Contents

1 Basic

1.1 Default code

```cpp
// test RE compile: g++ a.cpp -fsanitize=undefined -o a
#include<bits/stdc++.h>
#define int long long
#define mod 1000000007
#define endl '\n'
#define pii pair<int,int>
using namespace std;

signed main(){
  ios::sync_with_stdio(0),cin.tie(0);
}
```

1.2 Linux 對拍

```bash
set -e
for ((i=0;i<300;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done
```

1.3 Windows 對拍

```bat
@echo off
:loop
    echo %%x
    python gen.py > input
    ./ac.exe < input > ac.out
    ./wa.exe < input > wa.out
    fc ac.out wa.out
if not errorlevel 1 goto loop
```

1.4 builtin 函數

```cpp
// 右邊第一個 1 的位置
int __builtin_ffs(unsigned int);
int __builtin_ffsl(unsigned long);
int __builtin_ffsll(unsigned long long);
// 左邊第一個 1 之前 0 的數量
int __builtin_clz(unsigned int);
int __builtin_clzl(unsigned long);
int __builtin_clzll(unsigned long long);
// 右邊第一個 1 之後 0 的數量
int __builtin_ctz(unsigned int);
int __builtin_ctzl(unsigned long);
int __builtin_ctzll(unsigned long long);
// 1 的數量
int __builtin_popcount(unsigned int);
int __builtin_popcountl(unsigned long);
int __builtin_popcountll(unsigned long long);
// 1 的數量 mod 2
int __builtin_parity(unsigned int);
int __builtin_parityl(unsigned long);
int __builtin_parityll(unsigned long long);
// 二進制表示數字
int a = 0b101101;
```

1.5 輸入輸出

```cpp
// 開讀檔
fropen("input_file_name","r",stdin);
fropen("output_file_name","w",stdout);
```

1.6 Python 輸入輸出

```python
import sys, os

# 設定大數運算最大位數, 複雜度需考慮運算位數
sys.set_int_max_str_digits(100000)

# 開讀檔
if(os.path.exists('input_file.txt')):
    sys.stdin = open("input_file.txt","r")
    sys.stdout = open("output_file.txt","w")
```

2 Data Structure

2.1 持久化線段樹

```
struct Seg{
    struct Node{
        int v;
        Node* l,*r;
    };
    vector<Node*> version;
    Node* build(int l,int r){
        Node* node=new Node;
        if(l==r){
           node->v=l;
           return node;
        }
        int mid=(l+r)/2;
        node->l=build(l,mid);
        node->r=build(mid+1,r);
        return node;
    }
    int query(Node* cur,int l,int r,int x){
        if(l==r){
            return cur->v;
        }
        int mid=(l+r)/2;
        if(x<=mid) return query(cur->l,l,mid,x);
        else return query(cur->r,mid+1,r,x);
    }
    Node* update(Node* cur,int l,int r,int x,int y){
        Node* node=new Node;
        if(l==r){
           node->v=y;
           return node;
        }
        int mid=(l+r)/2;
        if(x<=mid){
            node->l=update(cur->l,l,mid,x,y);
            node->r=cur->r;
        }
        else{
            node->l=cur->l;
            node->r=update(cur->r,mid+1,r,x,y);
        }
        return node;
    }
};
```

2.2 Treap

```
mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count()); // C++ randomizer
struct Node {
    int k, p, sz = 1;
    Node *l = 0, *r = 0;
    bool tag = 0;
    Node(int kk) {
        k = kk;
        p = gen();
    }
};
Node *root = 0;
int size(Node *x) {return x ? x->sz : 0;}
void push(Node *x) {
    if(x->tag) {
        if(x->l) x->l->tag ^= true;
        if(x->r) x->r->tag ^= true;
        x->tag = false;
    }
}
void pull(Node* x) {
    x->sz = size(x->l) + size(x->r) + 1;
}
Node* merge(Node *a, Node *b) {
    if(!a || !b) return a ?: b;
    if(a->p > b->p) {
        push(a);
        a->r = merge(a->r, b);
        pull(a);
        return a;
    }
    else{
        push(b);
        b->l = merge(a, b->l);
        pull(b);
        return b;
    }
}
void splitKey(Node* x, int k, Node *&a, Node *&b) {
    if(!x) {a = b = 0; return;}
    push(x);
    if(x->k <= k) {
        a = x;
        splitKey(a->r, k, a->r, b);
        pull(a);
    }
    else{
        b = x;
        splitKey(b->l, k, a, b->l);
        pull(b);
    }
}
void splitKth(Node *x, int k, Node *&a, Node *&b) {
    if(!x) {a = b = 0; return;}
    push(x);
    if(size(x->l) < k) {
        a = x;
        splitKth(a->r, k - size(x->l) - 1, a->r, b);
        pull(a);
    }
    else{
        b = x;
        splitKth(b->l, k, a, b->l);
        pull(b);
    }
}
void insert(int id) {
    Node *l, *r;
    splitKey(root, id, l, r);
    Node *m = new Node(id);
    root = merge(l, merge(m, r));
}
void erase(int x) {
    Node *a, *b, *c;
    splitKey(root, x, b, c);
    splitKey(b, x - 1, a, b);
    root = merge(a, c);
}
```

2.3 線段樹

```
struct Seg{
    vector<int> seg,tag;
    #define cl (i<<1)+1
    #define cr (i<<1)+2
    void push(int i,int l,int r){
        if(tag[i]!=0){
            seg[i]+=tag[i]; // update by tag
            if(l!=r){
                tag[cl]+=tag[i]; // push
                tag[cr]+=tag[i]; // push
            }
            tag[i]=0;
        }
    }
    void pull(int i,int l,int r){
        int mid=(l+r)>>1;
        push(cl,l,mid);push(cr,mid+1,r);
        seg[i]=max(seg[cl],seg[cr]); // pull
    }
    void build(int i,int l,int r,vector<int>&arr){
        if(l==r){
            seg[i]=arr[l]; // set value
            return;
        }
        int mid=(l+r)>>1;
        build(cl,l,mid,arr);
        build(cr,mid+1,r,arr);
        pull(i,l,r);
    }
    void init(vector<int>& arr){
        seg.resize(arr.size()*4);
        tag.resize(arr.size()*4);
        build(0,0,arr.size()-1,arr);
```

```
  }
  void update(int i,int l,int r,int nl,int nr,int x){
    push(i,l,r);
    if(nl<=l&&r<=nr){
      tag[i]+=x;
      return;
    }
    int mid=(l+r)>>1;
    if(nl<=mid) update(cl,l,mid,nl,nr,x);
    if(nr>mid) update(cr,mid+1,r,nl,nr,x);
    pull(i,l,r);
  }
  int query(int i,int l,int r,int nl,int nr){
    push(i,l,r);
    if(nl<=l&&r<=nr){
      return seg[i];
    }
    int mid=(l+r)>>1;
    int ans=0;
    if(nl<=mid) ans=max(ans,query(cl,l,mid,nl,nr));
    if(nr>mid) ans=max(ans,query(cr,mid+1,r,nl,nr));
    return ans;
  }
};
```

3 Flow

3.1 Dinic

```
const int MXN=1000;
struct Dinic
{
 struct Edge
 {
  int v, f, re;
 };
 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 addEdge(int u, int v, int f)
 {
  E[u].push_back({v, f, (int)(E[v].size())});
  E[v].push_back({u, 0, (int)(E[u].size())-1});
 }
 bool BFS()
 {
  for (int i = 0; i < n; i++)
   level[i] = -1;
  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;
```

3.2 匈牙利

```
#define NIL -1
#define INF 100000000
int n,matched;
int cost[MAXN][MAXN];
bool sets[MAXN]; // whether x is in set S
bool sett[MAXN]; // whether y is in set T
int xlabel[MAXN],ylabel[MAXN];
int xy[MAXN],yx[MAXN]; // matched with whom
int slack[MAXN]; // given y: min{xlabel[x]+ylabel[y]-cost[x][y]} | x not in S
int prev[MAXN]; // for augmenting matching
inline void relabel() {
 int i,delta=INF;
 for(i=0;i<n;i++) if(!sett[i]) delta=min(slack[i],delta);
 for(i=0;i<n;i++) if(sets[i]) xlabel[i]-=delta;
 for(i=0;i<n;i++) {
  if(sett[i]) ylabel[i]+=delta;
  else slack[i]-=delta;
 }
}
inline void add_sets(int x) {
 int i;
 sets[x]=1;
 for(i=0;i<n;i++) {
  if(xlabel[x]+ylabel[i]-cost[x][i]<slack[i]) {
   slack[i]=xlabel[x]+ylabel[i]-cost[x][i];
   prev[i]=x;
  }
 }
}
inline void augment(int final) {
 int x=prev[final],y=final,tmp;
 matched++;
 while(1) {
  tmp=xy[x]; xy[x]=y; yx[y]=x; y=tmp;
  if(y==NIL) return;
  x=prev[y];
 }
}
inline void phase() {
 int i,y,root;
 for(i=0;i<n;i++) { sets[i]=sett[i]=0; slack[i]=INF; }
 for(root=0;root<n&&xy[root]!=NIL;root++);
 add_sets(root);
 while(1) {
  relabel();
  for(y=0;y<n;y++) if(!sett[y]&&slack[y]==0) break;
  if(yx[y]==NIL) { augment(y); return; }
  else { add_sets(yx[y]); sett[y]=1; }
 }
}
inline int hungarian() {
 int i,j,c=0;
 for(i=0;i<n;i++) {
  xy[i]=yx[i]=NIL;
  xlabel[i]=ylabel[i]=0;
```

```
    for(j=0;j<n;j++) xlabel[i]=max(cost[i][j],xlabel[i]);
  }
  for(i=0;i<n;i++) phase();
  for(i=0;i<n;i++) c+=cost[i][xy[i]];
  return c;
}
```

3.3 KM

```
struct KM{ // max weight, for min negate the weights
  int n, mx[MXN], my[MXN], pa[MXN];
  ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
  bool vx[MXN], vy[MXN];
  void init(int _n) { // 1-based
    n = _n;
    for(int i=1; i<=n; i++) fill(g[i], g[i]+n+1, 0);
  }
  void addEdge(int x, int y, ll w) {g[x][y] = w;}
  void augment(int y) {
    for(int x, z; y; y = z)
      x=pa[y], z=mx[x], my[y]=x, mx[x]=y;
  }
  void bfs(int st) {
    for(int i=1; i<=n; ++i) sy[i]=INF, vx[i]=vy[i]=0;
    queue<int> q; q.push(st);
    for(;;) {
      while(q.size()) {
        int x=q.front(); q.pop(); vx[x]=1;
        for(int y=1; y<=n; ++y) if(!vy[y]){
          ll t = lx[x]+ly[y]-g[x][y];
          if(t==0){
            pa[y]=x;
            if(!my[y]){augment(y);return;}
            vy[y]=1, q.push(my[y]);
          }else if(sy[y]>t) pa[y]=x,sy[y]=t;
      } }
      ll cut = INF;
      for(int y=1; y<=n; ++y)
        if(!vy[y]&&cut>sy[y]) cut=sy[y];
      for(int j=1; j<=n; ++j){
        if(vx[j]) lx[j] -= cut;
        if(vy[j]) ly[j] += cut;
        else sy[j] -= cut;
      }
      for(int y=1; y<=n; ++y) if(!vy[y]&&sy[y]==0){
        if(!my[y]){augment(y);return;}
        vy[y]=1, q.push(my[y]);
  } } }
  ll solve(){
    fill(mx, mx+n+1, 0); fill(my, my+n+1, 0);
    fill(ly, ly+n+1, 0); fill(lx, lx+n+1, -INF);
    for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y)
      lx[x] = max(lx[x], g[x][y]);
    for(int x=1; x<=n; ++x) bfs(x);
    ll ans = 0;
    for(int y=1; y<=n; ++y) ans += g[my[y]][y];
    return ans;
} }graph;
```

