {
    if (x->f->isr()) rotate(x);
    else if (x->dir()==x->f->dir()) rotate(x->f),rotate(x);
    else rotate(x),rotate(x);
  }
}

Splay* access(Splay *x) {
  Splay *q = nil;
  for (;x!=nil;x=x->f) {
    splay(x);
    x->setCh(q, 1);
    q = x;
  }
  return q;
}
void evert(Splay *x) {
  access(x);
  splay(x);
  x->rev ^= 1;
  x->push(); x->pull();
}
void link(Splay *x, Splay *y) {
//  evert(x);
  access(x);
  splay(x);
  evert(y);
  x->setCh(y, 1);
}
void cut(Splay *x, Splay *y) {
//  evert(x);
  access(y);
  splay(y);
  y->push();
  y->ch[0] = y->ch[0]->f = nil;
}

int N, Q;
Splay *vt[MXN];

int ask(Splay *x, Splay *y) {
  access(x);
  access(y);
  splay(x);
  int res = x->f->val;
  if (res == -1) res=x->val;
  return res;
}
int main(int argc, char** argv) {
  scanf("%d%d", &N, &Q);
  for (int i=1; i<=N; i++)
    vt[i] = new (Splay::pmem++) Splay(i);
  while (Q--) {
    char cmd[105];
    int u, v;
    scanf("%s", cmd);
    if (cmd[1] == 'i') {
      scanf("%d%d", &u, &v);
      link(vt[v], vt[u]);
    } else if (cmd[0] == 'c') {
      scanf("%d", &v);
      cut(vt[1], vt[v]);
    } else {
      scanf("%d%d", &u, &v);
      int res=ask(vt[u], vt[v]);
      printf("%d\n", res);
    }
  }

  return 0;
}
```

2.5 Pbds

```

#include <bits/extc++.h>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,tree_order_statistics_node_update> set_t;
#include <ext/pb_ds/assoc_container.hpp>
typedef cc_hash_table<int,int> umap_t;
typedef priority_queue<int> heap;
#include<ext/rope>
using namespace __gnu_cxx;
int main(){
  // Insert some entries into s.
  set_t s; s.insert(12); s.insert(505);
  // The order of the keys should be: 12, 505.
  assert(*s.find_by_order(0) == 12);
  assert(*s.find_by_order(3) == 505);
  // The order of the keys should be: 12, 505.
  assert(s.order_of_key(12) == 0);
  assert(s.order_of_key(505) == 1);
  // Erase an entry.
  s.erase(12);
  // The order of the keys should be: 505.
  assert(*s.find_by_order(0) == 505);
  // The order of the keys should be: 505.
  assert(s.order_of_key(505) == 0);

  heap h1 , h2; h1.join( h2 );

  rope<char> r[ 2 ];
  r[ 1 ] = r[ 0 ]; // persistenet
  string t = "abc";
  r[ 1 ].insert( 0 , t.c_str() );
  r[ 1 ].erase( 1 , 1 );
  cout << r[ 1 ].substr( 0 , 2 );
}
```

3 Flow

3.1 Minimunwieghtmatchclique

```
struct Graph {
    // Minimum General Weighted Matching (Perfect Match) clique
    static const int MXN = 105;
    int n, edge[MXN][MXN];
    int match[MXN],dis[MXN],onstk[MXN];
    vector<int> stk;
    void init(int _n) {
        n = _n;
        MEM(edge);
    }
    void add_edge(int u, int v, int w) {
        edge[u][v] = edge[v][u] = w;
    }
    bool SPFA(int u){
        if (onstk[u]) return true;
        stk.pb(u);
        onstk[u] = 1;
        for (int v=0; v<n; v++){
            if (u != v && match[u] != v && !onstk[v]){
                int m = match[v];
                if (dis[m] > dis[u] - edge[v][m] + edge[u][v]) {
                    dis[m] = dis[u] - edge[v][m] + edge[u][v];
                    onstk[v] = 1;
                    stk.pb(v);
                    if (SPFA(m)) return true;
                    stk.pop_back();
                    onstk[v] = 0;
                }
            }
        }
        onstk[u] = 0;
        stk.pop_back();
        return false;
    }
    int solve() {
        // find a match
        for (int i=0; i<n; i+=2){
            match[i] = i+1;
            match[i+1] = i;
        }
        while (true){
            int found = 0;
            MEM(dis); MEM(onstk);
            for (int i=0; i<n; i++){
                stk.clear();
                if (!onstk[i] && SPFA(i)){
                    found = 1;
                    while (stk.size()>=2){
                        int u = stk.back(); stk.pop_back();
                        int v = stk.back(); stk.pop_back();
                        match[u] = v;
                        match[v] = u;
                    }
                }
            }
            if (!found) break;
        }
        int ret = 0;
        for (int i=0; i<n; i++)
        ret += edge[i][match[i]];
        ret /= 2;
        return ret;
    }
}graph;
```

3.2 CostFlow

```
struct CostFlow {
    static const int MXN = 205;
    static const long long INF = 102938475610293847LL;
    struct Edge {
        int v, r;
        long long f, c;
        Edge(int a,int b,int _c,int d):v(a),r(b),f(_c),c(d){
    }
    };
    int n, s, t, prv[MXN], prvL[MXN], inq[MXN];
    long long dis[MXN], fl, cost;
    vector<Edge> E[MXN];
    void init(int _n, int _s, int _t) {
        n = _n; s = _s; t = _t;
        for (int i=0; i<n; i++) E[i].clear();
        fl = cost = 0;
    }
    void add_edge(int u, int v, long long f, long long c)
    {
        E[u].pb(Edge(v, E[v].size() , f, c));
        E[v].pb(Edge(u, E[u].size()-1, 0, -c));
    }
    pll flow() {
        while (true) {
            for (int i=0; i<n; i++) {
                dis[i] = INF;
                inq[i] = 0;
            }
            dis[s] = 0;
            queue<int> que;
            que.push(s);
            while (!que.empty()) {
                int u = que.front(); que.pop();
                inq[u] = 0;
                for (int i=0; i<E[u].size(); i++) {
                    int v = E[u][i].v;
                    long long w = E[u][i].c;
                    if (E[u][i].f > 0 && dis[v] > dis[u] + w) {
                        prv[v] = u; prvL[v] = i;
                        dis[v] = dis[u] + w;
                        if (!inq[v]) {
                            inq[v] = 1;
                            que.push(v);
                        }
                    }
                }
            }
            if (dis[t] == INF) break;
            long long tf = INF;
            for (int v=t, u, l; v!=s; v=u) {
                u=prv[v]; l=prvL[v];
                tf = min(tf, E[u][l].f);
            }
            for (int v=t, u, l; v!=s; v=u) {
                u=prv[v]; l=prvL[v];
                E[u][l].f -= tf;
                E[v][E[u][l].r].f += tf;
            }
            cost += tf * dis[t];
            fl += tf;
        }
        return {fl, cost};
    }
}flow;
```

3.3 MincutTree

```
set<int> temp;
int Vis[3005];
int cvis[3005];
void dfs(int n){
  Vis[n]=1;
  for(auto it=v[n].begin();it!=v[n].end();it++){
    if(val[n][*it]>flow[n][*it]&&!Vis[*it]){
      dfs(*it);
      if(cvis[*it])
      temp.insert(*it);
    }
  }
}
int n;
int dc(set<int> s,int flag){
  if(s.size()==1)
  return *s.begin();
  for(int i=0;i<n;i++)
    for(auto it=v[i].begin();it!=v[i].end();it++)
    flow[i][*it]=0;
  for(auto it=s.begin();it!=s.end();it++){
    cvis[*it]=1;
  }
  int res=Flow(*s.begin(),*s.rbegin());
  MEM(Vis);
  dfs(*s.begin());
  temp.insert(*s.begin());
  for(auto it=s.begin();it!=s.end();it++){
    cvis[*it]=0;
  }
  set<int> s1,s2;
  swap(s1,temp);
  temp.clear();
  for(auto it=s1.begin();it!=s1.end();it++)
  s.erase(*it);
  swap(s2,s);
  int x=dc(s1,0);
  int y=dc(s2,1);
  vt[x].pb(mp(y,res));
  vt[y].pb(mp(x,res));
  if(flag==0)
  return x;
  else
  return y;
}
```

3.4 Dinic

```
struct Dinic{
    static const int MXN = 10000;
    struct Edge{ int v,f,re; Edge(int a,int b,int c):v(a),f(b),re(c){}};
    int n,s,t,level[MXN];
    vector<Edge> E[MXN];
    void init(int _n, int _s, int _t){
        n = _n; s = _s; t = _t;
        for (int i=0; i<=n; i++) E[i].clear();
    }
    void add_edge(int u, int v, int f){
        E[u].pb(Edge(v,f,E[v].size()));
        E[v].pb(Edge(u,0,E[u].size()-1));//direct
    }
    bool BFS(){
        MEMS(level);
        queue<int> que;
        que.push(s);
        level[s] = 0;
        while (!que.empty()){
            int u = que.front(); que.pop();
            for (auto it : E[u]){
                if (it.f > 0 && level[it.v] == -1){
                    level[it.v] = level[u]+1;
                    que.push(it.v);
                }
            }
        }
        return level[t] != -1;
    }
    int DFS(int u, int nf){
        if (u == t) return nf;
        int res = 0;
        for (auto &it : E[u]){
            if (it.f > 0 && level[it.v] == level[u]+1){
                int tf = DFS(it.v, min(nf,it.f));
                res += tf; nf -= tf; it.f -= tf;
                E[it.v][it.re].f += tf;
                if (nf == 0) return res;
            }
        }
        if (!res) level[u] = -1;
        return res;
    }
    int flow(int res=0){
        while ( BFS() )
        res += DFS(s,2147483647);
        return res;
    }
}flow;
```

3.5 GeneralGraphmatch

```
struct GenMatch { // 1-base
    static const int MAXN = 505;
    int V;
    bool el[MAXN][MAXN];
    int pr[MAXN];
    bool inq[MAXN],inp[MAXN],inb[MAXN];
    queue<int> qe;
    int st,ed;
    int nb;
    int bk[MAXN],djs[MAXN];
    int ans;
    void init(int _V) {
        V = _V;
        MEM(el); MEM(pr);
        MEM(inq); MEM(inp); MEM(inb);
        MEM(bk); MEM(djs);
        ans = 0;
    }
    void add_edge(int u, int v) {
        el[u][v] = el[v][u] = 1;
    }
    int lca(int u,int v) {
        memset(inp,0,sizeof(inp));
        while(1) {
            u = djs[u];
            inp[u] = true;
            if(u == st) break;
            u = bk[pr[u]];
        }
        while(1) {
            v = djs[v];
            if(inp[v]) return v;
            v = bk[pr[v]];
        }
        return v;
    }
    void upd(int u) {
        int v;
        while(djs[u] != nb) {
            v = pr[u];
            inb[djs[u]] = inb[djs[v]] = true;
            u = bk[v];
            if(djs[u] != nb) bk[u] = v;
        }
    }
    void blo(int u,int v) {
        nb = lca(u,v);
        memset(inb,0,sizeof(inb));
        upd(u); upd(v);
        if(djs[u] != nb) bk[u] = v;
        if(djs[v] != nb) bk[v] = u;
        for(int tu = 1; tu <= V; tu++)
        if(inb[djs[tu]]) {
            djs[tu] = nb;
            if(!inq[tu]){
                qe.push(tu);
                inq[tu] = 1;
            }
        }
    }
    void flow() {
        memset(inq,false,sizeof(inq));
        memset(bk,0,sizeof(bk));
        for(int i = 1; i <= V;i++)
        djs[i] = i;
        while(qe.size()) qe.pop();
        qe.push(st);
        inq[st] = 1;
        ed = 0;
        while(qe.size()) {
            int u = qe.front(); qe.pop();
            for(int v = 1; v <= V; v++)
            if(el[u][v] && (djs[u] != djs[v]) && (pr[u] !=
            v)) {
                if((v == st) || ((pr[v] > 0) && bk[pr[v]] >
                0))
                blo(u,v);
                else if(bk[v] == 0) {
                    bk[v] = u;
```

```cpp
                    if(pr[v] > 0) {
                        if(!inq[pr[v]]) qe.push(pr[v]);
                        } else {
                        ed = v;
                        return;
                    }
                }
            }
        }
    }
    void aug() {
        int u,v,w;
        u = ed;
        while(u > 0) {
            v = bk[u];
            w = pr[v];
            pr[v] = u;
            pr[u] = v;
            u = w;
        }
    }
    int solve() {
        memset(pr,0,sizeof(pr));
        for(int u = 1; u <= V; u++)
        if(pr[u] == 0) {
            st = u;
            flow();
            if(ed > 0) {
                aug();
                ans ++;
            }
        }
        return ans;
    }
}gp;
```

3.6 KM

```cpp
struct KM{
    // Maximum Bipartite Weighted Matching (Perfect Match)
    static const int MXN = 650;
    const int INF = 2147483647; //LL
    int px[MAXN],py[MAXN],match[MAXN],par[MAXN],n;
    int g[MAXN][MAXN],lx[MAXN],ly[MAXN],slack_y[MAXN];
    // ^^^^ long long
    void init(int _n){
        n = _n;
        for (int i=0; i<n; i++)
        for (int j=0; j<n; j++)
        g[i][j] = 0;
    }
    void add_edge(int x, int y, int w){ // LL
        g[x][y] = w;
    }
    void adjust(int y){
        match[y]=py[y];
        if(px[match[y]]!=-2)
            adjust(px[match[y]]);
    }
    bool dfs(int x){
        for(int y=0;y<n;++y){
            if(py[y]!=-1)continue;
            int t=lx[x]+ly[y]-g[x][y];//LL
            if(t==0){
                py[y]=x;
                if(match[y]==-1){
                    adjust(y);
                    return 1;
                }
                if(px[match[y]]!=-1)continue;
                px[match[y]]=y;
                if(dfs(match[y]))return 1;
            }else if(slack_y[y]>t){
                slack_y[y]=t;
                par[y]=x;
            }
        }
        return 0;
    }
    int solve(){//LL
        fill(match,match+n,-1);
        fill(ly,ly+n,0);
        for(int i=0;i<n;++i){
            lx[i]=-INF;
            for(int y=0;y<n;++y){
                lx[i]=max(lx[i],g[i][y]);
            }
        }
        for(int i=0;i<n;++i){
            for(int j=0;j<n;++j)slack_y[j]=INF;
            fill(px,px+n,-1);
            fill(py,py+n,-1);
            px[i]=-2;
            if(dfs(i))continue;
            bool flag=1;
            while(flag){
                int cut=INF; //LL
                for(int j=0;j<n;++j)
                    if(py[j]==-1)cut=min(cut,slack_y[j]);
                for(int j=0;j<n;++j){
                    if(px[j]!=-1)lx[j]-=cut;
                    if(py[j]!=-1)ly[j]+=cut;
                    else slack_y[j]-=cut;
                }
                for(int y=0;y<n;++y){
                    if(py[y]==-1&&slack_y[y]==0){
                        py[y]=par[y];
                        if(match[y]==-1){
                            adjust(y);
                            flag=0;
                            break;
                        }
                        px[match[y]]=y;
                        if(dfs(match[y])){
                            flag=0;
                            break;
                        }
                    }
                }
            }
        }
        int res=0;//LL
        for(int i=0;i<n;++i)
            res+=g[match[i]][i];
        return res;
    }
}graph;
```

3.7 SWmincut