3.4 MCMF

```
struct MCMF {
    #define SZ(x) (int)(x.size())
    struct Edge {
        int v, f, re, c;
    };
    vector<vector<Edge>> E;
    vector<int> dis, x, y;
    int n, s, t;
    MCMF(int nn, int ss, int tt) {
        n = nn; s = ss; t = tt;
        E.resize(n);
        x.resize(n);
        y.resize(n);
    }
    void addEdge(int u, int v, int w, int c) {
        E[u].push_back({v, w, SZ(E[v]), c});
        E[v].push_back({u, 0, SZ(E[u]) - 1, -c});
    }
    bool spfa() {
        dis.assign(n, 0x3f3f3f3f);
        x.assign(n, -1);
        y.assign(n, -1);
        vector<bool> inq(n, false);
        queue<int> q;
        q.push(s);
        inq[s] = true;
        dis[s] = 0;
        while(q.size()) {
            int u = q.front(); q.pop();
            inq[u] = false;
            for(int i = 0; i < E[u].size(); i++) {
                auto& it = E[u][i];
                int v = it.v;
                if(it.f > 0 && dis[v] > dis[u] + it.c) {
                    dis[v] = dis[u] + it.c;
                    x[v] = u;
                    y[v] = i;
                    if(!inq[v]) {
                        q.push(v);
                        inq[v] = true;
                    }
                }
            }
        }
        return x[t] != -1;
    }
    pii solve() {
        int mf = 0, mc = 0;
        while(spfa()) {
            int nf = 0x3f3f3f3f;
            for(int i = t; i != s; i = x[i]) {
                nf = min(nf, E[x[i]][y[i]].f);
            }
            for(int i = t; i != s; i = x[i]) {
                auto& it = E[x[i]][y[i]];
                it.f -= nf;
                E[it.v][it.re].f += nf;
            }
            mf += nf;
            mc += nf * dis[t];
        }
        return {mf, mc};
    }
};
```

4 幾何

4.1 點宣告

```
typedef long double ld;
const ld eps = 1e-8;
int dcmp(ld x) {
  if(abs(x) < eps) return 0;
  else return x < 0 ? -1 : 1;
}
struct Pt {
  ld x, y;
  Pt(ld _x=0, ld _y=0):x(_x), y(_y) {}
  Pt operator+(const Pt &a) const {
    return Pt(x+a.x, y+a.y);  }
  Pt operator-(const Pt &a) const {
    return Pt(x-a.x, y-a.y);  }
  Pt operator*(const ld &a) const {
    return Pt(x*a, y*a);  }
  Pt operator/(const ld &a) const {
    return Pt(x/a, y/a);  }
  ld operator*(const Pt &a) const {
    return x*a.x + y*a.y;  }
  ld operator^(const Pt &a) const {
    return x*a.y - y*a.x;  }
  auto operator<=>(const Pt &a) const {
    return (x != a.x) ? x <=> a.x : y <=> a.y; }
    //return dcmp(x-a.x) < 0 || (dcmp(x-a.x) == 0 && dcmp(y-a.y) < 0); }
  bool operator==(const Pt &a) const {
    return dcmp(x-a.x) == 0 && dcmp(y-a.y) == 0;  }
};
ld norm2(const Pt &a) {
  return a*a; }
ld norm(const Pt &a) {
  return sqrt(norm2(a)); }
```

```
Pt perp(const Pt &a) {
  return Pt(-a.y, a.x); }
Pt rotate(const Pt &a, ld ang) {
  return Pt(a.x*cos(ang)-a.y*sin(ang), a.x*sin(ang)+a.y
      *cos(ang)); }
struct Line {
  Pt s, e, v; // start, end, end-start
  ld ang;
  Line(Pt _s=Pt(0, 0), Pt _e=Pt(0, 0)):s(_s), e(_e) { v
       = e-s; ang = atan2(v.y, v.x); }
  bool operator<(const Line &L) const {
    return ang < L.ang;
} };
struct Circle {
  Pt o; ld r;
  Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}
};
```

4.2 矩形面積

```
struct AreaofRectangles{
#define cl(x) (x<<1)
#define cr(x) (x<<1|1)
    ll n, id, sid;
    pair<ll,ll> tree[MXN<<3];   // count, area
    vector<ll> ind;
    tuple<ll,ll,ll,ll> scan[MXN<<1];
    void pull(int i, int l, int r){
        if(tree[i].first)  tree[i].second = ind[r+1] -
            ind[l];
        else if(l != r){
            int mid = (l+r)>>1;
            tree[i].second = tree[cl(i)].second + tree[
                cr(i)].second;
        }
        else    tree[i].second = 0;
    }
    void upd(int i, int l, int r, int ql, int qr, int v
        ){
        if(ql <= l && r <= qr){
            tree[i].first += v;
            pull(i, l, r); return;
        }
        int mid = (l+r) >> 1;
        if(ql <= mid) upd(cl(i), l, mid, ql, qr, v);
        if(qr > mid) upd(cr(i), mid+1, r, ql, qr, v);
        pull(i, l, r);
    }
    void init(int _n){
        n = _n; id = sid = 0;
        ind.clear(); ind.resize(n<<1);
        fill(tree, tree+(n<<2), make_pair(0, 0));
    }
    void addRectangle(int lx, int ly, int rx, int ry){
        ind[id++] = lx; ind[id++] = rx;
        scan[sid++] = make_tuple(ly, 1, lx, rx);
        scan[sid++] = make_tuple(ry, -1, lx, rx);
    }
    ll solve(){
        sort(ind.begin(), ind.end());
        ind.resize(unique(ind.begin(), ind.end()) - ind
            .begin());
        sort(scan, scan + sid);
        ll area = 0, pre = get<0>(scan[0]);
        for(int i = 0; i < sid; i++){
            auto [x, v, l, r] = scan[i];
            area += tree[1].second * (x-pre);
            upd(1, 0, ind.size()-1, lower_bound(ind.
                begin(), ind.end(), l)-ind.begin(),
                lower_bound(ind.begin(),ind.end(),r)-
                ind.begin()-1, v);
            pre = x;
        }
        return area;
}   }rect;
```

4.3 最近點對

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
using ld = long double;
const int mod = 1e9+7;
struct pt{
    int x,y;
    int id;
    ld dis(const pt& rhs){
        return sqrt((x-rhs.x)*(x-rhs.x)+(y-rhs.y)*(y-
            rhs.y));
    }
};
signed main(){
    int n;
    cin>>n;
    vector<pt> a(n);
    for(int i=0;i<n;i++){
        cin>>a[i].x>>a[i].y;
        a[i].id=i;
    }
    ld ans = 1e19;
    sort(a.begin(),a.end(),[](const pt&a,const pt&b){
        if(a.x==b.y)return a.y<b.y;
        return a.x<b.x;
    });
    pt ans2;
    function<void(int,int)> dnq = [&](int l,int r){
        if(r-l<4){
            for(int i=l;i<=r;i++){
                for(int j=i+1;j<=r;j++){
                    ld temans = a[i].dis(a[j]);
                    if(temans<ans){
                        ans=temans;
                        ans2 = {a[i].id,a[j].id};
                    }
                }
            }
            sort(a.begin()+l,a.begin()+r+1,[](const pt&
                a,const pt&b){return a.y<b.y;});
            return;
        }
        int mid = (l+r)/2;
        int midx = a[mid].x;
        dnq(l,mid);dnq(mid+1,r);
        inplace_merge(a.begin()+l,a.begin()+mid+1,a.
            begin()+r+1,[](const pt&a,const pt&b){
            return a.y<b.y;});
        vector<int> c;c.reserve(r-l+1);
        for(int i=l;i<=r;i++){
            if(abs(a[i].x-midx)<ans){
                for(int j=c.size()-1;j>=0&&a[i].y-a[c[j
                    ]].y<ans;j--){
                    ld temans = a[i].dis(a[c[j]]);
                    if(temans<ans){
                        ans=temans;
                        ans2 = {a[i].id,a[c[j]].id
                            };
                    }
                }
            }
            c.push_back(i);
        }

    };
    dnq(0,n-1);
    cout<<min(ans2.x,ans2.y)<<' '<<max(ans2.x,ans2.y)<<
        ' '<<fixed<<setprecision(6)<<ans<<'\n';
}
```

4.4 凸包

```
double cross(Pt o, Pt a, Pt b){
  return (a-o) ^ (b-o);
}
void convex_hull(vector<Pt> pt, vector<Pt>& hull){
  sort(pt.begin(),pt.end());
  int top=0;
  hull = vector<Pt>(2*pt.size());
  for (int i=0; i<(int)pt.size(); i++){
    while (top >= 2 && cross(hull[top-2],hull[top-1],pt
        [i]) <= 0)
      top--;
    hull[top++] = pt[i];
  }
  for (int i=pt.size()-2, t=top+1; i>=0; i--){
    while (top >= t && cross(hull[top-2],hull[top-1],pt
        [i]) <= 0)
      top--;
    hull[top++] = pt[i];
  }
  hull.resize(top-1);
}
```

4.5 兩直線交點

```
Pt LLIntersect(Line a, Line b) {
  Pt p1 = a.s, p2 = a.e, q1 = b.s, q2 = b.e;
  ld f1 = (p2-p1)^(q1-p1),f2 = (p2-p1)^(p1-q2),f;
  if(dcmp(f=f1+f2) == 0)
    return dcmp(f1)?Pt(NAN,NAN):Pt(INFINITY,INFINITY);
  return q1*(f2/f) + q2*(f1/f);
}
```

4.6 兩線段交點

```
int ori( const Pt& o , const Pt& a , const Pt& b ){
  LL ret = ( a - o ) ^ ( b - o );
  return (ret > 0) - (ret < 0);
}
// p1 == p2 || q1 == q2 need to be handled
bool banana( const Pt& p1 , const Pt& p2 ,
             const Pt& q1 , const Pt& q2 ){
  if( ( ( p2 - p1 ) ^ ( q2 - q1 ) ) == 0 ){ // parallel
    if( ori( p1 , p2 , q1 ) ) return false;
    return ( ( p1 - q1 ) * ( p2 - q1 ) ) <= 0 ||
           ( ( p1 - q2 ) * ( p2 - q2 ) ) <= 0 ||
           ( ( q1 - p1 ) * ( q2 - p1 ) ) <= 0 ||
           ( ( q1 - p2 ) * ( q2 - p2 ) ) <= 0;
  }
  return (ori( p1, p2, q1 ) * ori( p1, p2, q2 )<=0) &&
         (ori( q1, q2, p1 ) * ori( q1, q2, p2 )<=0);
}
```

4.7 李超線段樹

```
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; } // overflow
  };
  struct node{
    node *l,*r; line f;
    node(line v){ f=v; l=r=NULL; }
  };
  typedef node* pnode;
  pnode root; ll sz,ql,qr;
#define mid ((l+r)>>1)
  void insert(line v,ll l,ll r,pnode &nd){
    /* if(!(ql<=l&&r<=qr)){
      if(!nd) nd=new node(line(0,INF));
      if(ql<=mid) insert(v,l,mid,nd->l);
      if(qr>mid) insert(v,mid+1,r,nd->r);
      return;
    } used for adding segment */
    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(ll x,ll l,ll r,pnode &nd){
    if(!nd) return INF;
    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<=ll query_x<=sz */
  void init(ll _sz){ sz=_sz+1; root=NULL; }
  void add_line(ll m,ll c,ll l=-INF,ll r=INF){
    line v(m,c); ql=l; qr=r; insert(v,-sz,sz,root);
  }
  ll query(ll x) { return query(x,-sz,sz,root); }
};
```

4.8 最小包覆圓

```
/* minimum enclosing circle */
int n;
Pt p[ N ];
const Circle circumcircle(Pt a,Pt b,Pt c){
  Circle cir;
  double fa,fb,fc,fd,fe,ff,dx,dy,dd;
  if( iszero( ( b - a ) ^ ( c - a ) ) ){
    if( ( ( b - a ) * ( c - a ) ) <= 0 )
      return Circle((b+c)/2,norm(b-c)/2);
    if( ( ( c - b ) * ( a - b ) ) <= 0 )
      return Circle((c+a)/2,norm(c-a)/2);
    if( ( ( a - c ) * ( b - c ) ) <= 0 )
      return Circle((a+b)/2,norm(a-b)/2);
  }else{
    fa=2*(a.x-b.x);
    fb=2*(a.y-b.y);
    fc=norm2(a)-norm2(b);
    fd=2*(a.x-c.x);
    fe=2*(a.y-c.y);
    ff=norm2(a)-norm2(c);
    dx=fc*fe-ff*fb;
    dy=fa*ff-fd*fc;
    dd=fa*fe-fd*fb;
    cir.o=Pt(dx/dd,dy/dd);
    cir.r=norm(a-cir.o);
    return cir;
  }
}
inline Circle mec(int fixed,int num){
  int i;
  Circle cir;
  if(fixed==3) return circumcircle(p[0],p[1],p[2]);
  cir=circumcircle(p[0],p[0],p[1]);
  for(i=fixed;i<num;i++) {
    if(cir.inside(p[i])) continue;
    swap(p[i],p[fixed]);
    cir=mec(fixed+1,i+1);
  }
  return cir;
}
inline double min_radius() {
  if(n<=1) return 0.0;
  if(n==2) return norm(p[0]-p[1])/2;
  scramble();
  return mec(0,n).r;
}
```

4.9 最小包覆球

```
// Pt : { x , y , z }
#define N 202020
int n, nouter; Pt pt[ N ], outer[4], res;
double radius,tmp;
void ball() {
  Pt q[3]; double m[3][3], sol[3], L[3], det;
  int i,j; res.x = res.y = res.z = radius = 0;
  switch ( nouter ) {
    case 1: res=outer[0]; break;
    case 2: res=(outer[0]+outer[1])/2; radius=norm2(res
        , outer[0]); break;
    case 3:
      for (i=0; i<2; ++i) q[i]=outer[i+1]-outer[0];
      for (i=0; i<2; ++i) for(j=0; j<2; ++j) m[i][j]=(q
          [i] * q[j])*2;
      for (i=0; i<2; ++i) sol[i]=(q[i] * q[i]);
      if (fabs(det=m[0][0]*m[1][1]-m[0][1]*m[1][0])<eps
          ) return;
      L[0]=(sol[0]*m[1][1]-sol[1]*m[0][1])/det;
      L[1]=(sol[1]*m[0][0]-sol[0]*m[1][0])/det;
      res=outer[0]+q[0]*L[0]+q[1]*L[1];
      radius=norm2(res, outer[0]);
      break;
    case 4:
      for (i=0; i<3; ++i) q[i]=outer[i+1]-outer[0], sol
          [i]=(q[i] * q[i]);
      for (i=0;i<3;++i) for(j=0;j<3;++j) m[i][j]=(q[i]
          * q[j])*2;
      det= m[0][0]*m[1][1]*m[2][2]
        + m[0][1]*m[1][2]*m[2][0]
        + m[0][2]*m[2][1]*m[1][0]
        - m[0][2]*m[1][1]*m[2][0]
        - m[0][1]*m[1][0]*m[2][2]
        - m[0][0]*m[1][2]*m[2][1];
      if ( fabs(det)<eps ) return;
      for (j=0; j<3; ++j) {
        for (i=0; i<3; ++i) m[i][j]=sol[i];
        L[j]=( m[0][0]*m[1][1]*m[2][2]
               + m[0][1]*m[1][2]*m[2][0]
               + m[0][2]*m[2][1]*m[1][0]
               - m[0][2]*m[1][1]*m[2][0]
               - m[0][1]*m[1][0]*m[2][2]
               - m[0][0]*m[1][2]*m[2][1]
             ) / det;
        for (i=0; i<3; ++i) m[i][j]=(q[i] * q[j])*2;
      } res=outer[0];
      for (i=0; i<3; ++i ) res = res + q[i] * L[i];
      radius=norm2(res, outer[0]);
}}
void minball(int n){ ball();
  if( nouter < 4 ) for( int i = 0 ; i < n ; i ++ )
    if( norm2(res, pt[i]) - radius > eps ){
      outer[ nouter ++ ] = pt[ i ]; minball(i); --
          nouter;
      if(i>0){ Pt Tt = pt[i];
        memmove(&pt[1], &pt[0], sizeof(Pt)*i); pt[0]=Tt
            ;
}}}
double solve(){
  // n points in pt
  random_shuffle(pt, pt+n); radius=-1;
  for(int i=0;i<n;i++) if(norm2(res,pt[i])-radius>eps)
    nouter=1, outer[0]=pt[i], minball(i);
  return sqrt(radius);
}
```

4.10 旋轉卡尺

```
int FarthestPair(vector<Pt>& arr){
    int ret=0;
    for(int i = 0, j = i+1; i<arr.size(); i++){
        while(distance(arr[i], arr[j]) < distance(arr[i
            ], arr[(j+1)%arr.size()]) ){
            j = (j+1) % arr.size();
        }
        ret = max(ret, distance(arr[i],arr[j]));
    }
    return ret;
}
```

4.11 Circle Cover

```
#define N 1021
#define D long double
struct CircleCover{
  int C; Circle c[ N ]; //填入C(圓數量),c(圓陣列)
  bool g[ N ][ N ], overlap[ N ][ N ];
  // Area[i] : area covered by at least i circles
  D Area[ N ];
  void init( int _C ){ C = _C; }
  bool CCinter( Circle& a , Circle& b , Pt& p1 , Pt& p2
       ){
    Pt o1 = a.o , o2 = b.o;
    D r1 = a.r , r2 = b.r;
    if( norm( o1 - o2 ) > r1 + r2 ) return {};
    if( norm( o1 - o2 ) < max(r1, r2) - min(r1, r2) )
      return {};
    D d2 = ( o1 - o2 ) * ( o1 - o2 );
    D d = sqrt(d2);
    if( d > r1 + r2 ) return false;
    Pt u=(o1+o2)*0.5 + (o1-o2)*((r2*r2-r1*r1)/(2*d2));
    D A=sqrt((r1+r2+d)*(r1-r2+d)*(r1+r2-d)*(-r1+r2+d));
    Pt v=Pt( o1.y-o2.y , -o1.x + o2.x ) * A / (2*d2);
    p1 = u + v; p2 = u - v;
    return true;
  }
  struct Teve {
    Pt p; D ang; int add;
    Teve() {}
    Teve(Pt _a, D _b, int _c):p(_a), ang(_b), add(_c){}
    bool operator<(const Teve &a)const
    {return ang < a.ang;}
  }eve[ N * 2 ];
  // strict: x = 0, otherwise x = -1
  bool disjuct( Circle& a, Circle &b, int x )
  {return dcmp( norm( a.o - b.o ) - a.r - b.r ) > x;}
  bool contain( Circle& a, Circle &b, int x )
  {return dcmp( a.r - b.r - norm( a.o - b.o ) ) > x;}
  bool contain(int i, int j){
    /* c[j] is non-strictly in c[i]. */
    return (dcmp(c[i].r - c[j].r) > 0 ||
            (dcmp(c[i].r - c[j].r) == 0 && i < j) ) &&
                contain(c[i], c[j], -1);
  }
  void solve(){
    for( int i = 0 ; i <= C + 1 ; i ++ )
      Area[ i ] = 0;
    for( int i = 0 ; i < C ; i ++ )
      for( int j = 0 ; j < C ; j ++ )
        overlap[i][j] = contain(i, j);
    for( int i = 0 ; i < C ; i ++ )
      for( int j = 0 ; j < C ; j ++ )
        g[i][j] = !(overlap[i][j] || overlap[j][i] ||
                    disjuct(c[i], c[j], -1));
    for( int i = 0 ; i < C ; i ++ ){
      int E = 0, cnt = 1;
      for( int j = 0 ; j < C ; j ++ )
        if( j != i && overlap[j][i] )
          cnt ++;
      for( int j = 0 ; j < C ; j ++ )
        if( i != j && g[i][j] ){
          Pt aa, bb;
          CCinter(c[i], c[j], aa, bb);
          D A=atan2(aa.y - c[i].o.y, aa.x - c[i].o.x);
          D B=atan2(bb.y - c[i].o.y, bb.x - c[i].o.x);
          eve[E ++] = Teve(bb, B, 1);
          eve[E ++] = Teve(aa, A, -1);
          if(B > A) cnt ++;
        }
      if( E == 0 ) Area[ cnt ] += pi * c[i].r * c[i].r;
      else{
        sort( eve , eve + E );
        eve[E] = eve[0];
        for( int j = 0 ; j < E ; j ++ ){
          cnt += eve[j].add;
          Area[cnt] += (eve[j].p ^ eve[j + 1].p) * 0.5;
          D theta = eve[j + 1].ang - eve[j].ang;
          if (theta < 0) theta += 2.0 * pi;
          Area[cnt] +=
            (theta - sin(theta)) * c[i].r*c[i].r * 0.5;
}}}}} CC;
```

4.12 Convex Hull Trick

```
/* Given a convexhull, answer querys in O(\lg N)
CH should not contain identical points, the area should
be > 0, min pair(x, y) should be listed first
(run convex_hull() before pass in) */
struct Convex {
  #ifndef all
    #define all(x) (x).begin(), (x).end()
  #endif
  int n;
  vector < Pt > A, V, L, U;
  Convex(const vector < Pt > & _A): A(_A), n(_A.size())
       { // n >= 3
    auto it = max_element(all(A));
    L.assign(A.begin(), it + 1);
    U.assign(it, A.end()), U.push_back(A[0]);
    for (int i = 0; i < n; i++) {
      V.push_back(A[(i + 1) % n] - A[i]);
    }
  }
  int PtSide(Pt p, Line L) {
    return dcmp(L.v ^ (p - L.s));
  }
  int inside(Pt p,
    const vector < Pt > & h, auto f) {
    auto it = lower_bound(all(h), p, f);
    if (it == h.end()) return 0;
    if (it == h.begin()) return p == * it;
    return 1 - dcmp((p - * prev(it)) ^ ( * it - * prev(it)));
  }
  // 1. whether a given point is inside the CH
  // ret 0: out, 1: on, 2: in
  int inside(Pt p) {
    return min(inside(p, L, less<Pt>()), inside(p, U, greater<Pt>()));
  }
  static bool cmp(Pt a, Pt b) {
    return dcmp(a ^ b) > 0;
  }
  // 2. Find tangent points of a given vector
  // ret the idx of far/closer tangent point
  int tangent(Pt v, bool close = true) {
    assert(v != Pt {});
    auto l = V.begin(), r = V.begin() + L.size() - 1;
    if (v < Pt {}) l = r, r = V.end();
    if (close) return (lower_bound(l, r, v, cmp) - V.begin()) % n;
    return (upper_bound(l, r, v, cmp) - V.begin()) % n;
  }
  // 3. Find 2 tang pts on CH of a given outside point
  // return index of tangent points
  // return {-1, -1} if inside CH
  array < int, 2 > tangent2(Pt p) {
    array < int, 2 > t {
      -1, -1
    };
    if (inside(p) == 2) return t;
    if (auto it = lower_bound(all(L), p); it != L.end() and p == * it) {
      int s = it - L.begin();
      return {
        (s + 1) % n,
        (s - 1 + n) % n
      };
    }
    if (auto it = lower_bound(all(U), p, greater<Pt>()); it != U.end() and p == * it) {
      int s = it - U.begin() + L.size() - 1;
      return {
        (s + 1) % n,
        (s - 1 + n) % n
      };
    }
    for (int i = 0; i != t[0]; i = tangent((A[t[0] = i] - p), 0));
    for (int i = 0; i != t[1]; i = tangent((p - A[t[1] = i]), 1));
    return t;
  }
  int find(int l, int r, Line L) {
    if (r < l) r += n;
    int s = PtSide(A[l % n], L);
    return * ranges::partition_point(views::iota(l, r),
      [ & ](int m) {
        return PtSide(A[m % n], L) == s;
      }) - 1;
  };
  // 4. Find intersection point of a given line
  // intersection is on edge (i, next(i))
  vector < int > intersect(Line L) {
    int l = tangent(L.s - L.e), r = tangent(L.e - L.s);
    if (PtSide(A[l], L) == 0) return {
      l
    };
    if (PtSide(A[r], L) == 0) return {
      r
    };
    if (PtSide(A[l], L) * PtSide(A[r], L) > 0) return {};
    return {
      find(l, r, L) % n,
      find(r, l, L) % n
    };
  }
  #undef all
};
```

4.13 Half Plane Intersection