```cpp
struct SW{ // O(V^3)
    static const int MXN = 514;
    int n,vst[MXN],del[MXN];
    int edge[MXN][MXN],wei[MXN];
    void init(int _n){
        n = _n;
        MEM(edge);
        MEM(del);
    }
    void add_edge(int u, int v, int w){
        edge[u][v] += w;
        edge[v][u] += w;
    }
    void search(int &s, int &t){
        MEM(vst); MEM(wei);
        s = t = -1;
        while (true){
            int mx=-1, cur=0;
            for (int i=0; i<n; i++)
            if (!del[i] && !vst[i] && mx<wei[i])
            cur = i, mx = wei[i];
            if (mx == -1) break;
            vst[cur] = 1;
            s = t;
            t = cur;
            for (int i=0; i<n; i++)
            if (!vst[i] && !del[i]) wei[i] += edge[cur][i];
        }
    }
    int solve(){
        int res = 2147483647;
        for (int i=0,x,y; i<n-1; i++){
            search(x,y);
            res = min(res,wei[y]);
            del[y] = 1;
            for (int j=0; j<n; j++)
            edge[x][j] = (edge[j][x] += edge[y][j]);
        }
        return res;
    }
}graph;
```

4 Geometry

4.1 Circleintersection

```cpp
using ld = double;
vector<pdd> interCircle(pdd o1, double r1, pdd o2,
double r2) {
    ld d2 = (o1 - o2) * (o1 - o2);
    ld d = sqrt(d2);
    if (d > r1+r2||d<abs(r1-r2)) return {};
    pdd u = 0.5*(o1+o2) + ((r2*r2-r1*r1)/(2*d2))*(o1-o2);
    double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d) *
(-r1+r2+d));
    pdd v = A / (2*d2) * pdd(o1.S-o2.S, -o1.F+o2.F);
    return {u+v, u-v};
}
```

4.2 Fermat's Point

```
If a angle greater or equal than degree 120
  return this point
else
  make regular triangle ABC' BCA' CAB'
  interaction AA' BB' CC'
```

4.3 Pointoperators

```
#define x first
#define y second
#define cpdd const pdd
struct pdd : pair<double, double> {
    using pair<double, double>::pair;
    pdd operator + (cpdd &p) const {
        return {x+p.x, y+p.y};
    }
    pdd operator - () const {
        return {-x, -y};
    }
    pdd operator - (cpdd &p) const {
        return (*this) + (-p);
    }
    pdd operator * (double f) const {
        return {f*x, f*y};
    }
    double operator * (cpdd &p) const {
        return x*p.x + y*p.y;
    }
};
double abs(cpdd &p) { return hypot(p.x, p.y); }
double arg(cpdd &p) { return atan2(p.y, p.x); }
double cross(cpdd &p, cpdd &q) { return p.x*q.y - p.y*q
.x; }
double cross(cpdd &p, cpdd &q, cpdd &o) { return cross(
p-o, q-o); }
pdd operator * (double f, cpdd &p) { return p*f; } //!! Not f*p !!
```

4.4 3DConvexHull

```
int flag[MXN][MXN];
struct Point{
    ld x,y,z;
    Point operator - (const Point &b) const {
        return (Point){x-b.x,y-b.y,z-b.z};
    }
    Point operator * (const ld &b) const {
        return (Point){x*b,y*b,z*b};
    }
    ld len() const { return sqrtl(x*x+y*y+z*z); }
    ld dot(const Point &a) const {
        return x*a.x+y*a.y+z*a.z;
    }
    Point operator * (const Point &b) const {
        return (Point){y*b.z-b.y*z,z*b.x-b.z*x,x*b.y-b.x*y
        };
    }
};
Point ver(Point a, Point b, Point c) {
    return (b - a) * (c - a);
}
vector<Face> convex_hull_3D(const vector<Point> pt) {
    int n = SZ(pt);
    REP(i,n) REP(j,n)
    flag[i][j] = 0;
    vector<Face> now;
    now.push_back((Face){0,1,2});
    now.push_back((Face){2,1,0});
    int ftop = 0;
    for (int i=3; i<n; i++){
        ftop++;
        vector<Face> next;
        REP(j, SZ(now)) {
            Face& f=now[j];
            ld d=(pt[i]-pt[f.a]).dot(ver(pt[f.a], pt[f.b], pt
            [f.c]));
            if (d <= 0) next.push_back(f);
            int ff = 0;
            if (d > 0) ff=ftop;
            else if (d < 0) ff=-ftop;
            flag[f.a][f.b] = flag[f.b][f.c] = flag[f.c][f.a]
            = ff;
        }
        REP(j, SZ(now)) {
            Face& f=now[j];
            if (flag[f.a][f.b] > 0 and flag[f.a][f.b] != flag
            [f.b][f.a])
            next.push_back((Face){f.a,f.b,i});
            if (flag[f.b][f.c] > 0 and flag[f.b][f.c] != flag
            [f.c][f.b])
            next.push_back((Face){f.b,f.c,i});
            if (flag[f.c][f.a] > 0 and flag[f.c][f.a] != flag
            [f.a][f.c])
            next.push_back((Face){f.c,f.a,i});
        }
        now=next;
    }
    return now;
}
```

4.5 Count Lattices

```
LL count_lattices(Fraction k, Fraction b, LL n) {
    // number of points(x,y) 0<=x<n,0<y<=k*x+b
    LL fk = floor(k);
    LL fb = floor(b);
    LL cnt = 0;
    if (fk >= 1 || fb >= 1) {
        cnt += (fk * (n - 1) + 2 * fb) * n / 2;
        k -= fk;b -= fb;
    }
    double t = k * n + b;
    LL ft = floor(t);
    if (ft >= 1) {
        cnt += count_lattices(1 / k, (t - ft) / k, ft);
    }
    return cnt;
}
```

4.6 Commontagnet

```
void tangents (pt c, double r1, double r2, vector<line> & ans) {
    double r = r2 - r1;
    double z = sqr(c.x) + sqr(c.y);
    double d = z - sqr(r);
    if (d < -EPS)  return;
    d = sqrt (abs (d));
    line l;
    l.a = (c.x * r + c.y * d) / z;
    l.b = (c.y * r - c.x * d) / z;
    l.c = r1;
    ans.push_back (l);
}

vector<line> tangents (circle a, circle b) {
    vector<line> ans;
    for (int i=-1; i<=1; i+=2)
        for (int j=-1; j<=1; j+=2)
            tangents (b-a, a.r*i, b.r*j, ans);
    for (size_t i=0; i<ans.size(); ++i)
        ans[i].c -= ans[i].a * a.x + ans[i].b * a.y;
    return ans;
}
```

4.7 Halfplaneintersection

```
typedef pdd Point;
typedef vector<Point> Polygon;
typedef pair<Point,Point> Line;
#define N 10
#define p1 first
#define p2 second
pdd operator-(const pdd &a,const pdd &b){
  return mp(a.x-b.x,a.y-b.y);
}
pdd operator+(const pdd &a,const pdd &b){
  return mp(a.x+b.x,a.y+b.y);
}
pdd operator*(const pdd &a,const double &b){
  return mp(b*a.x,b*a.y);
}
double cross(Point a, Point b){
  return a.x * b.y - a.y * b.x;
}
double cross(Point o, Point a, Point b){
  return cross(a-o,b-o);
}
double cross(Line l, Point p){
    return cross(l.p1, l.p2, p);
}
double arg(const pdd &a){
  return atan2(a.y,a.x);
}
bool parallel(Line l1, Line l2){
    return cross(l1.p2 - l1.p1, l2.p2 - l2.p1) < 1e-8&&cross(l1.p2 - l1.p1, l2.p2 - l2.p1) > -1e-8;
}
Point intersection(Line l1, Line l2){
    Point& a1 = l1.p1, &a2 = l1.p2;
    Point& b1 = l2.p1, &b2 = l2.p2;
    Point a = a2 - a1, b = b2 - b1, s = b1 - a1;
    return a1 + a * (cross(b, s) / cross(b, a));
}
bool cmp(Line l1, Line l2){
    return arg(l1.p2 - l1.p1) < arg(l2.p2 - l2.p1);
}
Polygon halfplane_intersection(vector<Line> hp){
    sort(hp.begin(), hp.end(), cmp);
    int L = 0, R = 0;
    vector<Line> l(N);
  vector<Point> p(N);
    l[R] = hp[0];
    for (int i=1; i<hp.size(); i++)
    {
        while (L < R && cross(hp[i], p[R-1]) < 0) R--;
        while (L < R && cross(hp[i], p[L])   < 0) L++;
        l[++R] = hp[i];
        if (parallel(l[R-1], hp[i]) &&
            cross(l[--R], hp[i].p1) > 0) l[R] = hp[i];
        if (L < R) p[R-1] = intersection(l[R], l[R-1]);
    }
    while (L < R && cross(l[L], p[R-1]) < 0) R--;
    if (R-L <= 1) return Polygon();//printf("?");
    if (L < R) p[R] = intersection(l[L], l[R]);
    Polygon ch;
    for (int i=L; i<=R; i++) ch.push_back(p[i]);
    ch.resize(unique(ch.begin(), ch.end()) - ch.begin());
    if (ch.size() > 1 && ch.front() == ch.back())
        ch.pop_back();
    return ch;
}
double cal(Polygon p){
  if(p.empty())
  return 0;
  p.pb(*p.begin());
  double ans=0;
  for(int i=0;i<p.size()-1;i++){
    ans+=p[i].x*p[i+1].y;
    ans-=p[i].y*p[i+1].x;
  }
  ans/=2;
  ans=abs(ans);
  return ans;
}
```

4.8 ConvexHull

```
sort(p,p+n);
pii ans[N];
ans[0]=p[0];
int k=0,now=0;
for(int tt=0;tt<2;tt++){
  for(int i=1;i<n;i++){
    while(now!=k&&cross(ans[now],p[i],ans[now-1])<=0) now--;
    ans[++now]=p[i];
  }
  k=now;
  reverse(p,p+n);
}
```

4.9 Triangulation

```
bool inCircle(pdd a, pdd b, pdd c, pdd d) {
    b = b - a;
    c = c - a;
    d = d - a;
    if (cross(b, c) < 0) swap(b, c);
    double m[3][3] = {
        {b.x, b.y, b*b},
        {c.x, c.y, c*c},
        {d.x, d.y, d*d}
    };
    double det = m[0][0] * (m[1][1]*m[2][2] - m[1][2]*m
    [2][1])
    + m[0][1] * (m[1][2]*m[2][0] - m[1][0]*m
    [2][2])
    + m[0][2] * (m[1][0]*m[2][1] - m[1][1]*m
    [2][0]);
    return det < 0;
}
bool intersect(pdd a, pdd b, pdd c, pdd d) {
    return cross(b, c, a) * cross(b, d, a) < 0 and
    cross(d, a, c) * cross(d, b, c) < 0;
}
const double EPS = 1e-12;
struct Triangulation {
    static const int MXN = 1e5+5;
    int N;
    vector<int> ord;
    vector<pdd> pts;
    set<int> E[MXN];
    vector<vector<int>> solve(vector<pdd> p) {
        N = SZ(p);
        ord.resize(N);
        for (int i=0; i<N; i++) {
            E[i].clear();
            ord[i] = i;
        }
        sort(ALL(ord), [&p](int i, int j) {
            return p[i] < p[j];
        });
        pts.resize(N);
        for (int i=0; i<N; i++) pts[i] = p[ord[i]];
        go(0, N);
        vector<vector<int>> res(N);
        for (int i=0; i<N; i++) {
            int o = ord[i];
            for (auto x: E[i]) {
                res[o].PB(ord[x]);
            }
        }
        return res;
    }
    void add_edge(int u, int v) {
        E[u].insert(v);
        E[v].insert(u);
    }
    void remove_edge(int u, int v) {
        E[u].erase(v);
        E[v].erase(u);
    }
    void go(int l, int r) {
        int n = r - l;
        if (n <= 3) {
            for (int i=l; i<r; i++)
            for (int j=i+1; j<r; j++) add_edge(i, j
            );
            return;
        }
        int md = (l+r)/2;
        go(l, md);
        go(md, r);
        int il = l, ir = r-1;
        while (1) {
            int nx = -1;
            for (auto i: E[il]) {
                double cs = cross(pts[il], pts[i], pts[
                ir]);
                if (cs > EPS ||
                (abs(cs) < EPS and abs(pts[i]-pts[
                ir]) < abs(pts[il]-pts[ir]))) {
                    nx = i;
                    break;
                }
            }
            if (nx != -1) {
                il = nx;
                continue;
            }
            for (auto i: E[ir]) {
                double cs = cross(pts[ir], pts[i], pts[
                il]);
                if (cs < -EPS ||
                (abs(cs) < EPS and abs(pts[i]-pts[
                il]) < abs(pts[ir]-pts[il]))) {
                    nx = i;
                    break;
                }
            }
            if (nx != -1) {
                ir = nx;
            } else break;
        }
        add_edge(il, ir);
        while (1) {
            int nx = -1;
            bool is2 = false;
            for (int i: E[il]) {
                if (cross(pts[il], pts[i], pts[ir]) < -
                EPS and
                (nx == -1 or inCircle(pts[il], pts[
                ir], pts[nx], pts[i]))) nx = i;
            }
            for (int i: E[ir]) {
                if (cross(pts[ir], pts[i], pts[il]) >
                EPS and
                (nx == -1 or inCircle(pts[il], pts[
                ir], pts[nx], pts[i]))) nx = i,
                is2 = 1;
            }
            if (nx == -1) break;
            int a = il, b = ir;
            if (is2) swap(a, b);
            for (auto i: E[a]) {
                if (intersect(pts[a], pts[i], pts[b],
                pts[nx])) {
                    remove_edge(a, i);
                }
            }
            if (is2) {
                add_edge(il, nx);
                ir = nx;
                } else {
                add_edge(ir, nx);
                il = nx;
            }
        }
    }
} tri;
```

4.10 Minkowskisum

```
bool cmp(const pll &p,const pll &q){
  int a=(p.x<0||(p.x==0&&p.y<0));
  int b=(q.x<0||(q.x==0&&q.y<0));
  if(a!=b)return a<b;
  return (p^q)>0;
}
vector<pll> minkowski(vector<pll> p, vector<pll> q){
  int n = p.size() , m = q.size();
  pll st=p[0]+q[0];
  vector<pll> v1,v2,v(n + m);
  for(int i = 0;i<n;i++) v1.pb(p[(i+1)%n]-p[i]);
  for(int i = 0;i<m;i++) v2.pb(q[(i+1)%m]-q[i]);
```