```
// for point or line solution, change > to >=
bool onleft(Line L, Pt p) {
  return dcmp(L.v^(p-L.s)) > 0;
} // segment should add Counterclockwise
// assume that Lines intersect
vector<Pt> HPI(vector<Line>& L) {
  sort(L.begin(), L.end()); // sort by angle
  int n = L.size(), fir, las;
  Pt *p = new Pt[n];
  Line *q = new Line[n];
  q[fir=las=0] = L[0];
  for(int i = 1 ; i < n ; i++) {
    while(fir < las && !onleft(L[i], p[las-1])) las--;
    while(fir < las && !onleft(L[i], p[fir])) fir++;
    q[++las] = L[i];
    if(dcmp(q[las].v^q[las-1].v) == 0) {
      las--;
      if(onleft(q[las], L[i].s)) q[las] = L[i];
    }
    if(fir < las) p[las-1] = LLIntersect(q[las-1], q[las]);
  }
  while(fir < las && !onleft(q[fir], p[las-1])) las--;
  if(las-fir <= 1) return {};
  p[las] = LLIntersect(q[las], q[fir]);
  int m = 0;
  vector<Pt> ans(las-fir+1);
  for(int i = fir ; i <= las ; i++) ans[m++] = p[i];
  return ans;
}
```

4.14 Minkowski Sum

```
// P, Q, R(return) are counterclockwise order convex polygon
vector<Pt> minkowski(vector<Pt> P, vector<Pt> Q) {
    auto cmp = [&](Pt a, Pt b) {
        return Pt{a.y, a.x} < Pt{b.y, b.x};
    };
    auto reorder = [&](vector<Pt> &R) {
        rotate(R.begin(), min_element(all(R), cmp), R.end());
        R.push_back(R[0]), R.push_back(R[1]);
    };
    const int n = P.size(), m = Q.size();
    reorder(P), reorder(Q);
    vector<Pt> R;
    for (int i = 0, j = 0, s; i < n or j < m; ) {
        R.push_back(P[i] + Q[j]);
        s = dcmp((P[i + 1] - P[i]) ^ (Q[j + 1] - Q[j]));
        if (s >= 0) i++;
        if (s <= 0) j++;
    }
    return R;
}
```

4.15 多邊形聯集面積

```
inline double segP(Pt &p,Pt &p1,Pt &p2){
  if(dcmp(p1.x-p2.x)==0) return (p.y-p1.y)/(p2.y-p1.y);
  return (p.x-p1.x)/(p2.x-p1.x);
}
ld tri(Pt o, Pt a, Pt b){ return (a-o) ^ (b-o);}
double polyUnion(vector<vector<Pt>> py){ //py[0~n-1] must be filled
  int n = py.size();
  int i,j,ii,jj,ta,tb,r,d; double z,w,s,sum=0,tc,td,area;
  vector<pair<double,int>> c;
  for(i=0;i<n;i++){
    area=py[i][py[i].size()-1]^py[i][0];
    for(int j=0;j<py[i].size()-1;j++) area+=py[i][j]^py[i][j+1];
    if((area/=2)<0) reverse(py[i].begin(),py[i].end());
    py[i].push_back(py[i][0]);
  }
  for(i=0;i<n;i++){
    for(ii=0;ii+1<py[i].size();ii++){
      c.clear();
      c.emplace_back(0.0,0); c.emplace_back(1.0,0);
      for(j=0;j<n;j++){
        if(i==j) continue;
        for(jj=0;jj+1<py[j].size();jj++){
          ta=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj]))
              ;
          tb=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj
              +1]));
          if(ta==0 && tb==0){
            if((py[j][jj+1]-py[j][jj])*(py[i][ii+1]-py[
                i][ii])>0&&j<i){
              c.emplace_back(segP(py[j][jj],py[i][ii],
                  py[i][ii+1]),1);
              c.emplace_back(segP(py[j][jj+1],py[i][ii
                  ],py[i][ii+1]),-1);
            }
          }else if(ta>=0 && tb<0){
            tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
            td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
            c.emplace_back(tc/(tc-td),1);
          }else if(ta<0 && tb>=0){
            tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
            td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
            c.emplace_back(tc/(tc-td),-1);
      } } }
      sort(c.begin(),c.end());
      z=min(max(c[0].first,0.0),1.0); d=c[0].second; s
          =0;
      for(j=1;j<c.size();j++){
        w=min(max(c[j].first,0.0),1.0);
        if(!d) s+=w-z;
        d+=c[j].second; z=w;
      }
      sum+=(py[i][ii]^py[i][ii+1])*s;
  } }
  return sum/2;
}
```

4.16 Polygon Cover

```
// Need Line && norm()
double ori(Pt a, Pt b, Pt c) { return (b - a) ^ (c - a)
    ; }
int PtSide(Pt p, Line L) {
  return dcmp(ori(L.s, L.e, p) / norm(L.s - L.e));
}
bool argcmp(const Pt &a, const Pt &b) { // arg(a) < arg
    (b)
  int f = (Pt{a.y, -a.x} > Pt{} ? 1 : -1) * (a != Pt{})
      ;
  int g = (Pt{b.y, -b.x} > Pt{} ? 1 : -1) * (b != Pt{})
      ;
  return f == g ? (a ^ b) > 0 : f < g;
}
Pt LineInter(Line l, Line m) {
  double s = ori(m.s, m.e, l.s), t = ori(m.s, m.e, l.e)
      ;
  return (l.e * s - l.s * t) / (s - t);
}
#ifndef all
  #define all(x) (x).begin(), (x).end()
#endif
vector < double > PolyUnion(const vector < vector < Pt
    >> & P) {
  const int n = P.size();
  vector < double > Area(n + 1);
  vector < Line > Ls;
  for (int i = 0; i < n; i++)
    for (int j = 0; j < P[i].size(); j++)
      Ls.push_back({
        P[i][j],
        P[i][(j + 1) % P[i].size()]
      });
  auto cmp = [ & ](Line & l, Line & r) {
    Pt u = l.e - l.s, v = r.e - r.s;
    if (argcmp(u, v)) return true;
    if (argcmp(v, u)) return false;
    return PtSide(l.s, r) < 0;
  };
  sort(all(Ls), cmp);
  for (int l = 0, r = 0; l < Ls.size(); l = r) {
    while (r < Ls.size() and!cmp(Ls[l], Ls[r])) r++;
    Line L = Ls[l];
    vector < pair < Pt, int >> event;
    for (auto& ls : Ls) {
      auto c = ls.s, d = ls.e;
      if (dcmp((L.s - L.e) ^ (c - d)) != 0) {
        int s1 = PtSide(c, L) == 1;
        int s2 = PtSide(d, L) == 1;
        if (s1 ^ s2) event.emplace_back(LineInter(L, {
          c,
          d
        }), s1 ? 1 : -1);
      } else if (PtSide(c, L) == 0 and dcmp((L.s - L.e)
           * (c - d)) > 0) {
        event.emplace_back(c, 2);
        event.emplace_back(d, -2);
      }
    }
    sort(all(event), [ & ](auto i, auto j) {
      return (L.s - i.first) * (L.s - L.e) < (L.s - j.
          first) * (L.s - L.e);
    });
    int cov = 0, tag = 0;
    Pt lst {
      0,
      0
    };
    for (auto[p, s]: event) {
      if (cov >= tag) {
        Area[cov] += lst ^ p;
        Area[cov - tag] -= lst ^ p;
      }
      if (abs(s) == 1) cov += s;
      else tag += s / 2;
      lst = p;
    }
  }
  for (int i = n - 1; i >= 0; i--) Area[i] += Area[i +
      1];
  for (int i = 1; i <= n; i++) Area[i] /= 2;
  return Area;
};
#undef all
```

4.17 Tangent_line_of_two_circle

```
vector<Line> tang_of_two_circle( const Circle& c1 ,
    const Circle& c2 , int sign1 ){
  // sign1 = 1 for outer tang, -1 for inter tang
  vector<Line> ret;
  double d_sq = norm2( c1.o - c2.o );
  if( d_sq < eps ) return ret;
  double d = sqrt( d_sq );
  Pt v = ( c2.o - c1.o ) / d;
  double c = ( c1.r - sign1 * c2.r ) / d;
  if( c * c > 1 ) return ret;
  double h = sqrt( max( 0.0 , 1.0 - c * c ) );
  for( int sign2 = 1 ; sign2 >= -1 ; sign2 -= 2 ){
    Pt n = { v.x * c - sign2 * h * v.y ,
             v.y * c + sign2 * h * v.x };
    Pt p1 = c1.o + n * c1.r;
    Pt p2 = c2.o + n * ( c2.r * sign1 );
    if( fabs( p1.x - p2.x ) < eps and
        fabs( p1.y - p2.y ) < eps )
      p2 = p1 + perp( c2.o - c1.o );
    ret.push_back( { p1 , p2 } );
  }
  return ret;
}
```

4.18 basic.cpp Revival

```
// Calculate the area of a polygon
ld poly_area(const vector<Pt>& pt) {
  int sz = pt.size();
  ld ret = 0;
  for(int i = 1; i <= sz; ++i) {
    ret += pt[i - 1] ^ pt[i % sz];
  }
  return abs(ret) / 2;
}
// Polar Compare
template<class T> struct CmpByAngle {
  bool operator()(const T& lhs, const T& rhs) {
    using eps_compare::ge;
    T zero = T();
    if((lhs < zero) ^ (rhs < zero))
      return (lhs < zero) < (rhs < zero);
    return (lhs ^ rhs) > 0;
    // return ge(lhs ^ rhs, typename T::value_type());
  }
};
// Pick's Theorm
// A = i + b/2 - 1
// A: Area of polygon
// i: Grid number in the inner
// b: Grid number on the side
constexpr ld pi = 3.14159265359;
// Float compare
bool eq(ld l, ld r) { return abs(l - r) < EPS; }
bool ne(ld l, ld r) { return abs(l - r) > EPS; }
bool le(ld l, ld r) { return (l - r) < -EPS; }
bool ge(ld l, ld r) { return (l - r) > EPS; }
bool leq(ld l, ld r) { return (l - r) < EPS; }
bool geq(ld l, ld r) { return (l - r) > -EPS; }
```

4.19 intersection_of_two_circle