```cpp
  merge(v1.begin(),v1.end(),v2.begin(),v2.end(),v.begin(),cmp);
  vector<pll> h{st};
  pll last=mp(0,0);
  int fi=1;
  for(auto it:v){
    st=st+it;
    if(!fi&&!cmp(last,it)&&!cmp(it,last))
      h.pop_back();
    last=it;
    fi=0;
    h.pb(st);
  }
  h.pop_back();
  return h;
}
```

4.11 K-closet Pair

```cpp
#define F(n) Fi(i,n)
#define Fi(i,n) Fl(i,0,n)
#define Fl(i,l,n) for(int i=(l);i<(int)(n);++i)
#include <bits/stdc++.h>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/priority_queue.hpp>
using namespace std;
// using namespace __gnu_pbds;
typedef long long ll;
struct point {
  point(ll x_ = 0, ll y_ = 0): x(x_), y(y_) {}  ll x, y;
  inline bool operator<(const point &e_) const {
    return (x != e_.x ? x < e_.x : y < e_.y);
  }
  inline friend istream& operator>>(istream &is_, point& e_) {
    is_ >> e_.x >> e_.y;
    return is_;
  }
};
int k;
priority_queue<ll> PQ;
inline ll dist2(const point &e1, const point &e2) {
  ll res = (e1.x-e2.x)*(e1.x-e2.x)+(e1.y-e2.y)*(e1.y-e2.y);
  PQ.push(res);
  if (PQ.size() > k) {
    PQ.pop();
  }
  return res;
}
#define N 500005
point p[N];
queue<point> Q;
ll closet_point(int l, int m, int r, ll delta2) {
  ll xmid = p[m-1].x;
  while (!Q.empty()) {
    Q.pop();
  }
  for (int i = l, j = m ; i < m ; ++i) {
    if ((p[i].x-xmid)*(p[i].x-xmid) >= delta2) {
      continue;
    }
    while (j < r && p[j].y < p[i].y && (p[j].y-p[i].y)*(p[j].y-p[i].y) < delta2) {
      if ((p[j].x-xmid)*(p[j].x-xmid) < delta2) {
        Q.push(p[j]);
      }
      ++j;
    }
    while (!Q.empty() && Q.front().y < p[i].y && (Q.front().y-p[i].y)*(Q.front().y-p[i].y) > delta2) {
      Q.pop();
    }
    while (!Q.empty()) {
      delta2 = min(delta2, dist2(p[i], Q.front()));
      Q.pop();
    }
  }
  return delta2;
}
ll find_distance(int l, int r) {
  if (r - l <= 3000) {
    ll ans = 0x3f3f3f3f3f3f3f3f;
    for (int i = l ; i < r ; ++i)
      for (int j = i+1 ; j < r ; ++j)
        ans = min(ans, dist2(p[i], p[j]));
    return ans;
  }
  int m = (l+r)/2;
  ll delta2 = min(find_distance(l, m), find_distance(m, r));
  return min(delta2, closet_point(l, m, r, delta2));
}
int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n >> k;
  F(n) cin >> p[i];
  sort(p, p+n);
  find_distance(0, n);
  cout << PQ.top() << '\n';
}
```

4.12 MCC

```cpp
struct Mcc{
    // return pair of center and r^2
    static const int MAXN = 1000100;
    int n;
    pdd p[MAXN],cen;
    double r2;
    void init(int _n, pdd _p[]){
        n = _n;
        memcpy(p,_p,sizeof(pdd)*n);
    }
    double sqr(double a){ return a*a; }
    double abs2(pdd a){ return a*a; }
    pdd center(pdd p0, pdd p1, pdd p2) {
        pdd a = p1-p0;
        pdd b = p2-p0;
        double c1=abs2(a)*0.5;
        double c2=abs2(b)*0.5;
        double d = a.x*b.y-b.x*a.y;
        double x = p0.x + (c1 * b.y - c2 * a.y) / d;
        double y = p0.y + (a.x * c2 - b.x * c1) / d;
        return pdd(x,y);
    }
    pair<pdd,double> solve(){
        random_shuffle(p,p+n);
        r2=0;
        for (int i=0; i<n; i++){
            if (abs2(cen-p[i]) <= r2) continue;
            cen = p[i];
            r2 = 0;
            for (int j=0; j<i; j++){
                if (abs2(cen-p[j]) <= r2) continue;
                cen = 0.5 * (p[i]+p[j]);
                r2 = abs2(cen-p[j]);
                for (int k=0; k<j; k++){
                    if (abs2(cen-p[k]) <= r2) continue;
                    cen = center(p[i],p[j],p[k]);
                    r2 = abs2(cen-p[k]);
                }
            }
        }
        return {cen,r2};
    }
}mcc;
```

4.13 3Dto2D

```cpp
Point randomver(Point a){
  if(abs(a.x)<1e-6)return {1,0,0};
  if(abs(a.y)<1e-6)return {0,1,0};
  if(abs(a.z)<1e-6)return {0,0,1};
  Point ret = Point(a.y*a.z,a.x*a.z,-2*a.x*a.y);
  ret = ret * (1/ret.len());
  return ret;
}
vector<pdd> to2D(vector<Point> v,Point vec, Point p){
  for(auto &it:v){
    it = it-p;
  }
  //cout<<vec.x<<" "<<vec.y<<" "<<vec.z<<endl;
  Point r = randomver(vec);
  Point c = cross(vec,r);
  vector<pdd> ret;
  for(auto it:v){
    ld x = it*r/r.len();
    ld y = it*c/c.len();
    ret.pb(mp(x,y));
  //  cout<<x<<" "<<y<<" "<<it*r<<" "<<r.len()<<endl;
  }
  return ret;
}
```

4.14 LineIntersection

```cpp
pdd interPnt(pdd p1, pdd p2, pdd q1, pdd q2, bool &res)
{
    double f1 = cross(p2, q1, p1);
    double f2 = -cross(p2, q2, p1);
    double f = (f1 + f2);
    if(fabs(f) < EPS) {
        res = false;
        return {};
    }
    res = true;
    return (f2 / f) * q1 + (f1 / f) * q2;
}
```

5 Graph

5.1 KSP

```cpp
// time: O(|E| \lg |E| + |V| \lg |V| + K)
// memory: O(|E| \lg |E| + |V|)
```

```
struct KSP{ // 1-base
  struct nd{
    int u, v, d;
    nd(int ui = 0, int vi = 0, int di = INF)
    { u = ui; v = vi; d = di; }
  };
  struct heap{
    nd* edge; int dep; heap* chd[4];
  };
  static int cmp(heap* a,heap* b)
  { return a->edge->d > b->edge->d; }
  struct node{
    int v; LL d; heap* H; nd* E;
    node(){}
    node(LL _d, int _v, nd* _E)
    { d =_d; v = _v; E = _E; }
    node(heap* _H, LL _d)
    { H = _H; d = _d; }
    friend bool operator<(node a, node b)
    { return a.d > b.d; }
  };
  int n, k, s, t, dst[ N ];
  nd *nxt[ N ];
  vector<nd*> g[ N ], rg[ N ];
  heap *nullNd, *head[ N ];
  void init( int _n , int _k , int _s , int _t ){
    n = _n; k = _k; s = _s; t = _t;
    for( int i = 1 ; i <= n ; i ++ ){
      g[ i ].clear(); rg[ i ].clear();
      nxt[ i ] = head[ i ] = NULL;
      dst[ i ] = -1;
    }
  }
  void addEdge( int ui , int vi , int di ){
    nd* e = new nd(ui, vi, di);
    g[ ui ].push_back( e );
    rg[ vi ].push_back( e );
  }
  queue<int> dfsQ;
  void dijkstra(){
    while(dfsQ.size()) dfsQ.pop();
    priority_queue<node> Q;
    Q.push(node(0, t, NULL));
    while (!Q.empty()){
      node p = Q.top(); Q.pop();
      if(dst[p.v] != -1) continue;
      dst[ p.v ] = p.d;
      nxt[ p.v ] = p.E;
      dfsQ.push( p.v );
      for(auto e: rg[ p.v ])
        Q.push(node(p.d + e->d, e->u, e));
    }
  }
  heap* merge(heap* curNd, heap* newNd){
    if(curNd == nullNd) return newNd;
    heap* root = new heap;
    memcpy(root, curNd, sizeof(heap));
    if(newNd->edge->d < curNd->edge->d){
      root->edge = newNd->edge;
      root->chd[2] = newNd->chd[2];
      root->chd[3] = newNd->chd[3];
      newNd->edge = curNd->edge;
      newNd->chd[2] = curNd->chd[2];
      newNd->chd[3] = curNd->chd[3];
    }
    if(root->chd[0]->dep < root->chd[1]->dep)
      root->chd[0] = merge(root->chd[0],newNd);
    else
      root->chd[1] = merge(root->chd[1],newNd);
    root->dep = max(root->chd[0]->dep, root->chd[1]->dep) + 1;
    return root;
  }
  vector<heap*> V;
  void build(){
    nullNd = new heap;
    nullNd->dep = 0;
    nullNd->edge = new nd;
    fill(nullNd->chd, nullNd->chd+4, nullNd);
    while(not dfsQ.empty()){
      int u = dfsQ.front(); dfsQ.pop();
      if(!nxt[ u ]) head[ u ] = nullNd;
      else head[ u ] = head[nxt[ u ]->v];
      V.clear();
      for( auto&& e : g[ u ] ){
        int v = e->v;
        if( dst[ v ] == -1 ) continue;
        e->d += dst[ v ] - dst[ u ];
        if( nxt[ u ] != e ){
          heap* p = new heap;
          fill(p->chd, p->chd+4, nullNd);
          p->dep = 1;
          p->edge = e;
          V.push_back(p);
        }
      }
      if(V.empty()) continue;
      make_heap(V.begin(), V.end(), cmp);
#define L(X) ((X<<1)+1)
#define R(X) ((X<<1)+2)
      for( size_t i = 0 ; i < V.size() ; i ++ ){
        if(L(i) < V.size()) V[i]->chd[2] = V[L(i)];
        else V[i]->chd[2]=nullNd;
        if(R(i) < V.size()) V[i]->chd[3] = V[R(i)];
        else V[i]->chd[3]=nullNd;
      }
      head[u] = merge(head[u], V.front());
    }
  }
  vector<LL> ans;
  void first_K(){
    ans.clear();
    priority_queue<node> Q;
    if( dst[ s ] == -1 ) return;
    ans.push_back( dst[ s ] );
    if( head[s] != nullNd )
      Q.push(node(head[s], dst[s]+head[s]->edge->d));
    for( int _ = 1 ; _ < k and not Q.empty() ; _ ++ ){
      node p = Q.top(), q; Q.pop();
      ans.push_back( p.d );
      if(head[ p.H->edge->v ] != nullNd){
        q.H = head[ p.H->edge->v ];
        q.d = p.d + q.H->edge->d;
        Q.push(q);
      }
      for( int i = 0 ; i < 4 ; i ++ )
        if( p.H->chd[ i ] != nullNd ){
          q.H = p.H->chd[ i ];
          q.d = p.d - p.H->edge->d + p.H->chd[ i ]->edge->d;
          Q.push( q );
        }
    }
  }
  void solve(){
    dijkstra();
    build();
    first_K();
  }
} solver;
```

5.2 Planar

```
//skydog
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<int, int> ii;
typedef pair<ll, ll> l4;

#define mp make_pair
#define pb push_back

#define debug(x) cerr << #x << " = " << x << " "

const int N=400+1;

struct Planar
{
    int n,m,hash[N],fa[N],deep[N],low[N],ecp[N];
    vector<int> g[N],son[N];
    set< pair<int,int> > SDlist[N],proots[N];
    int nxt[N][2],back[N],rev[N];
    deque<int> q;
    void dfs(int u)
    {
        hash[u]=1; q.pb(u);
        ecp[u]=low[u]=deep[u];
        int v;
        for (int i = 0; i < g[u].size(); ++i)
            if(!hash[v=g[u][i]])
            {
                fa[v]=u;
                deep[v]=deep[u]+1;
                dfs(v);
                low[u]=min(low[u],low[v]);
                SDlist[u].insert(mp(low[v],v));
            }
            else ecp[u]=min(ecp[u],deep[v]);
        low[u]=min(low[u],ecp[u]);
    }

    int visited[N];

    void addtree(int u,int t1,int v,int t2)
    {
        nxt[u][t1]=v; nxt[v][t2]=u;
    }

    void findnxt(int u,int v,int& u1,int& v1)
    {
        u1=nxt[u][v^1];
        if(nxt[u1][0]==u) v1=0;
        else v1=1;
    }
```

```cpp

    void walkup(int u,int v)
    {
        back[v]=u;
        int v1=v,v2=v,u1=1,u2=0,z;
        for (;;)
        {
            if(hash[v1]==u || hash[v2]==u) break;
            hash[v1]=u;hash[v2]=u; z=max(v1,v2);
            if(z>n)
            {
                int p=fa[z-n];
                if(p!=u)
                {
                    proots[p].insert(mp(-low[z-n], z));
                    v1=p,v2=p,u1=0,u2=1;
                }
                else break;
            }
            else
            {
                findnxt(v1,u1,v1,u1);
                findnxt(v2,u2,v2,u2);
            }
        }
    }

    int topstack;
    pair<int,int> stack[N];

    int outer(int u,int v)
    {
        return ecp[v]<deep[u] || (SDlist[v].size() && SDlist[v].begin()->first<deep[u]);
    }

    int inside(int u,int v)
    {
        return proots[v].size()>0 || back[v]==u;
    }

    int active(int u,int v)
    {
        return inside(u,v) || outer(u,v);
    }

    void push(int a,int b)
    {
        stack[++topstack]=mp(a,b);
    }

    void mergestack()
    {
        int v1,t1,v2,t2,s,s1;
        v1=stack[topstack].first;t1=stack[topstack].second;
        topstack--;
        v2=stack[topstack].first;t2=stack[topstack].second;
        topstack--;

        s=nxt[v1][t1^1];
        s1=(nxt[s][1]==v1);
        nxt[s][s1]=v2;
        nxt[v2][t2]=s;