```
#define D ld
vector<Pt> interCircle( Circle& c1 , Circle& c2 ){
  auto o1 = c1.o, o2 = c2.o;
  auto r1 = c1.r, r2 = c2.r;
  if( norm( o1 - o2 ) > r1 + r2 ) return {};
  if( norm( o1 - o2 ) < max(r1, r2) - min(r1, r2) )
      return {};
  D d2 = ( o1 - o2 ) * ( o1 - o2 );
  D d = sqrt(d2);
  if( d > r1 + r2 ) return {};
  Pt u = (o1+o2)*0.5 + (o1-o2)*((r2*r2-r1*r1)/(2*d2));
  D A = sqrt((r1+r2+d)*(r1-r2+d)*(r1+r2-d)*(-r1+r2+d));
  Pt v = Pt( o1.y-o2.y , -o1.x + o2.x ) * A / (2*d2);
  return {u+v, u-v};
}
#undef D
```

5 圖論

5.1 BCC

```
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 addEdge(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);
        }
      }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);
      }
    REP(i,nScc) res.PB(sccv[i]);
    return res;
  }
}graph;
```

5.2 重心剖分

```
struct CentroidDecomposition {
    int n;
    vector<vector<int>> G, out;
    vector<int> sz, v;
    CentroidDecomposition(int _n) : n(_n), G(_n), out(
        _n), sz(_n), v(_n) {}
    int dfs(int x, int par){
        sz[x] = 1;
        for (auto &&i : G[x]) {
            if(i == par || v[i]) continue;
            sz[x] += dfs(i, x);
        }
        return sz[x];
    }
    int search_centroid(int x, int p, const int mid){
        for (auto &&i : G[x]) {
            if(i == p || v[i]) continue;
            if(sz[i] > mid) return search_centroid(i, x
                , mid);
        }
        return x;
    }
    void add_edge(int l, int r){
        G[l].PB(r); G[r].PB(l);
    }
    int get(int x){
        int centroid = search_centroid(x, -1, dfs(x,
            -1)/2);
        v[centroid] = true;
        for (auto &&i : G[centroid]) {
            if(!v[i]) out[centroid].PB(get(i));
        }
        v[centroid] = false;
        return centroid;
} };
```

5.3 極大團

```
#define N 80
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int lnk[N] , v[N];
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      lnk[i].reset(); v[i].reset();
  } }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int ans , stk[N], id[N] , di[N] , deg[N];
  Int cans;
  void dfs(int elem_num, Int candi, Int ex){
    if(candi.none()&&ex.none()){
      cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
      ans = elem_num; // cans is a maximal clique
      return;
    }
    int pivot = (candi|ex)._Find_first();
    Int smaller_candi = candi & (~lnk[pivot]);
    while(smaller_candi.count()){
      int nxt = smaller_candi._Find_first();
      candi[nxt] = smaller_candi[nxt] = 0;
      ex[nxt] = 1;
      stk[elem_num] = nxt;
      dfs(elem_num+1,candi&lnk[nxt],ex&lnk[nxt]);
  } }
  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]) lnk[di[i]][di[j]] = 1;
    ans = 1; cans.reset(); cans[0] = 1;
    dfs(0, Int(string(n,'1')), 0);
    return ans;
} }solver;
```

5.4 最大團

```
#define N 111
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;
} }solver;
```

5.5 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 addEdge(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();
    FZ(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++;
      }
  }
};
```

5.6 SPFA

```
#define MXN 200005
struct SPFA{
  int n;
  LL inq[MXN], len[MXN];
  vector<LL> dis;
  vector<pair<int, LL>> edge[MXN];
  void init(int _n){
    n = _n;
    dis.clear(); dis.resize(n, 1e18);
    for(int i = 0; i < n; i++){
      edge[i].clear();
      inq[i] = len[i] = 0;
  } }
  void addEdge(int u, int v, LL w){
    edge[u].push_back({v, w});
  }
  vector<LL> solve(int st = 0){
    deque<int> dq; //return {-1} if has negative cycle
    dq.push_back(st); //otherwise return dis from st
    inq[st] = 1; dis[st] = 0;
    while(!dq.empty()){
      int u = dq.front();  dq.pop_front();
      inq[u] = 0;
      for(auto [to, d] : edge[u]){
        if(dis[to] > d+dis[u]){
          dis[to] = d+dis[u];
          len[to] = len[u]+1;
          if(len[to] > n) return {-1};
          if(inq[to]) continue;
          (!dq.empty()&&dis[dq.front()] > dis[to]?
              dq.push_front(to) : dq.push_back(to));
          inq[to] = 1;
    } } }
    return dis;
} }spfa;
```

5.7 domainTree

```
#define MXN 200005
struct DominatorTree{ // O(N)
#define REP(i,s,e) for(int i=(s);i<=(e);i++)
#define REPD(i,s,e) for(int i=(s);i>=(e);i--)
  int n , m , s;
  vector< int > g[ MXN ] , pred[ MXN ];
  vector< int > cov[ MXN ];
  int dfn[ MXN ] , nfd[ MXN ] , ts;
  int par[ MXN ]; //idom[u] s到u的最後一個必經點
  int sdom[ MXN ] , idom[ MXN ];
  int mom[ MXN ] , mn[ MXN ];
  inline bool cmp( int u , int v )
  { return dfn[ u ] < dfn[ v ]; }
  int eval( int u ){
    if( mom[ u ] == u ) return u;
    int res = eval( mom[ u ] );
    if(cmp( sdom[ mn[ mom[ u ] ] ] , sdom[ mn[ u ] ] ))
      mn[ u ] = mn[ mom[ u ] ];
    return mom[ u ] = res;
  }
  void init( int _n , int _m , int _s ){
    ts = 0; n = _n; m = _m; s = _s;
    REP( i, 1, n ) g[ i ].clear(), pred[ i ].clear();
  }
  void addEdge( int u , int v ){
    g[ u ].push_back( v );
    pred[ v ].push_back( u );
  }
  void dfs( int u ){
    ts++;
    dfn[ u ] = ts;
    nfd[ ts ] = u;
    for( int v : g[ u ] ) if( dfn[ v ] == 0 ){
      par[ v ] = u;
      dfs( v );
  } }
  void build(){
    REP( i , 1 , n ){
      dfn[ i ] = nfd[ i ] = 0;
      cov[ i ].clear();
      mom[ i ] = mn[ i ] = sdom[ i ] = i;
    }
    dfs( s );
    REPD( i , n , 2 ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      for( int v : pred[ u ] ) if( dfn[ v ] ){
        eval( v );
        if( cmp( sdom[ mn[ v ] ] , sdom[ u ] ) )
          sdom[ u ] = sdom[ mn[ v ] ];
      }
      cov[ sdom[ u ] ].push_back( u );
      mom[ u ] = par[ u ];
      for( int w : cov[ par[ u ] ] ){
        eval( w );
        if( cmp( sdom[ mn[ w ] ] , par[ u ] ) )
          idom[ w ] = mn[ w ];
        else idom[ w ] = par[ u ];
      }
      cov[ par[ u ] ].clear();
    }
    REP( i , 2 , n ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      if( idom[ u ] != sdom[ u ] )
        idom[ u ] = idom[ idom[ u ] ];
} } }domT;
```

5.8 曼哈頓最小生成樹

```
//{{u,v},w}
vector<pair<pair<int,int>, int>> ManhattanMST(vector<Pt
    > P) {
    vector<int> id(P.size());
    iota(id.begin(),id.end(), 0);
    vector<pair<pair<int,int>, int>> edg;
    for (int k = 0; k < 4; k++) {
        sort(id.begin(),id.end(), [&](int i, int j) {
                return (P[i] - P[j]).x < (P[j] - P[i]).
                    y;
            });
        map<int, int> sweep;
        for (int i : id) {
            auto it = sweep.lower_bound(-P[i].y);
            while (it != sweep.end()) {
                int j = it->second;
                Pt d = P[i] - P[j];
                if (d.y > d.x) {
                    break;
                }
                edg.push_back({{i, j},d.x + d.y});
                it = sweep.erase(it);
            }
            sweep[-P[i].y] = i;
        }
        for (Pt &p : P) {
            if (k % 2) {
                p.x = -p.x;
            } else {
                swap(p.x, p.y);
            }
        }
    }
    return edg;
}
```

5.9 2-SAT

$(xory)$ adddege $((x \to \neg y))$, $((y \to \neg x))$

5.10 差分約束

約束條件:

- $V_j - V_i \leq W$ addEdge(i, j, W)
- $V_j - V_i \geq W$ addEdge$(j, i, -W)$
- $V_j = V_i$ addEdge$(i, j, 0)$,$(j, i, 0)$

接著跑 SPFA,Bellman-Ford

6 數論

6.1 離散根號

```
void calcH(LL &t, LL &h, const LL p) {
  LL tmp=p-1; for(t=0;(tmp&1)==0;tmp/=2) t++; h=tmp;
}
// solve equation x^2 mod p = a
bool solve(LL a, LL p, LL &x, LL &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 {
    LL 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;
}
```

6.2 ex-crt

```
typedef __int128 ll;
void exgcd(ll a,ll b,ll &g,ll &x,ll &y) {
    if (b == 0) {
        g = a;
        x = 1;
        y = 0;
        return;
    }
    exgcd(b,a%b,g,y,x);
    y-=(a/b)*x;
}
bool flag = false;
ll a1,a2,n1,n2;
ll abs(ll x) {
    return x>0?x:-x;
}
void china() {
    ll d = a2 - a1;
    ll g,x,y;
    exgcd(n1,n2,g,x,y);
    if (d % g == 0) {
        x = ((x*d/g)%(n2/g)+(n2/g))%(n2/g);
        a1 = x*n1 + a1;
        n1 = (n1*n2)/g;
    }
    else
        flag = true;
}
int n;
long long as[100001];  //算式答案 x
long long ns[100001];  //模數 MOD
ll realchina() {
    a1 = as[0];
    n1 = ns[0];
    for (ll i = 1;i<n;i++) {
        a2 = as[i];
        n2 = ns[i];
        china();
        if (flag)
            return -1;
    }
    return a1;
}
int main() {
    cin>>n;
    flag = false;
    for (ll i = 0;i<n;i++)
        cin>>ns[i]>>as[i];
    cout<<(long long)realchina()<<endl;
}
```

6.3 ex-gcd

```
int exgcd(int a,int b,int&x,int&y){
    if(b==0)return x=1,y=0,a;
    int d = exgcd(b,a%b,y,x);
    y-=a/b*x;
    return d;
}
```

6.4 FFT

```
// const int MAXN = 262144;
// (must be 2^k)
// before any usage, run pre_fft() first
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acosl(-1);
const cplx I(0, 1);
cplx omega[MAXN+1];
void pre_fft(){
  for(int i=0; i<=MAXN; i++)
    omega[i] = exp(i * 2 * PI / MAXN * I);
}
// n must be 2^k
void fft(int n, cplx a[], bool inv=false){
  int basic = MAXN / n;
  int theta = basic;
  for (int m = n; m >= 2; m >>= 1) {
    int mh = m >> 1;
    for (int i = 0; i < mh; i++) {
      cplx w = omega[inv ? MAXN-(i*theta%MAXN)
                         : i*theta%MAXN];
      for (int j = i; j < n; j += m) {
        int k = j + mh;
        cplx x = a[j] - a[k];
        a[j] += a[k];
        a[k] = w * x;
    } }
    theta = (theta * 2) % MAXN;
  }
  int i = 0;
  for (int j = 1; j < n - 1; j++) {
    for (int k = n >> 1; k > (i ^= k); k >>= 1);
    if (j < i) swap(a[i], a[j]);
  }
  if(inv) for (i = 0; i < n; i++) a[i] /= n;
}
cplx arr[MAXN+1];
inline void mul(int _n,ll a[],int _m,ll b[],ll ans[]){
  int n=1,sum=_n+_m-1;
  while(n<sum)
    n<<=1;
  for(int i=0;i<n;i++) {
    double x=(i<_n?a[i]:0),y=(i<_m?b[i]:0);
    arr[i]=complex<double>(x+y,x-y);
  }
  fft(n,arr);
  for(int i=0;i<n;i++)
    arr[i]=arr[i]*arr[i];
  fft(n,arr,true);
  for(int i=0;i<sum;i++)
    ans[i]=(long long int)(arr[i].real()/4+0.5);
}
```