        SDlist[v2].erase( make_pair(low[v1-n],v1-n) );
        proots[v2].erase( make_pair(-low[v1-n],v1) );
    }

    void findnxtActive(int u,int t,int& v,int& w1,int S)
    {
        findnxt(u,t,v,w1);
        while(u!=v && !active(S,v))
            findnxt(v,w1,v,w1);
    }

    void walkdown(int S,int u)
    {
        topstack=0;
        int t1,v=S,w1,x2,y2,x1,y1,p;
        for(t1=0;t1<2;++t1)
        {
            findnxt(S,t1^1,v,w1);
            while(v!=S)
            {
                if(back[v]==u)
                {
                    while(topstack>0) mergestack();
                    addtree(S,t1,v,w1); back[v]=0;
                }
                if(proots[v].size())
                {
                    push(v,w1);
                    p=proots[v].begin()->second;
                    findnxtActive(p,1,x1,y1,u);
                    findnxtActive(p,0,x2,y2,u);
                    if(active(u,x1) && !outer(u,x1))
                        v=x1,w1=y1;
                    else if(active(u,x2) && !outer(u,x2))
                        v=x2,w1=y2;
                    else if(inside(u,x1) || back[x1]==u)
                        v=x1,w1=y1;
                    else v=x2,w1=y2;
                    push(p,v==x2);
                }
                else if(v>n || ( ecp[v]>=deep[u] && !outer(u,v ) ))
                    findnxt(v,w1,v,w1);
                else if(v<=n && outer(u,v) && !topstack)
                {
                    addtree(S,t1,v,w1); break;
                }
                else break;
            }
        }
    }

    int work(int u)
    {
        int v;
        for (int i = 0; i < g[u].size(); ++i)
            if(fa[v=g[u][i]]==u)
            {
                son[u].push_back(n+v);
                proots[n+v].clear();
                addtree(n+v,1,v,0);
                addtree(n+v,0,v,1);
            }
        for (int i = 0; i < g[u].size(); ++i)
            if(deep[v=g[u][i]]>deep[u]+1)
                walkup(u,v);
        topstack=0;
        for (int i = 0; i < son[u].size(); ++i) walkdown(son[u][i], u);
        for (int i = 0; i < g[u].size(); ++i)
            if(deep[v=g[u][i]]>deep[u]+1 && back[v])
                return 0;
        return 1;
    }

    void init(int _n)
    {
        n = _n;
        m = 0;
        for(int i=1;i<=2*n;++i)
        {
            g[i].clear();
            SDlist[i].clear();
            son[i].clear();
            proots[i].clear();
            nxt[i][0]=nxt[i][1]=0;
            fa[i]=0;
            hash[i]=0;low[i]=ecp[i]=deep[i]=back[i]=0;
            q.clear();
        }
    }
    void add(int u, int v)
    {
        ++m;
        g[u].pb(v); g[v].pb(u);
    }
    bool check_planar()
    {
        if(m>3*n-5)
            return false;
        //  memset(hash,0,sizeof hash);
        for(int i=1;i<=n;++i)
            if(!hash[i])
            {
                deep[i]=1;
                dfs(i);
            }
        memset(hash,0,sizeof hash);
        //memset(hash, 0, (2*n+1)*sizeof(hash[0]));
        // originally only looks at last n element
        assert(q.size() == n);
        while (!q.empty())
        {
            if (!work(q.back()))
                return false;
            q.pop_back();
        }
        return true;
    }
} base, _new;
vector<ii> edges;
int n, m;
inline void build(int n, Planar &_new)
{
    _new.init(n);
    for (auto e : edges)
        _new.add(e.first, e.second);
}
void end()
{
    puts("-1");
    exit(0);
}
bool vis[N];
const int maxp = 5;
int path[maxp], tp=0;
void dfs(int cur)
{
    vis[cur] = true;
    path[tp++] = cur;
    if (tp == maxp)
    {
    auto it = lower_bound(base.g[cur].begin(), base.g[cur].end(), path[0]);
        if ( it != base.g[cur].end() && *it == path[0])
        {
            //a cycle
            int x = n+1;
            for (int i = 0; i < 5; ++i) edges.pb(mp(x, path[i]));
            build(x, _new);
            if (_new.check_planar())
            {
                for (int i = 0; i < maxp; ++i) printf("%d%c", path[i], i==maxp-1?'\n':' ');
                exit(0);
            }
            for (int i = 0; i < 5; ++i) edges.pop_back();
        }
    }
    else
    {
        for (auto e : base.g[cur]) if (!vis[e]) dfs(e);
    }
    vis[cur] = false;
    --tp;
}
int main()
{
    scanf("%d %d", &n, &m);
    if (n <= 4)
      {
      assert(false);
  puts("0"); return 0;
      }
    for (int i = 0; i < m; ++i)

    {
        int u, v; scanf("%d %d", &u, &v);
        edges.pb(mp(u, v));
    }
    build(n, base);
    if (!base.check_planar()) end();
    for (int i = 1; i <= n; ++i)
        sort(base.g[i].begin(), base.g[i].end());
    for (int i = 1; i <= n; ++i)
        dfs(i);
    end();
}
```

5.3 MMC

```
/* minimum mean cycle 最小平均值環*/
const int MXN = 16004;
const int MAXE = 1805;
const int MAXN = 35;
const double inf = 1029384756;
const double eps = 1e-6;
struct Edge {
    int v,u;
    double c;
};
int n,m,prv[MAXN][MAXN], prve[MAXN][MAXN], vst[MAXN];
Edge e[MAXE];
vector<int> edgeID, cycle, rho;
double d[MAXN][MAXN];
inline void bellman_ford() {
    for(int i=0; i<n; i++) d[0][i]=0;
    for(int i=0; i<n; i++) {
        fill(d[i+1], d[i+1]+n, inf);
        for(int j=0; j<m; j++) {
            int v = e[j].v, u = e[j].u;
            if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
                d[i+1][u] = d[i][v]+e[j].c;
                prv[i+1][u] = v;
                prve[i+1][u] = j;
            }
        }
    }
}
double karp_mmc() {
    // returns inf if no cycle, mmc otherwise
    double mmc=inf;
    int st = -1;
    bellman_ford();
    for(int i=0; i<n; i++) {
        double avg=-inf;
        for(int k=0; k<n; k++) {
            if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i])
            /(n-k));
            else avg=max(avg,inf);
        }
        if (avg < mmc) tie(mmc, st) = tie(avg, i);
    }
    MEM(vst); edgeID.clear(); cycle.clear(); rho.clear();
    for (int i=n; !vst[st]; st=prv[i--][st]) {
        vst[st]++;
        edgeID.pb(prve[i][st]);
        rho.pb(st);
    }
    while (vst[st] != 2) {
        int v = rho.back(); rho.pop_back();
        cycle.pb(v);
        vst[v]++;
    }
    reverse(edgeID.begin(),edgeID.end());
    edgeID.resize(cycle.size());
    return mmc;
}
```

5.4 SomeTheroem

```
/*
General graph
|maximum independent set|+|minimum vertex cover|=|V|
|maximum independent edge|+|minimum edge cover|=|V|
      ||
Max_match
Bipartite graph
|Maximun independent set|=|Minimun edge cover|
|Maximun independent edge|=|Minimun vertex cover|
|Maximun Independent set|+|Minimun vertex cover|=|V|
           +                    +
|Maximun Independent edge|+|Minimun edge cover|=|V|
          ||                        ||
         |V|                       |V|
DAG dilworth's theorem
Minimal chain cover = Maximal antichain
Maximal atichain = Minimal antichain cover
number of labeled forest n vertices with k tree
1,2,3,4...k belong different tree
kn^(n-k-1)
Erdős – Gallai theorem
d_1 \geq d_2 ... \geq d_n
\sum^k_{i=1} d_i \leq k(k-1) + \sum_{i=k+1}^n min(d_i,k)
*/
```

5.5 Dominator

```
struct dominator_tree{
  static const int MAXN=5005;
  int n;// 1-base
  vector<int> G[MAXN],rG[MAXN];//存圖和反向圖
  int pa[MAXN],dfn[MAXN],id[MAXN],dfnCnt;
  int semi[MAXN],idom[MAXN],best[MAXN];
  vector<int> tree[MAXN];//dominator_tree存這裡
  void init(int _n){
    n=_n;
    for(int i=1;i<=n;++i)G[i].clear(),rG[i].clear();
  }
  void add_edge(int u,int v){
    G[u].push_back(v);
    rG[v].push_back(u);
  }
  void dfs(int u){
    id[dfn[u]=++dfnCnt]=u;
    for(auto v:G[u]) if(!dfn[v]){
      dfs(v),pa[dfn[v]]=dfn[u];
    }
  }
  int find(int y,int x){
    if(y<=x)return y;
    int tmp=find(pa[y],x);
    if(semi[best[y]]>semi[best[pa[y]]])
      best[y]=best[pa[y]];
    return pa[y]=tmp;
  }
  void tarjan(int root){
    dfnCnt=0;
    for(int i=1;i<=n;++i){
      dfn[i]=idom[i]=0;
      tree[i].clear();
      best[i]=semi[i]=i;
    }
    dfs(root);
    for(int i=dfnCnt;i>1;--i){
      int u=id[i];
      for(auto v:rG[u]) if(v=dfn[v]){
        find(v,i);
        semi[i]=min(semi[i],semi[best[v]]);
      }
      tree[semi[i]].push_back(i);
      for(auto v:tree[pa[i]]){
        find(v,pa[i]);
        idom[v] = semi[best[v]]==pa[i] ? pa[i] : best[v];
      }
      tree[pa[i]].clear();
    }
    for(int i=2; i<=dfnCnt; ++i){
      if(idom[i]!=semi[i]) idom[i]=idom[idom[i]];
      tree[id[idom[i]]].push_back(id[i]);
    }
  }
};
```

5.6 DMST

```
struct zhu_liu{
  static const int MAXN=1100,MAXM=1005005;
  struct node{
    int u,v;
    LL w,tag;
    node *l,*r;
    node(int u=0,int v=0,LL w=0):u(u),v(v),w(w),tag(0),l(0),r(0){}
    void down(){
      w+=tag;
      if(l)l->tag+=tag;
      if(r)r->tag+=tag;
      tag=0;
    }
  }mem[MAXM];
  node *pq[MAXN*2],*E[MAXN*2];
  int st[MAXN*2],id[MAXN*2],m,from[MAXN*2];
  void init(int n){
    for(int i=1;i<=n;++i){
      pq[i]=E[i]=0;
      st[i]=id[i]=i;
      from[i]=0;
    }m=0;
  }
  node *merge(node *a,node *b){//skew heap
    if(!a||!b)return a?a:b;
    a->down(),b->down();
    if(b->w<a->w)return merge(b,a);
    if(b->w==a->w&&b->v<a->v)return merge(b,a);//
    swap(a->l,a->r);
    a->l=merge(b,a->l);
    return a;
  }
  void add_edge(int u,int v,LL w){
    if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=node(u,v,w)));
  }
  int find(int x,int *st){
    return st[x]==x?x:st[x]=find(st[x],st);
  }
  LL build(int root,int n){
    LL ans=0;int N=n,all=n;
    for(int i=1;i<=N;++i){
      if(i==root||!pq[i])continue;
      while(pq[i]){
        pq[i]->down(),E[i]=pq[i];
        pq[i]=merge(pq[i]->l,pq[i]->r);
        if(find(E[i]->u,id)!=find(i,id))break;
      }
      if(find(E[i]->u,id)==find(i,id))continue;
      from[E[i]->v]=E[i]->u;
      ans+=E[i]->w;
      if(find(E[i]->u,st)==find(i,st)){
        if(pq[i])pq[i]->tag-=E[i]->w;
        pq[++N]=pq[i],id[N]=N;
        for(int u=find(E[i]->u,id);u!=i;u=find(E[u]->u,id)){
          if(pq[u])pq[u]->tag-=E[u]->w;
          id[find(u,id)]=N;
          pq[N]=merge(pq[N],pq[u]);
        }
        st[N]=find(i,st);
        id[find(i,id)]=N;
      }else st[find(i,st)]=find(E[i]->u,st),--all;
    }
    return all==1?ans:-1;//圖不連通就無解
  }
}MST;
```

5.7 SCC

```
struct Scc{
    int n, nScc, vst[MXN], bln[MXN];
    vector<int> E[MXN], rE[MXN], vec;
    void init(int _n){
        n = _n;
        for (int i=0; i<MXN; i++){
            E[i].clear();
            rE[i].clear();
        }
    }
    void add_edge(int u, int v){
        E[u].pb(v);
        rE[v].pb(u);
    }
    void DFS(int u){
        vst[u]=1;
        for (auto v : E[u])
        if (!vst[v]) DFS(v);
        vec.pb(u);
    }
    void rDFS(int u){
        vst[u] = 1;
        bln[u] = nScc;
        for (auto v : rE[u])
        if (!vst[v]) rDFS(v);
    }
    void solve(){
        nScc = 0;
        vec.clear();
        MEM(vst);
        for (int i=0; i<n; i++)
        if (!vst[i]) DFS(i);
        reverse(vec.begin(),vec.end());
        FZ(vst);
        for (auto v : vec){
            if (!vst[v]){
                rDFS(v);
                nScc++;
            }
        }
    }
};
```

5.8 GeneralGraphMaximunValueMatch

```
#include<bits/stdc++.h>
using namespace std;
//from vfleaking
//自己進行一些進行一些小修改
#define INF INT_MAX
#define MAXN 400
struct edge{
  int u,v,w;
  edge(){}
  edge(int u,int v,int w):u(u),v(v),w(w){}
};
int n,n_x;
edge g[MAXN*2+1][MAXN*2+1];
int lab[MAXN*2+1];
int match[MAXN*2+1],slack[MAXN*2+1],st[MAXN*2+1],pa[MAXN*2+1];
int flower_from[MAXN*2+1][MAXN+1],S[MAXN*2+1],vis[MAXN*2+1];
vector<int> flower[MAXN*2+1];
queue<int> q;
inline int e_delta(const edge &e){ // does not work inside blossoms
  return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
}
inline void update_slack(int u,int x){
  if(!slack[x]||e_delta(g[u][x])<e_delta(g[slack[x]][x]))slack[x]=u;
}
inline void set_slack(int x){
  slack[x]=0;
  for(int u=1;u<=n;++u)
    if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x){
  if(x<=n)q.push(x);
  else for(size_t i=0;i<flower[x].size();i++)q_push(flower[x][i]);
}
inline void set_st(int x,int b){
  st[x]=b;
  if(x>n)for(size_t i=0;i<flower[x].size();++i)
      set_st(flower[x][i],b);
}
inline int get_pr(int b,int xr){
  int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
  if(pr%2==1){//檢查他在前一層圖是奇點還是偶點
    reverse(flower[b].begin()+1,flower[b].end());
    return (int)flower[b].size()-pr;
  }else return pr;
}
inline void set_match(int u,int v){
  match[u]=g[u][v].v;
  if(u>n){
    edge e=g[u][v];
    int xr=flower_from[u][e.u],pr=get_pr(u,xr);
    for(int i=0;i<pr;++i)set_match(flower[u][i],flower[u][i^1]);
    set_match(xr,v);
    rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
  }
}
inline void augment(int u,int v){
  for(;;){
    int xnv=st[match[u]];
    set_match(u,v);
    if(!xnv)return;
    set_match(xnv,st[pa[xnv]]);
    u=st[pa[xnv]],v=xnv;
  }
}
inline int get_lca(int u,int v){
  static int t=0;
  for(++t;u||v;swap(u,v)){
    if(u==0)continue;
    if(vis[u]==t)return u;
    vis[u]=t;//這種方法可以不用清空v陣列
    u=st[match[u]];
    if(u)u=st[pa[u]];
  }
  return 0;
}
inline void add_blossom(int u,int lca,int v){
  int b=n+1;
  while(b<=n_x&&st[b])++b;
  if(b>n_x)++n_x;
  lab[b]=0,S[b]=0;
  match[b]=match[lca];
  flower[b].clear();
  flower[b].push_back(lca);
  for(int x=u,y;x!=lca;x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  reverse(flower[b].begin()+1,flower[b].end());
  for(int x=v,y;x!=lca;x=st[pa[y]])
    flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
  set_st(b,b);
  for(int x=1;x<=n_x;++x)g[b][x].w=g[x][b].w=0;
  for(int x=1;x<=n;++x)flower_from[b][x]=0;
  for(size_t i=0;i<flower[b].size();++i){
    int xs=flower[b][i];
    for(int x=1;x<=n_x;++x)
      if(g[b][x].w==0||e_delta(g[xs][x])<e_delta(g[b][x]))
        g[b][x]=g[xs][x],g[x][b]=g[x][xs];
    for(int x=1;x<=n;++x)
      if(flower_from[xs][x])flower_from[b][x]=xs;
  }
  set_slack(b);
}
inline void expand_blossom(int b){ // S[b] == 1
  for(size_t i=0;i<flower[b].size();++i)
    set_st(flower[b][i],flower[b][i]);
  int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
  for(int i=0;i<pr;i+=2){
    int xs=flower[b][i],xns=flower[b][i+1];
    pa[xs]=g[xns][xs].u;
    S[xs]=1,S[xns]=0;
    slack[xs]=0,set_slack(xns);
    q_push(xns);
  }
  S[xr]=1,pa[xr]=pa[b];
  for(size_t i=pr+1;i<flower[b].size();++i){
    int xs=flower[b][i];
    S[xs]=-1,set_slack(xs);
  }
  st[b]=0;
}
inline bool on_found_edge(const edge &e){
  int u=st[e.u],v=st[e.v];
  if(S[v]==-1){
    pa[v]=e.u,S[v]=1;
    int nu=st[match[v]];
    slack[v]=slack[nu]=0;
    S[nu]=0,q_push(nu);
  }else if(S[v]==0){
    int lca=get_lca(u,v);
    if(!lca)return augment(u,v),augment(v,u),true;
    else add_blossom(u,lca,v);
  }
  return false;
}
inline bool matching(){
  memset(S+1,-1,sizeof(int)*n_x);
  memset(slack+1,0,sizeof(int)*n_x);
  q=queue<int>();
  for(int x=1;x<=n_x;++x)
    if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
  if(q.empty())return false;
  for(;;){
    while(q.size()){
      int u=q.front();q.pop();
      if(S[st[u]]==1)continue;
      for(int v=1;v<=n;++v)
        if(g[u][v].w>0&&st[u]!=st[v]){
          if(e_delta(g[u][v])==0){
            if(on_found_edge(g[u][v]))return true;
          }else update_slack(u,st[v]);
        }
    }
    int d=INF;
    for(int b=n+1;b<=n_x;++b)
      if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
    for(int x=1;x<=n_x;++x)
      if(st[x]==x&&slack[x]){
        if(S[x]==-1)d=min(d,e_delta(g[slack[x]][x]));
        else if(S[x]==0)d=min(d,e_delta(g[slack[x]][x])/2);
      }
    for(int u=1;u<=n;++u){
      if(S[st[u]]==0){
        if(lab[u]<=d)return 0;
        lab[u]-=d;
      }else if(S[st[u]]==1)lab[u]+=d;
    }
    for(int b=n+1;b<=n_x;++b)
      if(st[b]==b){
        if(S[st[b]]==0)lab[b]+=d*2;
        else if(S[st[b]]==1)lab[b]-=d*2;
      }
    q=queue<int>();
    for(int x=1;x<=n_x;++x)
      if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&e_delta(g[slack[x]][x])==0)
        if(on_found_edge(g[slack[x]][x]))return true;
    for(int b=n+1;b<=n_x;++b)
      if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
  }
  return false;
}
inline pair<long long,int> weight_blossom(){
  memset(match+1,0,sizeof(int)*n);
  n_x=n;
  int n_matches=0;
  long long tot_weight=0;
  for(int u=0;u<=n;++u)st[u]=u,flower[u].clear();
  int w_max=0;
  for(int u=1;u<=n;++u)
    for(int v=1;v<=n;++v){
      flower_from[u][v]=(u==v?u:0);
      w_max=max(w_max,g[u][v].w);
    }
  for(int u=1;u<=n;++u)lab[u]=w_max;
  while(matching())++n_matches;
  for(int u=1;u<=n;++u)
    if(match[u]&&match[u]<u)
      tot_weight+=g[u][match[u]].w;
  return make_pair(tot_weight,n_matches);
}
inline void init_weight_graph(){
  for(int u=1;u<=n;++u)
    for(int v=1;v<=n;++v)
      g[u][v]=edge(u,v,0);
}
int main(){
  int m;
  scanf("%d%d",&n,&m);
  init_weight_graph();
  for(int i=0;i<m;++i){
    int u,v,w;
    scanf("%d%d%d",&u,&v,&w);
    g[u][v].w=g[v][u].w=w;
  }
  printf("%lld\n",weight_blossom().first);
  for(int u=1;u<=n;++u)printf("%d ",match[u]);puts("");
  return 0;
}
```

5.9 Stable Marriage

```
#define F(n) Fi(i, n)
#define Fi(i, n) Fl(i, 0, n)
#define Fl(i, l, n) for(int i = l ; i < n ; ++i)
#include <bits/stdc++.h>
using namespace std;
int D, quota[205], weight[205][5];
int S, scoretodep[12005][205], score[5];
int P, prefer[12005][85], iter[12005];
int ans[12005];
typedef pair<int, int> PII;
map<int, int> samescore[205];
typedef priority_queue<PII, vector<PII>, greater<PII>> QQQ;
QQQ pri[205];
void check(int d) {
  PII t = pri[d].top();
  int v;
  if (pri[d].size() - samescore[d][t.first] + 1 <= quota[d]) return;
  while (pri[d].top().first == t.first) {
    v = pri[d].top().second;
    ans[v] = -1;
    --samescore[d][t.first];
    pri[d].pop();
  }
}
void push(int s, int d) {
  if (pri[d].size() < quota[d]) {
    pri[d].push(PII(scoretodep[s][d], s));
    ans[s] = d;
    ++samescore[s][scoretodep[s][d]];
  } else if (scoretodep[s][d] >= pri[d].top().first) {
    pri[d].push(PII(scoretodep[s][d], s));
```

```cpp
    ans[s] = d;
    ++samescore[s][scoretodep[s][d]];
    check(d);
  }
}
void f() {
  int over;
  while (true) {
    over = 1;
    Fi (q, S) {
      if (ans[q] != -1 || iter[q] >= P) continue;
      push(q, prefer[q][iter[q]++]);
      over = 0;
    }
    if (over) break;
  }
}
main() {
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  int sadmit, stof, dexceed, dfew;
  while (cin >> D, D) { // Beware of the input format or judge may troll us.
    sadmit = stof = dexceed = dfew = 0;
    memset(iter, 0, sizeof(iter));
    memset(ans, 0, sizeof(ans));
    Fi (q, 205) {
      pri[q] = QQQ();
      samescore[q].clear();
    }
    cin >> S >> P;
    Fi (q, D) {
      cin >> quota[q];
      Fi (w, 5) cin >> weight[q][w];
    }
    Fi (q, S) {
      Fi (w, 5) cin >> score[w];
      Fi (w, D) {
        scoretodep[q][w] = 0;
        F (5) scoretodep[q][w] += weight[w][i] * score[i];
      }
    }
    Fi (q, S) Fi (w, P) {
      cin >> prefer[q][w];
      --prefer[q][w];
    }
    f();
    Fi (q, D) sadmit += pri[q].size();
    Fi (q, S) if (ans[q] == prefer[q][0]) ++stof;
    Fi (q, D) if (pri[q].size() > quota[q]) ++dexceed;
    Fi (q, D) if (pri[q].size() < quota[q]) ++dfew;
    cout << sadmit << ' ' << stof << ' ' << dexceed << ' ' << dfew << '\n';
  }
}
```

5.10 BCCvertex

```cpp
const int MXN = 16004;
struct BccVertex {
    int n,nScc,step,dfn[MXN],low[MXN];
    vector<int> E[MXN],sccv[MXN];
    int top,stk[MXN];
    void init(int _n) {
        n = _n;
        nScc = step = 0;
        for (int i=0; i<n; i++) E[i].clear();
    }
    void add_edge(int u, int v) {
        E[u].pb(v);
        E[v].pb(u);
    }
    void DFS(int u, int f) {
        dfn[u] = low[u] = step++;
        stk[top++] = u;
        for (auto v:E[u]) {
            if (v == f) continue;
            if (dfn[v] == -1) {
                DFS(v,u);
                low[u] = min(low[u], low[v]);
                if (low[v] >= dfn[u]) {
                    int z;
                    sccv[nScc].clear();
                    do {
                        z = stk[--top];
                        sccv[nScc].pb(z);
                    } while (z != v);
                    sccv[nScc].pb(u);
                    nScc++;
                }
            } else {
                low[u] = min(low[u],dfn[v]);
            }
        }
    }
    vector<vector<int>> solve() {
        vector<vector<int>> res;
        for (int i=0; i<n; i++) {
            dfn[i] = low[i] = -1;
        }
        for (int i=0; i<n; i++) {
            if (dfn[i] == -1) {
                top = 0;
                DFS(i,i);
            }
        }
        for(int i=0;i<nScc;i++) res.pb(sccv[i]);
        return res;
    }
}graph;
```

5.11 MaxClique

```cpp
struct MaxClique{ // 0-base
  typedef bitset< N > Int;
  Int linkto[ N ] , v[ N ];
  int n;
  void init( int _n ){
    n = _n;
    for( int i = 0 ; i < n ; i ++ ){
      linkto[ i ].reset();
      v[ i ].reset();
    }
  }
  void addEdge( int a , int b ){
    v[ a ][ b ] = v[ b ][ a ] = 1;
  }
  int popcount(const Int& val)
  { return val.count(); }
  int lowbit(const Int& val)
  { return val._Find_first(); }
  int ans , stk[ N ];
  int id[ N ] , di[ N ] , deg[ N ];
  Int cans;
  void maxclique(int elem_num, Int candi){
    if(elem_num > ans){
      ans = elem_num;
      cans.reset();
      for( int i = 0 ; i < elem_num ; i ++ )
        cans[ id[ stk[ i ] ] ] = 1;
    }
    int potential = elem_num + popcount(candi);
    if(potential <= ans) return;
    int pivot = lowbit(candi);
    Int smaller_candi = candi & (~linkto[pivot]);
    while(smaller_candi.count() && potential > ans){
      int next = lowbit(smaller_candi);
      candi[next] = !candi[next];
      smaller_candi[ next ] = !smaller_candi[ next ];
      potential --;
      if(next == pivot || (smaller_candi & linkto[next]).count() ){
        stk[elem_num] = next;
        maxclique(elem_num + 1, candi & linkto[next]);
      }
    }
  }
  int solve(){
    for( int i = 0 ; i < n ; i ++ ){
      id[ i ] = i;
      deg[ i ] = v[ i ].count();
    }
    sort( id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; } );
    for( int i = 0 ; i < n ; i ++ )
      di[ id[ i ] ] = i;
    for( int i = 0 ; i < n ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        if( v[ i ][ j ] )
          linkto[ di[ i ] ][ di[ j ] ] = 1;
    Int cand; cand.reset();
    for( int i = 0 ; i < n ; i ++ )
      cand[ i ] = 1;
    ans = 1;
    cans.reset(); cans[ 0 ] = 1;
    maxclique(0, cand);
    return ans;
  }
} g;
```

5.12 BCCedge

```cpp
vector<vector<int> > v;
int vis[100005],lwn[100005];
vector<int> stk;
int f[100005];
int bln[100005];
int Find(int a){
  if(bln[a]==a)return a;
  return bln[a]=Find(bln[a]);
}
int t;
void dfs(int a,int p){
  stk.pb(a);
  bln[a]=a;
  vis[a]=lwn[a]=++t;
  int cnt=0;
  for(int i=0;i<v[a].size();i++){
    int x=v[a][i];
    if(x!=p||cnt==1){
      if(vis[x]==0){
        dfs(x,a);
        if(lwn[x]>vis[a]){
          int fa=Find(x);
          f[x]=Find(a);
          while(stk.back()!=x){
            bln[stk.back()]=fa;
            stk.pop_back();
          }
          bln[stk.back()]=fa;
          stk.pop_back();
        }
        lwn[a]=min(lwn[a],lwn[x]);
      }
      else{
        lwn[a]=min(lwn[a],vis[x]);
      }
    }
    else{
      cnt++;
    }
  }
}
```

5.13 MinimumSteinerTree

```
// Minimum Steiner Tree
// O(V 3^T + V^2 2^T)
struct SteinerTree{
#define V 33
#define T 8
#define INF 1023456789
  int n , dst[V][V] , dp[1 << T][V] , tdst[V];
  void init( int _n ){
    n = _n;
    for( int i = 0 ; i < n ; i ++ ){
      for( int j = 0 ; j < n ; j ++ )
        dst[ i ][ j ] = INF;
      dst[ i ][ i ] = 0;
    }
  }
  void add_edge( int ui , int vi , int wi ){
    dst[ ui ][ vi ] = min( dst[ ui ][ vi ] , wi );
    dst[ vi ][ ui ] = min( dst[ vi ][ ui ] , wi );
  }
  void shortest_path(){
    for( int k = 0 ; k < n ; k ++ )
      for( int i = 0 ; i < n ; i ++ )
        for( int j = 0 ; j < n ; j ++ )
          dst[ i ][ j ] = min( dst[ i ][ j ],
                dst[ i ][ k ] + dst[ k ][ j ] );
  }
  int solve( const vector<int>& ter ){
    int t = (int)ter.size();
    for( int i = 0 ; i < ( 1 << t ) ; i ++ )
      for( int j = 0 ; j < n ; j ++ )
        dp[ i ][ j ] = INF;
    for( int i = 0 ; i < n ; i ++ )
      dp[ 0 ][ i ] = 0;
    for( int msk = 1 ; msk < ( 1 << t ) ; msk ++ ){
      if( msk == ( msk & (-msk) ) ){
        int who = __lg( msk );
        for( int i = 0 ; i < n ; i ++ )
          dp[ msk ][ i ] = dst[ ter[ who ] ][ i ];
        continue;
      }
      for( int i = 0 ; i < n ; i ++ )
        for( int submsk = ( msk - 1 ) & msk ; submsk ;
                 submsk = ( submsk - 1 ) & msk )
            dp[ msk ][ i ] = min( dp[ msk ][ i ],
                            dp[ submsk ][ i ] +
                            dp[ msk ^ submsk ][ i ] );
      for( int i = 0 ; i < n ; i ++ ){
        tdst[ i ] = INF;
        for( int j = 0 ; j < n ; j ++ )
          tdst[ i ] = min( tdst[ i ],
                     dp[ msk ][ j ] + dst[ j ][ i ] );
      }
      for( int i = 0 ; i < n ; i ++ )
        dp[ msk ][ i ] = tdst[ i ];
    }
    int ans = INF;
    for( int i = 0 ; i < n ; i ++ )
      ans = min( ans , dp[ ( 1 << t ) - 1 ][ i ] );
    return ans;
  }
} solver;
```

6 JAVAAndPy

6.1 Big Integer