6.5 高斯消去法

```
const int GAUSS_MOD = 100000007LL;
struct GAUSS{
    int n;
    vector<vector<int>> v;
    int ppow(int a , int k){
        if(k == 0) return 1;
        if(k % 2 == 0) return ppow(a * a % GAUSS_MOD ,
            k >> 1);
        if(k % 2 == 1) return ppow(a * a % GAUSS_MOD ,
            k >> 1) * a % GAUSS_MOD;
    }
    vector<int> solve(){
        vector<int> ans(n);
        REP(now , 0 , n){
            REP(i , now , n) if(v[now][now] == 0 && v[i
                ][now] != 0)
                swap(v[i] , v[now]); // det = -det;
            if(v[now][now] == 0) return ans;
            int inv = ppow(v[now][now] , GAUSS_MOD - 2)
                ;
            REP(i , 0 , n) if(i != now){
                int tmp = v[i][now] * inv % GAUSS_MOD;
                REP(j , now , n + 1) (v[i][j] +=
                    GAUSS_MOD - tmp * v[now][j] %
                    GAUSS_MOD) %= GAUSS_MOD;
            }
        }
        REP(i , 0 , n) ans[i] = v[i][n + 1] * ppow(v[i
            ][i] , GAUSS_MOD - 2) % GAUSS_MOD;
        return ans;
    }
    // gs.v.clear() , gs.v.resize(n , vector<int>(n + 1
        , 0));
} gs;
```

6.6 喬瑟夫問題

```
int josephus(int n, int m){ //n人每m次
    int ans = 0;
    for (int i=1; i<=n; ++i)
        ans = (ans + m) % i;
    return ans;
}
```

6.7 定理

- Lucas's Theorem :
  For $n, m \in \mathbb{Z}^*$ and prime P, $C(m,n)$ mod P $= \Pi(C(m_i, n_i))$ where m_i is the i-th digit of m in base P.

- Stirling approximation :
  $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n e^{\frac{1}{12n}}$

- Stirling Numbers(permutation $|P| = n$ with k cycles):
  $S(n,k) =$ coefficient of x^k in $\Pi_{i=0}^{n-1}(x+i)$

- Stirling Numbers(Partition n elements into k non-empty set):
  $S(n,k) = \frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$

- Pick's Theorem : $A = i + b/2 - 1$
  A: Area、i: grid number in the inner、b: grid number on the side

- Catalan number : $C_n = \binom{2n}{n}/(n+1)$
  $C_n^{n+m} - C_{n+1}^{n+m} = (m+n)!\frac{n-m+1}{n+1} \quad for \quad n \geq m$
  $C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$
  $C_0 = 1 \quad and \quad C_{n+1} = 2(\frac{2n+1}{n+2})C_n$
  $C_0 = 1 \quad and \quad C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} \quad for \quad n \geq 0$

- Euler Characteristic:
  planar graph: $V - E + F - C = 1$
  convex polyhedron: $V - E + F = 2$
  V, E, F, C: number of vertices, edges, faces(regions), and components

- Kirchhoff's theorem :
  $A_{ii} = deg(i), A_{ij} = (i,j) \in E \ ? -1 : 0$, Deleting any one row, one column, and cal the det(A)

- Polya' theorem (c is number of color·m is the number of cycle size):
  $(\sum_{i=1}^{m} c^{gcd(i,m)})/m$

- Burnside lemma:
  $|X/G| = \frac{1}{|G|}\sum\limits_{g \in G}|X^g|$

- 錯排公式：（n 個人中，每個人皆不再原來位置的組合數）：
  $dp[0] = 1; dp[1] = 0;$
  $dp[i] = (i-1) * (dp[i-1] + dp[i-2]);$

- Bell 數 (有 n 個人, 把他們拆組的方法總數) :
  $B_0 = 1$
  $B_n = \sum_{k=0}^{n} s(n,k) \quad (second - stirling)$
  $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$

- Wilson's theorem :
  $(p-1)! \equiv -1 (mod \ p)$

- Fermat's little theorem :
  $a^p \equiv a (mod \ p)$

- Euler's totient function:
  $A^{B^C} mod \ p = pow(A, pow(B, C, p-1)) mod \ p$

- 歐拉函數降冪公式:
  $A^B \mod C = A^{B \mod \phi(c) + \phi(c)} \mod C$

- 6 的倍數:
  $(a-1)^3 + (a+1)^3 + (-a)^3 + (-a)^3 = 6a$

6.8 Miller Rabin

```
// n < 4,759,123,141        3 :  2, 7, 61
// n < 1,122,004,669,633    4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383          6 :  pirmes <= 13
// n < 2^64                       7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n-2] if
// you want to use magic.
LL magic[]={}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size)
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

6.9 NTT

```
// Remember coefficient are mod P
/* p=a*2^n+1
   n    2^n         p           a    root
   16   65536       65537       1    3
   20   1048576     7340033     7    3 */
// (must be 2^k)
template<LL P, LL root, int MAXN>
struct NTT{
  static LL bigmod(LL a, LL b) {
    LL res = 1;
    for (LL bs = a; b; b >>= 1, bs = (bs * bs) % P)
      if(b&1) res=(res*bs)%P;
    return res;
  }
  static LL inv(LL a, LL b) {
    if(a==1)return 1;
    return (((LL)(a-inv(b%a,a))*b+1)/a)%b;
  }
  LL omega[MAXN+1];
  NTT() {
    omega[0] = 1;
    LL r = bigmod(root, (P-1)/MAXN);
    for (int i=1; i<=MAXN; i++)
      omega[i] = (omega[i-1]*r)%P;
  }
  // n must be 2^k
  void tran(int n, LL a[], bool inv_ntt=false){
    int basic = MAXN / n , theta = basic;
    for (int m = n; m >= 2; m >>= 1) {
      int mh = m >> 1;
      for (int i = 0; i < mh; i++) {
        LL w = omega[i*theta%MAXN];
        for (int j = i; j < n; j += m) {
          int k = j + mh;
          LL x = a[j] - a[k];
          if (x < 0) x += P;
          a[j] += a[k];
          if (a[j] > P) a[j] -= P;
          a[k] = (w * x) % P;
        }
      }
      theta = (theta * 2) % MAXN;
    }
    int i = 0;
    for (int j = 1; j < n - 1; j++) {
      for (int k = n >> 1; k > (i ^= k); k >>= 1);
      if (j < i) swap(a[i], a[j]);
    }
    if (inv_ntt) {
      LL ni = inv(n,P);
      reverse( a+1 , a+n );
      for (i = 0; i < n; i++)
        a[i] = (a[i] * ni) % P;
} } };
const LL P=2013265921,root=31;
const int MAXN=4194304;
NTT<P, root, MAXN> ntt;
```

6.10 Pollard's Rho

```
// does not work when n is prime  O(n^(1/4))
LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
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;
} }
```

6.11 質數

```
/* 12721, 13331, 14341, 75577, 123457, 222557, 556679
* 999983, 1097774749, 1076767633, 100102021, 999997771
* 1001010013, 1000512343, 987654361, 999991231
* 999888733, 98789101, 987777733, 999991921, 1010101333
* 1010102101, 1000000000039, 1000000000000037
* 2305843009213693951, 4611686018427387847
* 9223372036854775783, 18446744073709551557 */
int mu[ N ] , p_tbl[ N ];
vector<int> primes;
void sieve() {
  mu[ 1 ] = p_tbl[ 1 ] = 1;
  for( int i = 2 ; i < N ; i ++ ){
    if( !p_tbl[ i ] ){
      p_tbl[ i ] = i;
      primes.push_back( i );
      mu[ i ] = -1;
    }
    for( int p : primes ){
      int x = i * p;
      if( x >= M ) break;
      p_tbl[ x ] = p;
      mu[ x ] = -mu[ i ];
      if( i % p == 0 ){
        mu[ x ] = 0;
        break;
} } } }
vector<int> factor( int x ){
  vector<int> fac{ 1 };
  while( x > 1 ){
    int fn = SZ(fac), p = p_tbl[ x ], pos = 0;
    while( x % p == 0 ){
      x /= p;
      for( int i = 0 ; i < fn ; i ++ )
        fac.PB( fac[ pos ++ ] * p );
  } }
  return fac;
}
```

6.12 phi

```
ll phi(ll n){   // 計算小於n的數中與n互質的有幾個
    ll res = n, a=n;    // O(sqrtN)
    for(ll i=2;i*i<=a;i++){
        if(a%i==0){
            res = res/i*(i-1);
            while(a%i==0) a/=i;
    }   }
    if(a>1) res = res/a*(a-1);
    return res;
}
```

6.13 矩陣快速冪

```
LL len,mod;
vector<vector<LL>> operator*(vector<vector<LL>> x,
    vector<vector<LL>> y){
    vector<vector<LL>> ret(len,vector<LL>(len,0));
    for(int i=0;i<len;i++){
        for(int j=0;j<len;j++){
            for(int k=0;k<len;k++){
                ret[i][j]=(ret[i][j]+x[i][k]*y[k][j])%
                    mod;
    }   }   }
    return ret;
}
struct Martix_fast_pow{ //O(len^3 lg k)
    LL init(int _len,LL m=9223372036854775783LL){
        len=_len, mod=m;
    }   // mfp.solve(k,{0, 1}, {1, 1}) k'th fib {值,係
        數} // 0-base
    LL solve(LL n,vector<vector<LL>> poly){
        if(n<len)   return poly[n][0];
        vector<vector<LL>> mar(len,vector<LL>(len,0)),x
            (len,vector<LL>(len,0));
        for(int i=0;i<len;i++)    mar[i][i]=1;
        for(int i=0;i+1<len;i++)  x[i][i+1]=1;
        for(int i=0;i<len;i++)    x[len-1][i]=poly[i
            ][1];
        while(n){
            if(n&1) mar=mar*x;
            n>>=1, x=x*x;
        }
        LL ans=0;
        for(int i=0;i<len;i++)   ans=(ans+mar[len-1][i
            ]*poly[i][0]%mod)%mod;
        return ans;
    }
}mfp;
```

6.14 矩陣相乘

```
template<class T> using Matrix = std::vector<std::
  vector<T>>;
template<class T> void set_matrix(Matrix<T>& m, int x,
    int y, T v) {
  m = Matrix<T>(x, std::vector<T>(y, v));
}
template<class T> Matrix<T> operator*(const Matrix<T>&
  a, const Matrix<T>& b) {
  assert(a[0].size() == b.size());
  int sa = a.size(), sb = b[0].size(), sc = a[0].size()
      ;
  Matrix<T> ret; set_matrix(ret, sa, sb, 0);
  for(int i = 0; i < sa; ++i) {
    for(int j = 0; j < sb; ++j) {
      for(int k = 0; k < sc; ++k) {
        ret[i][j] += a[i][k] * b[k][j];
        // add ret[i][j] %= MOD if needed
      }
    }
  }
  return ret;
}
```

7 字串

7.1 KMP

```
/* len-failure[k]:
在k結尾的情況下，這個子字串可以由開頭
長度為(len-failure[k])的部分重複出現來表達

failure[k]為次長相同前綴後綴
如果我們不只想求最多，而且以0-base做為考量
，那可能的長度由大到小會是
failuer[k]、failure[failuer[k]-1]
、failure[failure[failuer[k]-1]-1]..
直到有值為0為止 */
int failure[MXN];
vector<int>ret;
void KMP(string& t, string& p){
    if (p.size() > t.size()) return;
    for (int i=1, j=failure[0]=-1; i<p.size(); ++i){
        while (j >= 0 && p[j+1] != p[i])
            j = failure[j];
        if (p[j+1] == p[i]) j++;
        failure[i] = j;
    }
    for (int i=0, j=-1; i<t.size(); ++i){
        while (j >= 0 && p[j+1] != t[i])
            j = failure[j];
        if (p[j+1] == t[i]) j++;
        if (j == p.size()-1){
            ret.push_back( i - p.size() + 1 );
            j = failure[j];
}   }   return ;}
```

7.2 馬拉車

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

7.3 回文樹