```
import java.math.*;
import java.io.*;
import java.util.*;
public class Main{
    public static void main(String []argv){
        c[0][0]=BigInteger.ONE;
        for(int i=1;i<3001;i++){
            c[i][0]=BigInteger.ONE;
            c[i][i]=BigInteger.ONE;
            for(int j=1;j<i;j++)c[i][j]=c[i-1][j].add(c[i-1][j-1]);
        }
        Scanner scanner = new Scanner(System.in);
        int T = scanner.nextInt();
        BigInteger x;
        BigInteger ans;
        while(T-- > 0){
            ans = BigInteger.ZERO;
            int n = scanner.nextInt();
            for(int i=0;i<n;i++){
                x = new BigInteger(scanner.next());
                if(i%2 == 1)ans=ans.subtract(c[n-1][i].multiply(x));
                else ans=ans.add(c[n-1][i].multiply(x));
            }
            if(n%2 == 0)ans=BigInteger.ZERO.subtract(ans);
            System.out.println(ans);
        }
    }
}
```

6.2 Fraction Limit

```
from fractions import Fraction
Fraction.limit_denominator(max_denominator=1000)
```

7 Other

7.1 Fraction Binary

```
pair<pll,pll> fraction_b(int n,int m){
    pll Max=mp(1,0),Min=mp(0,1);
    int Big=0;
    while(true){
        if(Max.x+Min.x>n||Max.y+Min.y>m)break;
        if(Big){
            LL large;
            for(large = 1;;large<<=1){
                pll p=mp(Max.x*large+Min.x,Max.y*large+Min.y);
                if(cal(p,n,m)>=x)break;
                if(p.x>n||p.y>m){
                    large>>=1;
                    break;
                }
            }
            int add=0;
            for(;large;large>>=1){
                pll p=mp(Max.x*(add+large)+Min.x,Max.y*(add+large)+Min.y);
                if(cal(p,n,m)<x&&p.x<=n&&p.y<=m)add+=large;
            }
            Min=mp(Max.x*add+Min.x,Max.y*add+Min.y);
            if(Max.x+Min.x<=n&&Max.y+Min.y<=m)
            Max=mp(Max.x+Min.x,Max.y+Min.y);
        }
        else{
            int large;
            for(large = 1;;large<<=1){
                pll p = mp(Max.x+Min.x*large,Max.y+Min.y*large);
                if(cal(p,n,m)<x)break;
                    if(p.x>n||p.y>m){
                    large>>=1;
                    break;
                }
            }
            int add=0;
            for(;large;large>>=1){
                pll p =mp(Min.x*(add+large)+Max.x,Min.y*(add+large)+Max.y);
                if(cal(p,n,m)>=x&&p.x<=n&&p.y<=m)add+=large;

            }
            Max=mp(Min.x*add+Max.x,Min.y*add+Max.y);
            if(Max.x+Min.x<=n&&Max.y+Min.y<=m)
            Min=mp(Max.x+Min.x,Max.y+Min.y);
        }
        Big^=1;
    }
    return mp(Min,Max);
}
```

7.2 Annealing

```
double distForAllPoints(double x, double y,
            vector< pair<int, int> > &D) {
  double sum = 0;
  for(int i = D.size()-1; i >= 0; i--) {
    sum += hypot(D[i].first - x, D[i].second - y);
  }
  return sum;
}
double randDouble() {
  return (rand() % 32767) / 32767.0;
}
double annealing(vector< pair<int, int> > &D) {
#define S_MUL 0.6f
#define S_LEN 1000
#define T_CNT 10
#define E_CNT 10
  double step = S_LEN;
  double x[E_CNT], y[E_CNT], val[E_CNT];
  double Lx, Ly, Rx, Ry, tx, ty, tcost;
  Lx = Rx = D[0].first;
  Ly = Ry = D[0].second;
  for(int i = 0; i < D.size(); i++) {
    Lx = min(Lx, (double)D[i].first);
    Rx = max(Rx, (double)D[i].first);
    Ly = min(Ly, (double)D[i].second);
    Ry = max(Ry, (double)D[i].second);
  }
  for(int i = 0; i < E_CNT; i++) {
    x[i] = randDouble() * (Rx - Lx) + Lx;
    y[i] = randDouble() * (Ry - Ly) + Ly;
    val[i] = distForAllPoints(x[i], y[i], D);
  }
  while(step > 0.1) {
    for(int i = 0; i < E_CNT; i++) {
      for(int j = 0; j < T_CNT; j++) {
        tx = x[i] + randDouble() * 2 * step - step;
        ty = y[i] + randDouble() * 2 * step - step;
        tcost = distForAllPoints(tx, ty, D);
        if(tcost < val[i]) {
          val[i] = tcost, x[i] = tx, y[i] = ty;
        }
      }
    }
    step *= S_MUL;
  }
  double ret = val[0];
  for(int i = 0; i < E_CNT; i++) {
    ret = min(ret, val[i]);
  }
  printf("%.0lf\n", ret);
}
int main() {
  int testcase, N;
  scanf("%d", &testcase);
  while(testcase--) {
    scanf("%d", &N);
    vector< pair<int, int> > D;
    int x, y;
    for(int i = 0; i < N; i++) {
      scanf("%d %d", &x, &y);
      D.push_back(make_pair(x, y));
    }
    annealing(D);
    if(testcase)
      puts("");
  }
  return 0;
}
```

7.3 MahattanMST

```
#include<bits/stdc++.h>
#define REP(i,n) for(int i=0;i<n;i++)
using namespace std;
typedef long long LL;
const int N=200100;
int n,m;
struct PT {int x,y,z,w,id;}p[N];
inline int dis(const PT &a,const PT &b){return abs(a.xb.x)+abs(a.y-b.y);}
inline bool cpx(const PT &a,const PT &b){return a.x!=b.
x? a.x>b.x:a.y>b.y;}
inline bool cpz(const PT &a,const PT &b){return a.z<b.z
;}
struct E{int a,b,c;}e[8*N];
bool operator<(const E&a,const E&b){return a.c<b.c;}
struct Node{
    int L,R,key;
}node[4*N];
int s[N];
int F(int x){return s[x]==x?x:s[x]=F(s[x]);}
void U(int a,int b){s[F(b)]=F(a);}
void init(int id,int L,int R) {
    node[id]=(Node){L,R,-1};
    if(L==R)return
    ;
    init(id*2,L,(L+R)/2);
    init(id*2+1,(L+R)/2+1,R);
}
void ins(int id,int x) {
    if(node[id].key==-1 || p[node[id].key].w>p[x].w)node[
    id].key=x;
    if(node[id].L==node[id].R)return
    ;
    if(p[x].z<=(node[id].L+node[id].R)/2)ins(id*2,x);
    else ins(id*2+1,x);
}
int Q(int id,int L,int R){
    if(R<node[id].L || L>node[id].R)return -1;
    if(L<=node[id].L && node[id].R<=R)return node[id].key ;
    int a=Q(id*2,L,R),b=Q(id*2+1,L,R);
    if(b==-1 || (a!=-1 && p[a].w<p[b].w)) return a;
    else return b;
}
void calc() {
    REP(i,n) {
        p[i].z=p[i].y-p[i].x;
        p[i].w=p[i].x+p[i].y;
    }
    sort(p,p+n,cpz);
    int cnt=0,j,k;
    for
    (int i=0;i<n;i=j){
        for(j=i+1;p[j].z==p[i].z && j<n;j++);
        for(k=i,cnt++;k<j;k++)p[k].z=cnt;
    }
    init(1,1,cnt);
    sort(p,p+n,cpx);
    REP(i,n) {
        j=Q(1,p[i].z,cnt);
        if(j!=-1)e[m++]=(E){p[i].id,p[j].id,dis(p[i],p[j])
        };
        ins(1,i);
    }
}
LL MST() {
    LL r=0;
    sort(e,e+m);
    REP(i,m) {
        if(F(e[i].a)==F(e[i].b))continue;
        U(e[i].a,e[i].b);
        r+=e[i].c;
    }
    return r;
}
int main(){
    int ts;
    scanf("%d", &ts);
    while (ts--) {
        m = 0;
        scanf("%d",&n);
        REP(i,n) {scanf("%d%d",&p[i].x,&p[i].y);p[i].id=s[i]=i;}
        calc();
        REP(i,n)p[i].y= -p[i].y;
        calc();
        REP(i,n)swap(p[i].x,p[i].y);
        calc();
        REP(i,n)p[i].x=-p[i].x;
        calc();
        printf("%lld\n",MST()*2);
    }
    return 0;
}
```

7.4 Dynamic Convex

```
struct LiChao_min{
  struct line{
    LL m, c;
    line(LL _m=0, LL _c=0) { m = _m; c = _c; }
    LL eval(LL x) { return m * x + c; }
  };
  struct node{
    node *l, *r; line f;
    node(line v) { f = v; l = r = NULL; }
  };
  typedef node* pnode;
  pnode root; int sz;
#define mid ((l+r)>>1)
  void insert(line &v, int l, int r, pnode &nd){
    if(!nd) { nd = new node(v); return; }
    LL trl = nd->f.eval(l), trr = nd->f.eval(r);
    LL vl = v.eval(l), vr = v.eval(r);
    if(trl <= vl && trr <= vr) return;
    if(trl > vl && trr > vr) { nd->f = v; return; }
    if(trl > vl) swap(nd->f, v);
    if(nd->f.eval(mid) < v.eval(mid)) insert(v, mid + 1, r, nd->r);
    else swap(nd->f, v), insert(v, l, mid, nd->l);
  }
  LL query(int x, int l, int r, pnode &nd){
    if(!nd) return LLONG_MAX;
    if(l == r) return nd->f.eval(x);
    if(mid >= x) return min(nd->f.eval(x), query(x, l, mid, nd->l));
    return min(nd->f.eval(x), query(x, mid + 1, r, nd->r));
  }
  /* -sz <= query_x <= sz */
  void init(int _sz){ sz = _sz + 1; root = NULL; }
  void add_line(LL m, LL c){ line v(m, c); insert(v, -sz, sz, root); }
  LL query(LL x) { return query(x, -sz, sz, root); }
};
```

7.5 Dp Optimizer

```
list<int> mylist;
vector<list<int>::iterator> v;
vector<int> vis;
int search(list<int>::iterator x,int i,int n){
    int Max=n+1,Min=*x;
    list<int>::iterator last=x;
    last--;
    while(Max>Min+1){
        int mid=(Max+Min)/2;
        int a=*last,b=*x;
        int val1=dp[a][i-1]+cost[a+1][mid],val2=dp[b][i-1]+cost[b+1][mid];
        if(val1>=val2)Max=mid;
        else Min=mid;
    }
    return Max;
}
priority_queue<pii,vector<pii>,greater<pii> > pq;
for(int i=2;i<=m;i++){
    mylist.clear();mylist.pb(i-1);
    v.clear();v.resize(n);
    vis.clear();vis.resize(n+1,0);
    for(int j=i;j<=n;j++){
        while(!pq.empty()&&pq.top().x<=j){
            pii p=pq.top();pq.pop();
            if(vis[p.y])continue;
            auto it=v[p.y];it--;
            vis[*it]=1;mylist.erase(it);
            if(v[p.y]!=mylist.begin())
                pq.push(mp(search(v[p.y],i,n),p.y));
        }
        int opt=mylist.front();
        dp[j][i]=dp[opt][i-1]+cost[opt+1][j];
        mylist.push_back(j);
        v[j]=mylist.end();v[j]--;
        pq.push(mp(search(v[j],i,n),j));
    }
    while(!pq.empty())pq.pop();
}
```

7.6 Det

```
LL det(LL a[][20],int n){
    LL ret = 1;
    for(int i = 1 ; i < n ; i++){
        for(int j = i + 1 ; j < n ; j++){
            while(a[j][i]){
                LL t = a[i][i] / a[j][i];
                for(int k = i ; k < n ; k++)
                    a[i][k] = a[i][k] - a[j][k] * t;
                for(int k = i ; k < n ; k++)
                    swap(a[i][k], a[j][k]);
                ret = -ret;
            }
        }
        if(a[i][i] == 0)return 0;
        ret = ret * a[i][i];
    }
    return ret;
}
```

8 String

8.1 AC

```
struct Node{
  Node *index[30];
  Node *fail;
  int word;
  int num;
  Node(){
    for(int i=0;i<30;i++)
    index[i]=NULL;
    fail=NULL;
    word=0;
    num=-1;
  }
}*root=new Node();
void add(char c[]){
  Node *n=root;
  for(int i=0;c[i]!=0;i++){

    if(!n->index[c[i]-'a'])
    n->index[c[i]-'a']=new Node();
    n=n->index[c[i]-'a'];
  }
  n->word=1;
  n->num=t++;
}
void ac(){
  queue<Node*> q;
  q.push(root);
  root->fail=NULL;
  while(!q.empty()){
    Node *n=q.front();
    q.pop();
    for(int i=0;i<30;i++){
      if(n->index[i]){
        q.push(n->index[i]);
        Node* p=n->fail;
        while(p!=NULL&&!p->index[i])
        p=p->fail;
        if(p)
        n->index[i]->fail=p->index[i];
        else
        n->index[i]->fail=root;
      }
    }
  }
}
void search(char c[]){
  Node *n=root;
  for(int i=0;c[i]!=0;i++){

    while(!n->index[c[i]-'a']&&n!=root){
      n=n->fail;
    }
    if(n->index[c[i]-'a'])
    n=n->index[c[i]-'a'];
    Node *p=n;
    while(p){
      if(p->num!=-1)
      {
        ans[p->num]++;
      }
      p=p->fail;
    }
  }
}
void del(Node *n=root){
  for(int i=0;i<30;i++)
  if(n->index[i])
  del(n->index[i]);
  free(n);
}
```

8.2 SuffixAutomata