```
// len[s]是對應的回文長度
// num[s]是有幾個回文後綴
// cnt[s]是這個回文子字串在整個字串中的出現次數
// fail[s]是他長度次長的回文後綴，aba的fail是a
const int MXN = 1000010;
struct PalT{
  int nxt[MXN][26],fail[MXN],len[MXN];
  int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
  int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
  char s[MXN]={-1};
  int newNode(int l,int f){
    len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
    memset(nxt[tot],0,sizeof(nxt[tot]));
    diff[tot]=(l>0?l-len[f]:0);
    sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
    return tot++;
  }
  int getfail(int x){
    while(s[n-len[x]-1]!=s[n]) x=fail[x];
    return x;
  }
  int getmin(int v){
    dp[v]=fac[n-len[sfail[v]]-diff[v]];
    if(diff[v]==diff[fail[v]])
        dp[v]=min(dp[v],dp[fail[v]]);
    return dp[v]+1;
  }
  int push(){
    int c=s[n]-'a',np=getfail(lst);
    if(!(lst=nxt[np][c])){
      lst=newNode(len[np]+2,nxt[getfail(fail[np])][c]);
      nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
    }
    fac[n]=n;
    for(int v=lst;len[v]>0;v=sfail[v])
        fac[n]=min(fac[n],getmin(v));
    return ++cnt[lst],lst;
  }
  void init(const char *_s){
    tot=lst=n=0;
    newNode(0,1),newNode(-1,1);
    for(;_s[n];) s[n+1]=_s[n],++n,state[n-1]=push();
    for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
  }
}palt;
```

7.4 SA

```
const int N = 300010;
struct SA{
#define REP(i,n) for ( int i=0; i<(int)(n); i++ )
#define REP1(i,a,b) for ( int i=(a); i<=(int)(b); i++ )
  bool _t[N*2];
  int _s[N*2], _sa[N*2], _c[N*2], x[N], _p[N], _q[N*2],
       hei[N], r[N];
  int operator [] (int i){ return _sa[i]; }
  void build(int *s, int n, int m){
    memcpy(_s, s, sizeof(int) * n);
    sais(_s, _sa, _p, _q, _t, _c, n, m);
    mkhei(n);
  }
  void mkhei(int n){
    REP(i,n) r[_sa[i]] = i;
    hei[0] = 0;
    REP(i,n) if(r[i]) {
      int ans = i>0 ? max(hei[r[i-1]] - 1, 0) : 0;
      while(_s[i+ans] == _s[_sa[r[i]-1]+ans]) ans++;
      hei[r[i]] = ans;
    }
  }
  void sais(int *s, int *sa, int *p, int *q, bool *t,
      int *c, int n, int z){
    bool uniq = t[n-1] = true, neq;
    int nn = 0, nmxz = -1, *nsa = sa + n, *ns = s + n,
        lst = -1;
#define MS0(x,n) memset((x),0,n*sizeof(*(x)))
#define MAGIC(XD) MS0(sa, n); \
    memcpy(x, c, sizeof(int) * z); \
    XD; \
    memcpy(x + 1, c, sizeof(int) * (z - 1)); \
    REP(i,n) if(sa[i] && !t[sa[i]-1]) sa[x[s[sa[i
        ]-1]]++] = sa[i]-1; \
    memcpy(x, c, sizeof(int) * z); \
    for(int i = n - 1; i >= 0; i--) if(sa[i] && t[sa[i
        ]-1]) sa[--x[s[sa[i]-1]]] = sa[i]-1;
    MS0(c, z);
    REP(i,n) uniq &= ++c[s[i]] < 2;
    REP(i,z-1) c[i+1] += c[i];
    if (uniq) { REP(i,n) sa[--c[s[i]]] = i; return; }
    for(int i = n - 2; i >= 0; i--) t[i] = (s[i]==s[i
        +1] ? t[i+1] : s[i]<s[i+1]);
    MAGIC(REP1(i,1,n-1) if(t[i] && !t[i-1]) sa[--x[s[i
        ]]]=p[q[i]=nn++]=i);
    REP(i, n) if (sa[i] && t[sa[i]] && !t[sa[i]-1]) {
      neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
          [i])*sizeof(int));
      ns[q[lst=sa[i]]]=nmxz+=neq;
    }
    sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmxz
       + 1);
    MAGIC(for(int i = nn - 1; i >= 0; i--) sa[--x[s[p[
        nsa[i]]]]] = p[nsa[i]]);
  }
}sa;
int H[ N ], SA[ N ];
void suffix_array(int* ip, int len) {
  // should padding a zero in the back
  // ip is int array, len is array length
  // ip[0..n-1] != 0, and ip[len] = 0
  ip[len++] = 0;
  sa.build(ip, len, 128);
  for (int i=0; i<len; i++) {
    H[i] = sa.hei[i + 1];
    SA[i] = sa._sa[i + 1];
  }
  // resulting height, sa array \in [0,len)
}
```

7.5 SAM

```
// any path start from root forms a substring of S
// occurrence of P : iff SAM can run on input word P
// number of different substring : ds[1]-1
// total length of all different substring : dsl[1]
// max/min length of state i : mx[i]/mx[mom[i]]+1
// assume a run on input word P end at state i:
// number of occurrences of P : cnt[i]
// first occurrence position of P : fp[i]-|P|+1
// all position of P : fp of "dfs from i through rmom"
const int MXM = 1000010;
struct SAM{
  int tot, root, lst, mom[MXM], mx[MXM]; //ind[MXM]
  int nxt[MXM][33]; //cnt[MXM],ds[MXM],dsl[MXM],fp[MXM]
  // bool v[MXM]
  int newNode(){
    int res = ++tot;
    fill(nxt[res], nxt[res]+33, 0);
    mom[res] = mx[res] = 0; //cnt=ds=dsl=fp=v=0
    return res;
  }
  void init(){
    tot = 0;
    root = newNode();
    lst = root;
  }
  void push(int c){
    int p = lst;
    int np = newNode(); //cnt[np]=1
    mx[np] = mx[p]+1; //fp[np]=mx[np]-1
    for(; p && nxt[p][c] == 0; p = mom[p])
      nxt[p][c] = np;
    if(p == 0) mom[np] = root;
    else{
      int q = nxt[p][c];
      if(mx[p]+1 == mx[q]) mom[np] = q;
      else{
        int nq = newNode(); //fp[nq]=fp[q]
        mx[nq] = mx[p]+1;
        for(int i = 0; i < 33; i++)
          nxt[nq][i] = nxt[q][i];
        mom[nq] = mom[q];
        mom[q] = nq;
        mom[np] = nq;
        for(; p && nxt[p][c] == q; p = mom[p])
          nxt[p][c] = nq;
    } }
    lst = np;
  }
  void calc(){
    calc(root);
    iota(ind,ind+tot,1);
    sort(ind,ind+tot,[&](int i,int j){return mx[i]<mx[j
        ];});
    for(int i=tot-1;i>=0;i--)
    cnt[mom[ind[i]]]+=cnt[ind[i]];
  }
  void calc(int x){
    v[x]=ds[x]=1;dsl[x]=0; //rmom[mom[x]].push_back(x);
    for(int i=1;i<=26;i++){
      if(nxt[x][i]){
        if(!v[nxt[x][i]]) calc(nxt[x][i]);
        ds[x]+=ds[nxt[x][i]];
        dsl[x]+=ds[nxt[x][i]]+dsl[nxt[x][i]];
  } } }
  void push(const string& str){
    for(int i = 0; i < str.size() ; i++)
      push(str[i]-'a'+1);
  }
} sam;
```

7.6 樹哈希

```
map<vector<int>,int>id;
```

```
ll dfs(int u){
    vector<ll> h;
    for(ll child : edge[u]){
        h.push_back(dfs(child));
    }
    sort(h.begin(), h.end());
    if(id.count(h))return id[h];
    else return id[h]=id.size();
}
```

7.7 trie

```
//01 bitwise trie
struct trie{
    trie *nxt[2];  // 差別
    int cnt;    //紀錄有多少個數字以此節點結尾
    int sz;     //有多少數字的前綴包括此節點
    trie():cnt(0),sz(0){
        memset(nxt,0,sizeof(nxt));
    }
};
//創建新的字典樹
trie *root;
void insert(int x){
    trie *now = root;  // 每次從根節點開始
    for(int i=22;i>=0;i--){ // 從最高位元開始往低位元走
        now->sz++;
        //cout<<(x>>i&1)<<endl;
        if(now->nxt[x>>i&1] == NULL){ //判斷當前第 i 個
            位元是 0 還是 1
            now->nxt[x>>i&1] = new trie();
        }
        now = now->nxt[x>>i&1];  //走到下一個位元
    }
    now->cnt++;
    now->sz++;
}
```

7.8 Z-value

```
int z[MAXN];
void Z_value(const string& s) { //z[i] = lcp(s[1...],s[
    i...])
  int i, j, left, right, len = s.size();
  left=right=0; z[0]=len;
  for(i=1;i<len;i++) {
    j=max(min(z[i-left],right-i),0);
    for(;i+j<len&&s[i+j]==s[j];j++);
    z[i]=j;
    if(i+z[i]>right) {
      right=i+z[i];
      left=i;
}   }   }
```

7.9 minRotation

```
//rotate(begin(s),begin(s)+minRotation(s),end(s))
int minRotation(string s) {
  int a = 0, N = s.size(); s += s;
  for(int b=0;b<N;b++)
  for(int k=0;k<N;k++){
    if(a+k == b || s[a+k] < s[b+k])
      {b += max(0, k-1); break;}
    if(s[a+k] > s[b+k]) {a = b; break;}
  } return a;
}
```

8 DP

8.1 數位 dp

```
ll dp[MXN_BIT][PRE_NUM][LIMIT][F0];//字串位置, 根據題目
    的值, 是否上界, 前導0
ll dfs(int i,int pre, bool lim, bool f0, const string&
    str){
    if(v[i][pre][f0][lim])  return dp[i][pre][f0][lim];
    v[i][pre][f0][lim] = true;

    if(i == str.size())
        return dp[i][pre][f0][lim] = 1;

    ll ret = 0, h = lim ? str[i] : '9';
    for(int j='0'; j<=h; j++){
        if(abs(j-pre)>=2 || f0){
            ret += dfs(i+1, j, j==h && lim, f0 && j=='0
                ', str);
    }}
    return dp[i][pre][f0][lim] = ret;
}
```

8.2 SOS dp

```
for(int i = 0; i<(1<<N); ++i)
  F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<
    N); ++mask){
  if(mask & (1<<i))
    F[mask] += F[mask^(1<<i)];
}
```

8.3 p-median