```
// BZOJ 3998
const int MAX_N = 500000 + 10;
struct Node {
    static Node mem[MAX_N<<1] , *pmem;
    Node *ch[26] , *fail;
    int mx , val;
    ll dp;
    int tag , deg;
    Node():mx(0),fail(0),dp(0),val(0),tag(0),deg(0){
        MS(ch , 0);
    }
}
Node::mem[MAX_N<<1] , *Node::pmem = Node::mem , *root
, *last;
int T , N;
char s[MAX_N];
inline void init() {
    last = root = new (Node::pmem++)Node();
}
inline int idx(char c) {
    return c -'a';
}
inline void insert(char c) {
    c = idx(c);
    Node *p = last;
    Node *np = new (Node::pmem++)Node();
    np->mx = p->mx + 1;
    np->val = 1;
    while(p && !p->ch[c]) {
        p->ch[c] = np;
        np->deg++;
        p = p->fail;
    }
    if(!p) np->fail = root;
    else
    {
        Node *q = p->ch[c];
        if(q->mx == p->mx + 1) np->fail = q;
        else
        {
            Node *nq = new (Node::pmem++)Node();
            nq->mx = p->mx + 1;
            nq->val = 0;
            memcpy(nq->ch , q->ch , sizeof(q->ch));
            REP(i , 26) {
                if(nq->ch[i]) nq->ch[i]->deg++;
            }
            nq->fail = q->fail;
            q->fail = np->fail = nq;
            while(p && p->ch[c] == q) {
                p->ch[c] = nq;
                q->deg--;
                nq->deg++;
                p = p->fail;
            }
        }
    }
    last = np;
}
inline void bfs() {
    static Node* que[MAX_N<<1];
    int l = 0 , r = 0;
    que[r++] = root;
    root->tag = 2;
    vector<Node*> vec;
    while(l < r) {
        Node *u = que[l++];
        REP(i , 26) {
            if(u->ch[i]) {
                if(--u->ch[i]->deg == 0 && u->ch[i]->
                tag != 1) {
                    u->ch[i]->tag = 1;
                    que[r++] = u->ch[i];
                    vec.PB(u->ch[i]);
                }
            }
        }
    }
    for(int i = SZ(vec) - 1; i >= 0; i--) {
        Node *u = vec[i];
        if(T) {
            if(u->fail) u->fail->val += u->val;
        }
        else u->val = 1;
    }
    root->val = 0;
    for(int i = SZ(vec) - 1; i >= 0; i--) {
        Node *u = vec[i];
        u->dp = u->val;
        REP(j , 26) {
            if(u->ch[j]) u->dp += u->ch[j]->dp;
        }
    }
    REP(i , 26) {
        if(root->ch[i]) root->dp += root->ch[i]->dp;
    }
}
inline void solve(int k) {
    Node *p = root;
    if(k > p->dp || k <= 0) {
        puts("-1");
        return;
    }
    while(k > 0) {
        int flag = 0;
        REP(i , 26) {
            if(!p->ch[i]) continue;
            if(k <= p->ch[i]->dp) {
                putchar('a' + i);
                k -= p->ch[i]->val;
                p = p->ch[i];
                flag = 1;
                break
                ;
            }
            else k -= p->ch[i]->dp;
        }
        if(!flag) break;
    }
}
int main() {
    scanf("%s",s);
    int n = strlen(s);
    N = n;
    init();
    REP(i , n) insert(s[i]);
    int K;
    scanf("%d%d",&T,&K);
    bfs();
    solve(K);
    return 0;
}
```

8.3 Palindromic Tree

```
#include<bits/stdc++.h>
#include<unistd.h>
using namespace std;
#define F first
#define S second
#define MP make_pair
#define PB push_back
#define IOS ios_base::sync_with_stdio(0); cin.tie(0);
#define SZ(x) ((int)((x).size()))
#define ALL(x) begin(x),end(x)
#define REP(i,x) for (int i=0; i<(x); i++)
#define REP1(i,a,b) for (int i=(a); i<=(b); i++)

struct palindromic_tree{
  struct node{
    int next[26],fail,len;
    int cnt,num,st,ed;
    node(int l=0):fail(0),len(l),cnt(0),num(0){
      for(int i=0;i<26;++i)next[i]=0;
    }
  };
  vector<node> state;
  vector<char> s;
  int last,n;

  void init(){
    state.clear();
    s.clear();
    last=1;
    n=0;
    state.push_back(0);
    state.push_back(-1);
    state[0].fail=1;
    s.push_back(-1);
  }
  int get_fail(int x){
    while(s[n-state[x].len-1]!=s[n])x=state[x].fail;
    return x;
  }
  void add(int c){
    s.push_back(c-='a');
    ++n;
    int cur=get_fail(last);
    if(!state[cur].next[c]){
      int now=state.size();
      state.push_back(state[cur].len+2);
      state[now].fail=state[get_fail(state[cur].fail)].next[c];
      state[cur].next[c]=now;
      state[now].num=state[state[now].fail].num+1;
    }
    last=state[cur].next[c];
    ++state[last].cnt;
  }
  int size(){
    return state.size()-2;
  }
}pt;

int main() {
  string s;
  cin >> s;
  pt.init();
  for (int i=0; i<SZ(s); i++) {
    int prvsz = pt.size();
    pt.add(s[i]);
    if (prvsz != pt.size()) {
      int r = i;
      int l = r - pt.state[pt.last].len + 1;
      cout << "Find pal @ [" << l << " " << r << "] : " << s.substr(l,r-l+1) << endl;
    }
  }

  return 0;
}
```

8.4 MinLexicographicalRotate

```
string min_cyclic_string(string s) {
    s += s;
    int n = s.size();
    int i = 0, ans = 0;
    while (i < n / 2) {
        ans = i;
        int j = i + 1, k = i;
        while (j < n && s[k] <= s[j]) {
            if (s[k] < s[j]) k = i;
            else k++;
            j++;
        }
        while (i <= k) i += j - k;
    }
    return s.substr(ans, n / 2);
}
```

8.5 CLCS

```
#define L 0
#define LU 1
#define U 2
const int mov[3][2]={0,-1, -1,-1, -1,0};
int al,bl;
char a[MAXL*2],b[MAXL*2]; // 0-indexed
int dp[MAXL*2][MAXL];
```

```
char pred[MAXL*2][MAXL];
inline int lcs_length(int r) {
  int i=r+al,j=bl,l=0;
  while(i>r) {
    char dir=pred[i][j];
    if(dir==LU) l++;
    i+=mov[dir][0];
    j+=mov[dir][1];
  }
  return l;
}
inline void reroot(int r) { // r = new base row
  int i=r,j=1;
  while(j<=bl&&pred[i][j]!=LU) j++;
  if(j>bl) return;
  pred[i][j]=L;
  while(i<2*al&&j<=bl) {
    if(pred[i+1][j]==U) {
      i++;
      pred[i][j]=L;
    } else if(j<bl&&pred[i+1][j+1]==LU) {
      i++;
      j++;
      pred[i][j]=L;
    } else {
      j++;
    }
  }
}
int cyclic_lcs() {
  // a, b, al, bl should be properly filled
  // note: a WILL be altered in process
  //         -- concatenated after itself
  char tmp[MAXL];
  if(al>bl) {
    swap(al,bl);
    strcpy(tmp,a);
    strcpy(a,b);
    strcpy(b,tmp);
  }
  strcpy(tmp,a);
  strcat(a,tmp);
  // basic lcs
  for(int i=0;i<=2*al;i++) {
    dp[i][0]=0;
    pred[i][0]=U;
  }
  for(int j=0;j<=bl;j++) {
    dp[0][j]=0;
    pred[0][j]=L;
  }
  for(int i=1;i<=2*al;i++) {
    for(int j=1;j<=bl;j++) {
      if(a[i-1]==b[j-1]) dp[i][j]=dp[i-1][j-1]+1;
      else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
      if(dp[i][j-1]==dp[i][j]) pred[i][j]=L;
      else if(a[i-1]==b[j-1]) pred[i][j]=LU;
      else pred[i][j]=U;
    }
  }
  // do cyclic lcs
  int clcs=0;
  for(int i=0;i<al;i++) {
    clcs=max(clcs,lcs_length(i));
    reroot(i+1);
  }
  // recover a
  a[al]='\0';
  return clcs;
}
```

8.6 ZvaluePalindromes

```
inline void manacher(char *s,int len,int *z){
  int l=0,r=0;
  for(int i=1;i<len;++i){
    z[i]=r>i?min(z[2*l-i],r-i):1;
    while(s[i+z[i]]==s[i-z[i]])++z[i];
    if(z[i]+i>r)r=z[i]+i,l=i;
  }
}
```

8.7 SuffixArray

```
int ss[N];
int heigh[N];
int sa[N];
int rank[N];
int length;
int val[30];
int c[N];   // counting sort array
int temp[2][N];
void suffix_array(){
    int A = 250;
    int* rank = temp[0];
    int* new_rank = temp[1];
    for (int i=0; i<A; ++i) c[i] = 0;
    for (int i=0; i<length; ++i) c[rank[i] = ss[i]]++;
    for (int i=1; i<A; ++i) c[i] += c[i-1];
    for (int i=length-1; i>=0; --i) sa[--c[ss[i]]] = i;
    for (int n=1; n<length; n*=2){
        for (int i=0; i<A; ++i) c[i] = 0;
        for (int i=0; i<length; ++i) c[rank[i]]++;
        for (int i=1; i<A; ++i) c[i] += c[i-1];
        int* sa2 = new_rank;
        int r = 0;
        for (int i=length-n; i<length; ++i)
            sa2[r++] = i;
        for (int i=0; i<length; ++i)
            if (sa[i] >= n)
                sa2[r++] = sa[i] - n;
        for (int i=length-1; i>=0; --i)
            sa[--c[rank[sa2[i]]]] = sa2[i];
        new_rank[sa[0]] = r = 0;
        for (int i=1; i<length; ++i){
            if (!(rank[sa[i-1]] == rank[sa[i]] &&
                sa[i-1]+n < length &&    // stable sort trick
                rank[sa[i-1]+n] == rank[sa[i]+n]))
                r++;
            new_rank[sa[i]] = r;
        }
        swap(rank, new_rank);
        if (r == length-1) break;
        A = r + 1;
    }
}
void lcp_array(){
    for (int i=0; i<length; ++i)
        rank[sa[i]] = i;

    for (int i=0, lcp=0,h=0; i<length; i++){
        if (rank[i] == 0)
            heigh[0] = 0;
        else{
            int j = sa[rank[i]-1];
            if (lcp > 0) lcp-=val[ss[i-1]-'a'],h--;
            while (ss[i+h] == ss[j+h]) lcp+=val[ss[i+h]-'a'],h++;
            heigh[rank[i]] = lcp;
        }
    }
}
```

8.8 Zvalue

```
inline void z_alg1(char *s,int len,int *z){
  int l=0,r=0;
  z[0]=len;
  for(int i=1;i<len;++i){
    z[i]=r>i?min(r-i+1,z[z[l]-(r-i+1)]):0;
    while(i+z[i]<len&&s[z[i]]==s[i+z[i]])++z[i];
    if(i+z[i]-1>r)r=i+z[i]-1,l=i;
  }
}
```

9 Math

9.1 MillerRabin

```
// 4759123141 2, 7, 61
//2^64 2, 325, 9375, 28178, 450775, 9780504, 1795265022
bool Isprime(LL n)
{
    if (n == 2) return true;
    if (n < 2 || n % 2 == 0) return false;
    LL u = n - 1, t = 0;
    while (u % 2 == 0) {u >>= 1; t++;}
    LL sprp[7] = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
    for (int k=0; k<7; ++k)
    {
        LL a = sprp[k] % n;
        if (a == 0 || a == 1 || a == n-1) continue;
        long long x = f_pow(a, u, n);
        if (x == 1 || x == n-1) continue;
        for (int i = 0; i < t-1; i++)
        {
            x = f_pow(x, 2, n);
            if (x == 1) return false;
            if (x == n-1) break;
        }
        if (x == n-1) continue;
        return false;
    }
    return true;
}
```

9.2 Simplex

```
const int maxn = 111;
const int maxm = 111;
const double eps = 1E-10;

double a[maxn][maxm], b[maxn], c[maxm], d[maxn][maxm];
```

```cpp
double x[maxm];
int ix[maxn + maxm]; // !!! array all indexed from 0
// max{cx} subject to {Ax<=b,x>=0}
// n: constraints, m: vars !!!
// x[] is the optimal solution vector
//
// usage :
// value = simplex(a, b, c, N, M);
double simplex(double a[maxn][maxm], double b[maxn], double c[maxm], int n, int m) {
    ++m;
    int r = n, s = m - 1;
    memset(d, 0, sizeof(d));
    for (int i = 0; i < n + m; ++i) ix[i] = i;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m - 1; ++j)
            d[i][j] = -a[i][j];
        d[i][m - 1] = 1;
        d[i][m] = b[i];
        if (d[r][m] > d[i][m]) r = i;
    }
    for (int j = 0; j < m - 1; ++j) d[n][j] = c[j];
    d[n + 1][m - 1] = -1;
    for (double dd;; ) {
        if (r < n) {
            int t = ix[s];
            ix[s] = ix[r + m]; ix[r + m] = t;
            d[r][s] = 1.0 / d[r][s];
            for (int j = 0; j <= m; ++j)
                if (j != s) d[r][j] *= -d[r][s];
            for (int i = 0; i <= n + 1; ++i)
                if (i != r) {
                    for (int j = 0; j <= m; ++j)
                        if (j != s)
                            d[i][j] += d[r][j]*d[i][s];
                    d[i][s] *= d[r][s];
                }
        }
        r = -1; s = -1;
        for (int j = 0; j < m; ++j)
            if (s < 0 || ix[s] > ix[j]) {
                if (d[n + 1][j] > eps || (d[n + 1][j] > -eps && d[n][j] > eps)) s = j;
            }
        if (s < 0) break;
        for (int i=0; i<n; ++i) if (d[i][s] < -eps) {
            if (r < 0 || (dd = d[r][m] / d[r][s] - d[i][m] / d[i][s]) < -eps || (dd < eps && ix[r + m] > ix[i + m])) r = i;
        }
        if (r < 0) return -1; // not bounded
    }
    if (d[n + 1][m] < -eps) return -1; // not executable
    double ans = 0;
    for(int i=0; i<m; i++) x[i] = 0;
    for (int i = m; i < n + m; ++i) { // the missing enumerated x[i] = 0
        if (ix[i] < m - 1)
        {
            ans += d[i - m][m] * c[ix[i]];
            x[ix[i]] = d[i-m][m];
        }
    }
    return ans;
}
```

9.3 Theorem

```
/*
Lucas's Theorem:
For non-negative integer n,m and prime P,
C(m,n) mod P = C(m/P,n/P) * C(m%P,n%P) mod P
-------------------------------------------------------
Pick's Theorem
A = i + b/2 - 1
-------------------------------------------------------
Erdős – Gallai theorem
\sum_{i=1}^k d_i \leq k*(k-1)+\sum_{i=k+1}^n min(d_i,k)
d_i decrease
-------------------------------------------------------
meissel-lehmer
p_1=2
pi(n) = phi(n,m) - P2(n,m) + m - 1 p_m>=n^1/3, pi=prime count
P2(n,m) = \sum_{p_m\lt p \le sqrt(n)}(pi(n/p) - pi(p) + 1)
P2(n,m) = number of x whose has 2 prime factor and greater than p_m
phi(n,m) = phi(n,m-1) - phi(n/p_m,m-1)
phi(n,0) = n
if(n<p_m)phi(n,m)=1
phi = number of x whose prime factor greater than p_m
10^11 4118054813
\item Stirling Numbers(permutation $|P|=n$ with $k$ cycles): \\
  $S(n,k) = \text{coefficient of }x^k \text{ in } \Pi_{i=0}^{n-1} (x+i)$
\item Stirling Numbers(Partition $n$ elements into $k$ non-empty set): \\
  $S(n,k) = \frac{1}{k!} \sum\limits_{j=0}^k (-1)^{k-j} {k \choose j} j^n$
\item Stirling approximation
  $n!=\sqrt{2\pi n}(\frac{n}{e})^n(1+\frac{1}{12n}+\frac{1}{288n^2}-\frac{139}{51840n^3})$
*/
```

9.4 Rombeg

```cpp
// Estimates the definite integral of
// \int_a^b f(x) dx
template<class T>
double romberg( T& f, double a, double b, double eps=1e-8){
  vector<double>t; double h=b-a,last,curr; int k=1,i=1;
  t.push_back(h*(f(a)+f(b))/2);
  do{ last=t.back(); curr=0; double x=a+h/2;
    for(int j=0;j<k;j++) curr+=f(x), x+=h;
    curr=(t[0] + h*curr)/2; double k1=4.0/3.0,k2=1.0/3.0;
    for(int j=0;j<i;j++){ double temp=k1*curr-k2*t[j];
      t[j]=curr; curr=temp; k2/=4*k1-k2; k1=k2+1;
    } t.push_back(curr); k*=2; h/=2; i++;
  }while( fabs(last-curr) > eps);
  return t.back();
}
```

9.5 SchreierSims

```cpp
namespace SchreierSimsAlgorithm{
  typedef vector<int> Permu;
  Permu inv( const Permu& p ){
    Permu ret( p.size() );
    for( int i = 0; i < int(p.size()); i ++ )
      ret[ p[ i ] ] = i;
    return ret;
  }
  Permu operator*( const Permu& a, const Permu& b ){
    Permu ret( a.size() );
    for( int i = 0 ; i < (int)a.size(); i ++ )
      ret[ i ] = b[ a[ i ] ];
    return ret;
  }
  typedef vector<Permu> Bucket;
  typedef vector<int> Table;
  typedef pair<int,int> pii;
  int n, m;
  vector<Bucket> bkts, bktsInv;
  vector<Table> lookup;
  int fastFilter( const Permu &g, bool addToG = 1 ){
    n = bkts.size();
    Permu p=g;
    for( int i = 0 ; i < n ; i ++ ){
      int res = lookup[ i ][ p[ i ] ];
      if( res == -1 ){
        if( addToG ){
          bkts[ i ].push_back( p );
          bktsInv[ i ].push_back( inv( p ) );
          lookup[ i ][ p[i] ] = (int)bkts[i].size()-1;
        }
        return i;
      }
      p = p * bktsInv[i][res];
    }
    return -1;
  }
  long long calcTotalSize(){
    long long ret = 1;
    for( int i = 0 ; i < n ; i ++ )
      ret *= bkts[i].size();
    return ret;
  }
  bool inGroup( const Permu &g ){
    return fastFilter( g, false ) == -1;
  }
  void solve( const Bucket &gen, int _n ){
    n = _n, m = gen.size(); // m perm[0..n-1]s
    {//clear all
      bkts.clear(); bkts.resize(n);
      bktsInv.clear();bktsInv.resize(n);
      lookup.clear();lookup.resize(n);
    }
    for(int i = 0 ; i < n ; i ++ ){
      lookup[i].resize(n);
      fill(lookup[i].begin(), lookup[i].end(), -1);
    }
    Permu id( n );
    for(int i = 0 ; i < n ; i ++ ) id[i] = i;
    for(int i = 0 ; i < n ; i ++ ){
      bkts[i].push_back(id);
      bktsInv[i].push_back(id);
      lookup[i][i] = 0;
    }
    for(int i = 0 ; i < m ; i ++)
      fastFilter( gen[i] );
    queue< pair<pii,pii> > toUpd;
    for(int i = 0; i < n; i ++)
      for(int j = i; j < n; j ++)
        for(int k = 0; k < (int)bkts[i].size(); k ++)
```

```
          for(int l = 0; l < (int)bkts[j].size(); l ++)
            toUpd.push( {pii(i,k), pii(j,l)} );
    while( !toUpd.empty() ){
      pii a = toUpd.front().first;
      pii b = toUpd.front().second;
      toUpd.pop();
      int res = fastFilter(bkts[a.first][a.second] *
                           bkts[b.first][b.second]);
      if(res == -1) continue;
      pii newPair(res, (int)bkts[res].size() - 1);
      for(int i = 0; i < n; i ++)
        for(int j = 0; j < (int)bkts[i].size(); ++j){
          if(i <= res)
            toUpd.push(make_pair(pii(i , j), newPair));
          if(res <= i)
            toUpd.push(make_pair(newPair, pii(i, j)));
        }
    }
  }
}
```

9.6 FFT

```
#define N 524288
#define pi acos(-1)
typedef complex<double> C;
int n,m,i,t,g[N];
C a[N],b[N];
void FFTinit(){
  for (i=1;i<N;i++) g[i]=g[i>>1]>>1|((i&1)<<18);
}
void FFT(C *a,int f){
  int i,j,k,p;
  for (i=0;i<N;i++)
    if (g[i]>i) swap(a[i],a[g[i]]);
  for (i=1;i<N;i<<=1){
    C e(cos(pi/i),f*sin(pi/i));
    for (j=0;j<N;j+=i<<1){
      C w(1,0);
      for (k=0;k<i;k++,w*=e){
        C x=a[j+k],y=w*a[j+k+i];
        a[j+k]=x+y;a[j+k+i]=x-y;
      }
    }
  }
}
int res[400005];
int main(){
  FFTinit();
  FFT(a,1);
  FFT(b,1);
  for(i=0;i<N;i++) a[i]=a[i]*b[i];
  FFT(a,-1);
  for (i=0;i<n+m;i++)
  (int)a[i].real()/N+0.5)
}
```

9.7 NTT

```
int P=998244353,root=3,MAXNUM=1<<23;
// Remember coefficient are mod P
/*
p=a*2^n+1 degree(poly) <= 2^n
n   2^n             p           a     root
16    65536               65537         1     3
20      1048576             7340033       7     3
23                          998244353           119     3
*/
int bigmod(long long a,int b){
  if(b==0)return 1;
  return (bigmod((a*a)%P,b/2)*(b%2?a:1ll))%P;
}
int inv(int a,int b){
  if(a==1)return 1;
  return (((long long)(a-inv(b%a,a))*b+1)/a)%b;
}
std::vector<long long> ps(MAXNUM);
std::vector<int> rev(MAXNUM);
    LL f_pow(unsigned int a,LL b){
        LL res=1,temp=a;
        while(b){
            if(b&1)res=res*temp%P;
            temp=temp*temp%P;
            b>>=1;
        }
        return res;
    }
struct poly{
  std::vector<unsigned int> co;
  int n;//polynomial degree = n
  poly(int d){n=d;co.resize(n+1,0);}
  void ntt(int NN){
    int r=0,st,N;
    unsigned int a,b;
    while((1<<r)<(NN>>1))++r;//inv:r=0
    for(N=2;N<=NN;N<<=1,--r){
        //inv for(N=NN;N>1;N>>=1,r++)
      for(st=0;st<NN;st+=N){
        int i,ss=st+(N>>1);
        for(i=(N>>1)-1;i>=0;--i){
          a=co[st+i]; b=(ps[i<<r]*co[ss+i])%P;
                                //inv b=co[ss+i];
          co[st+i]=a+b; if(co[st+i]>=P)co[st+i]-=P;
          co[ss+i]=a+P-b; if(co[ss+i]>=P)co[ss+i]-=P;
                    //inv co[ss+i]=((a+P-b)*ps[i<<r])%p;
        }
      }
    }
  }
  void ntt_inv(int NN){
  }
    poly operator*(const poly& _b)const{
    poly a=*this,b=_b;
    int k=n+b.n,i,N=1;
    while(N<=k)N*=2;
    a.co.resize(N,0); b.co.resize(N,0);
    int r=bigmod(root,(P-1)/N),Ni=inv(N,P);
    ps[0]=1;
    for(i=1;i<N;++i)ps[i]=(ps[i-1]*r)%P;
    a.ntt(N);b.ntt(N);
    for(i=0;i<N;++i)a.co[i]=((long long)a.co[i]*b.co[i])%P;
    r=inv(r,P);
    for(i=1;i<N/2;++i)std::swap(ps[i],ps[N-i]);
    a.ntt_inv(N);
    for(i=0;i<N;++i)a.co[i]=((long long)a.co[i]*Ni)%P;
    a.n=n+_b.n; return a;
  }
};
```

9.8 Crt Solve2

```
LL solve(LL x1, LL m1, LL x2, LL m2){
    LL g = __gcd(m1, m2);
    if((x2 - x1) % g)return -1;
    m1 /= g; m2 /= g;
    pll p = gcd(m1, m2);
    LL lcm = m1 * m2 * g;
    LL res = p.x * (x2 - x1) * m1 + x1;
    return (res % lcm + lcm) % lcm;
}
```

9.9 DiscreteSqrt

```
void calcH(int &t, int &h, const int p) {
  int tmp=p-1; for(t=0;(tmp&1)==0;tmp/=2) t++; h=tmp;
}
// solve equation x^2 mod p = a
bool solve(int a, int p, int &x, int &y) {
  if(p == 2) { x = y = 1; return true; }
  int p2 = p / 2, tmp = mypow(a, p2, p);
  if (tmp == p - 1) return false;
  if ((p + 1) % 4 == 0) {
    x=mypow(a,(p+1)/4,p); y=p-x; return true;
  } else {
    int t, h, b, pb; calcH(t, h, p);
    if (t >= 2) {
      do {b = rand() % (p - 2) + 2;
      } while (mypow(b, p / 2, p) != p - 1);
      pb = mypow(b, h, p);
    } int s = mypow(a, h / 2, p);
    for (int step = 2; step <= t; step++) {
      int ss = (((LL)(s * s) % p) * a) % p;
      for(int i=0;i<t-step;i++) ss=mul(ss,ss,p);
      if (ss + 1 == p) s = (s * pb) % p;
      pb = ((LL)pb * pb) % p;
    } x = ((LL)s * a) % p; y = p - x;
  } return true;
}
```

9.10 FWT

```
void FWT(int *x,int inv) {
    for(int i=1;i<lim;i<<=1){
        for(int j=0;j<lim;++j)/* or */
            if(j&i)
                x[j]= inv ? x[j]-x[j^i] : x[j]+x[j^i];
        for(int j=0;j<lim;j+=(i<<1))/* and */
            for(int k=0;k<i;++k)
                x[j+k]=inv ? x[j+k]-x[j+k+i] : x[j+k]+x[j+k+i
     ];
        for(int j=0;j<lim;j+=(i<<1))/* xor */
            for(int k=0;k<i;++k) {
                int y=x[j+k],z=x[j+k+i];
                x[j+k]=inv ? (y+z)/2 : y+z;
                x[j+k+i]=inv ? (y-z)/2 : y-z;
            }
    }
}
```

9.11 Floor Sum

```
LL floor_sum(LL n, LL m, LL a, LL b) {
    //sum_0^{n-1}floor((a*i+b)/m)
    LL ans = 0;
    ans += (n - 1) * n * (a / m) / 2;
    a %= m;
    ans += n * (b / m);
    b %= m;
    LL y_max = (a * n + b) / m, x_max = (y_max * m - b);
    if (y_max == 0) return ans;
    ans += (n - (x_max + a - 1) / a) * y_max;
    ans += floor_sum(y_max, a, m, (a - x_max % a) % a);
    return ans;
}
```

9.12 Faulhaber

```
/* faulhaber ’ s formula -
 * cal power sum formula of all p=1~k in O(k^2) */
#define MAXK 2500
const int mod = 1000000007;
int b[MAXK]; // bernoulli number
int inv[MAXK+1]; // inverse
int cm[MAXK+1][MAXK+1]; // combinactories
int co[MAXK][MAXK+2]; // coeeficient of x^j when p=i
inline int getinv(int x) {
  int a=x,b=mod,a0=1,a1=0,b0=0,b1=1;
  while(b) {
    int q,t;
    q=a/b; t=b; b=a-b*q; a=t;
    t=b0; b0=a0-b0*q; a0=t;
    t=b1; b1=a1-b1*q; a1=t;
  }
  return a0<0?a0+mod:a0;
}
inline void pre() {
  /* combinational */
  for(int i=0;i<=MAXK;i++) {
    cm[i][0]=cm[i][i]=1;
    for(int j=1;j<i;j++)
      cm[i][j]=add(cm[i-1][j-1],cm[i-1][j]);
  }
  /* inverse */
  for(int i=1;i<=MAXK;i++) inv[i]=getinv(i);
  /* bernoulli */
  b[0]=1; b[1]=getinv(2); // with b[1] = 1/2
  for(int i=2;i<MAXK;i++) {
    if(i&1) { b[i]=0; continue; }
    b[i]=1;
    for(int j=0;j<i;j++)
      b[i]=sub(b[i],
               mul(cm[i][j],mul(b[j], inv[i-j+1])));
  }
  /* faulhaber */
  // sigma_x=1~n {x^p} =
  //   1/(p+1) * sigma_j=0~p {C(p+1,j)*Bj*n^(p-j+1)}
  for(int i=1;i<MAXK;i++) {
    co[i][0]=0;
    for(int j=0;j<=i;j++)
      co[i][i-j+1]=mul(inv[i+1], mul(cm[i+1][j], b[j]))
  }
}
/* sample usage: return f(n,p) = sigma_x=1~n (x^p) */
inline int solve(int n,int p) {
  int sol=0,m=n;
  for(int i=1;i<=p+1;i++) {
    sol=add(sol,mul(co[p][i],m));
    m = mul(m, n);
  }
  return sol;
}
```

9.13 Extgcd

```
typedef pair<int, int> pii;
pii gcd(int a, int b){
    if(b == 0) return mp(1, 0);
    else{
        int p = a / b;
        pii q = gcd(b, a % b);
        return make_pair(q.y, q.x - q.y * p);
    }
}
```

9.14 Pollard’sRho

```
// does not work when n is prime
inline LL f(LL x , LL mod) {
return (x * x % mod + 1) % mod;
}
inline LL pollard_rho(LL n) {
  if(!(n&1)) return 2;
  while(true) {
    LL y = 2 , x = rand() % (n - 1) + 1 , res = 1;
    for(int sz = 2; res == 1; sz *= 2) {
      for(int i = 0; i < sz && res <= 1; i++) {
        x = f(x , n);
        res = __gcd(abs(x - y) , n);
      }
      y = x;
    }
    if (res != 0 && res != n) return res;
  }
}
```

10 monge

$i \leq i^{'} < j \leq j^{'}$

$m(i,j) + m(i^{'},j^{'}) \leq m(i^{'},j) + m(i,j^{'})$

$k(i,j-1) <= k(i,j) <= k(i+1,j)$

11 四心

$\frac{sa*A+sb*B+sc*C}{sa+sb+sc}$

外心 sin 2A : sin 2B : sin 2C

內心 sin A : sin B : sin C

垂心 tan A : tan B : tan C

重心 1 : 1 : 1

12 Runge-Kutta

$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$k_1 = f(t_n, y_n)$

$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$

$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_3)$

$k_2 = f(t_n + h, y_n + hk_3)$

13 Householder Matrix

$I - 2\frac{vv^T}{v^Tv}$

14 Simpson’s-rule

$\int_a^b f(x)dx = \frac{b-a}{6}(f(a) + 4f(\frac{a+b}{2}) + f(b))$