```
void p_Median(){
    for (int i=1; i<=N; ++i)
        for (int j=i; j<=N; ++j){
            m = (i+j)/2,d[i][j] = 0;        // m是中位
                數，d[i][j]為距離的總和
            for (int k=i; k<=j; ++k) d[i][j] += abs(arr
                [k] - arr[m]);
        }
    for (int p=1; p<=P; ++p)
        for (int n=1; n<=N; ++n){
            dp[p][n] = 1e9;
            for (int k=p; k<=n; ++k)
                if (dp[p-1][k-1] + d[k][n] < dp[p][n]){
                    dp[p][n] = dp[p-1][k-1] + d[k][n];
                    r[p][n] = k;    // 從第k個位置往右
                        到第j個位置
                }
        }
}
```

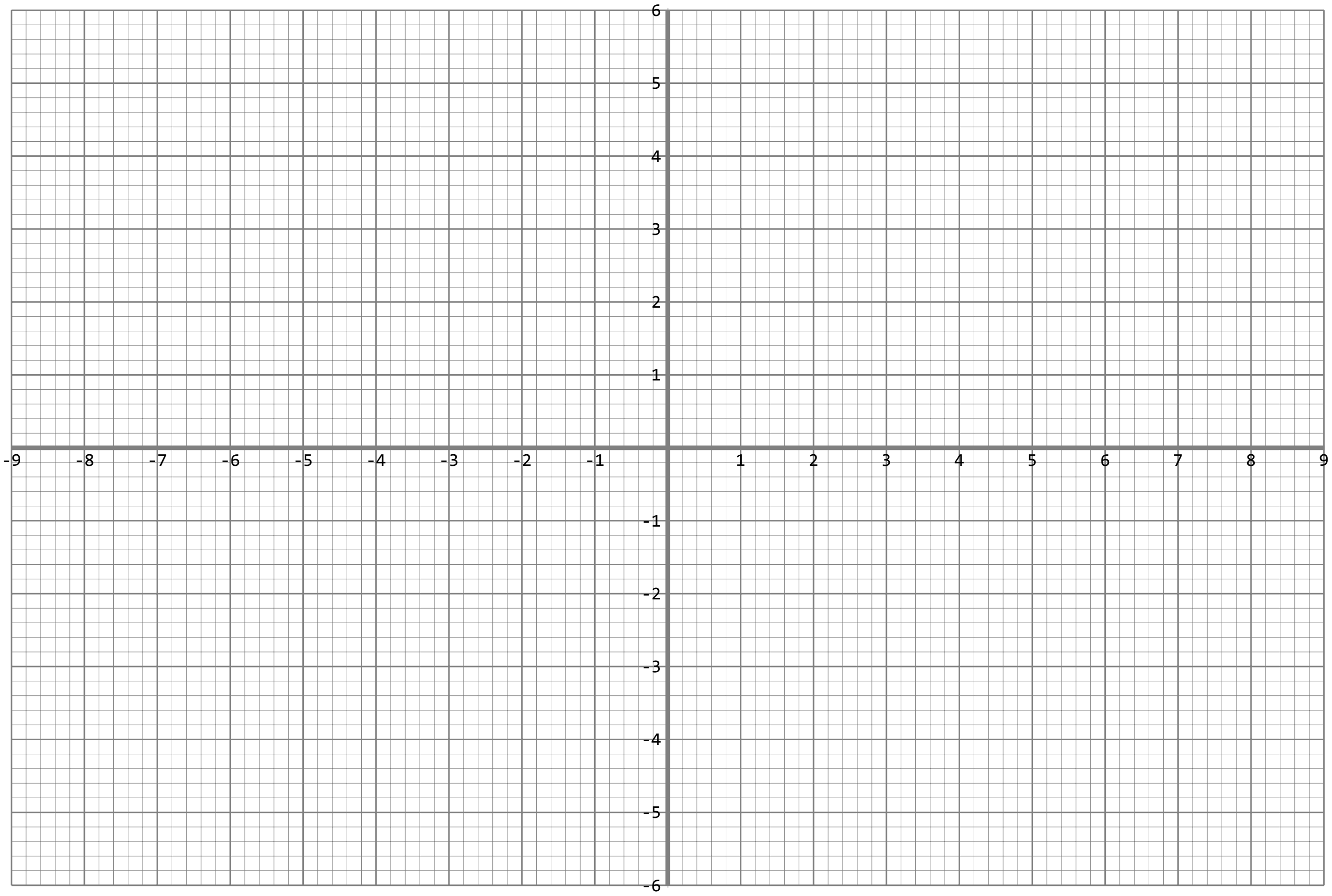
9 Other

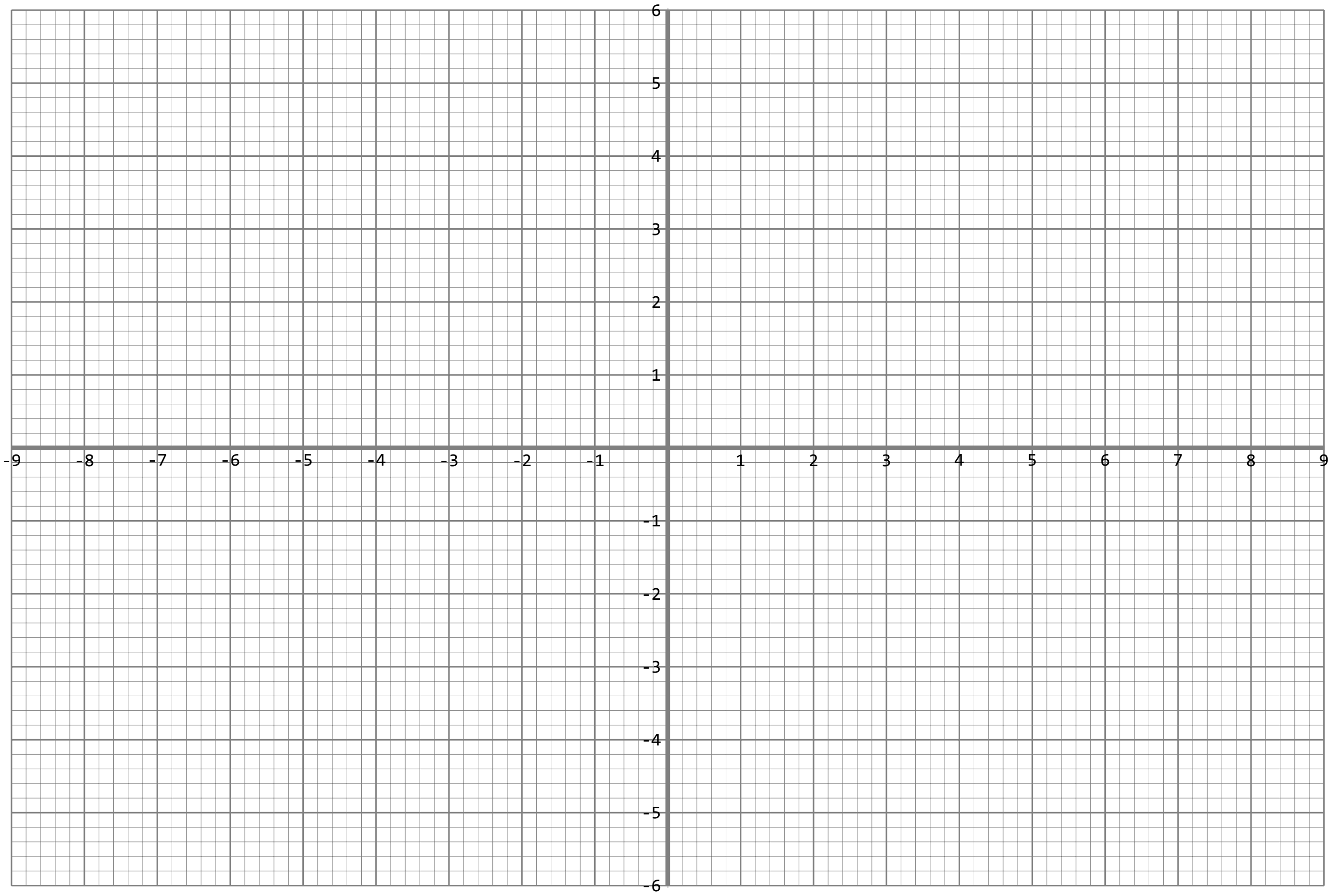
9.1 黑魔法、名次樹

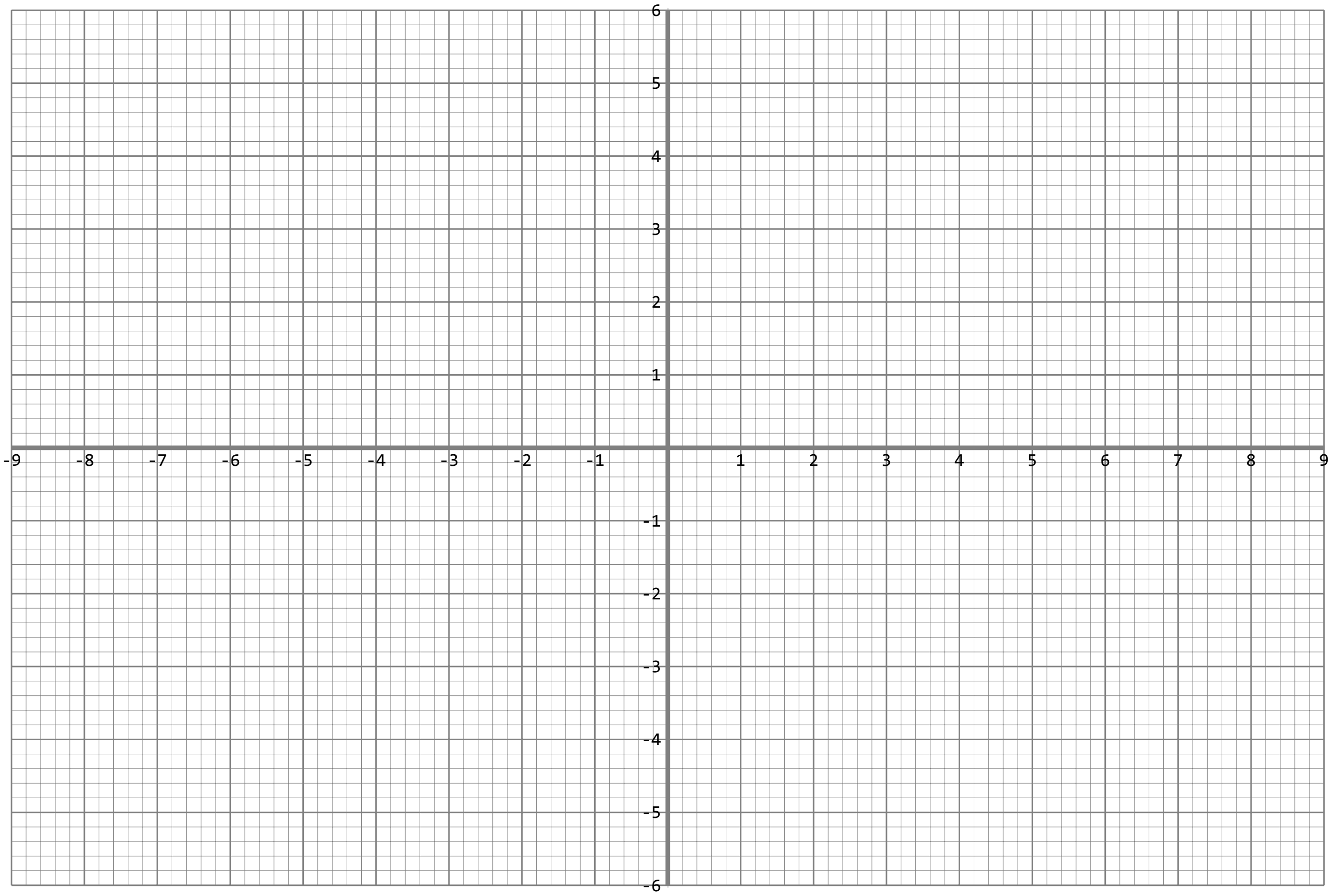
```
#include <bits/extc++.h>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> set_t;
#include <ext/pb_ds/assoc_container.hpp>
typedef cc_hash_table<int,int> umap_t;
typedef priority_queue<int> heap;
#include<ext/rope>
using namespace __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 );
}
```

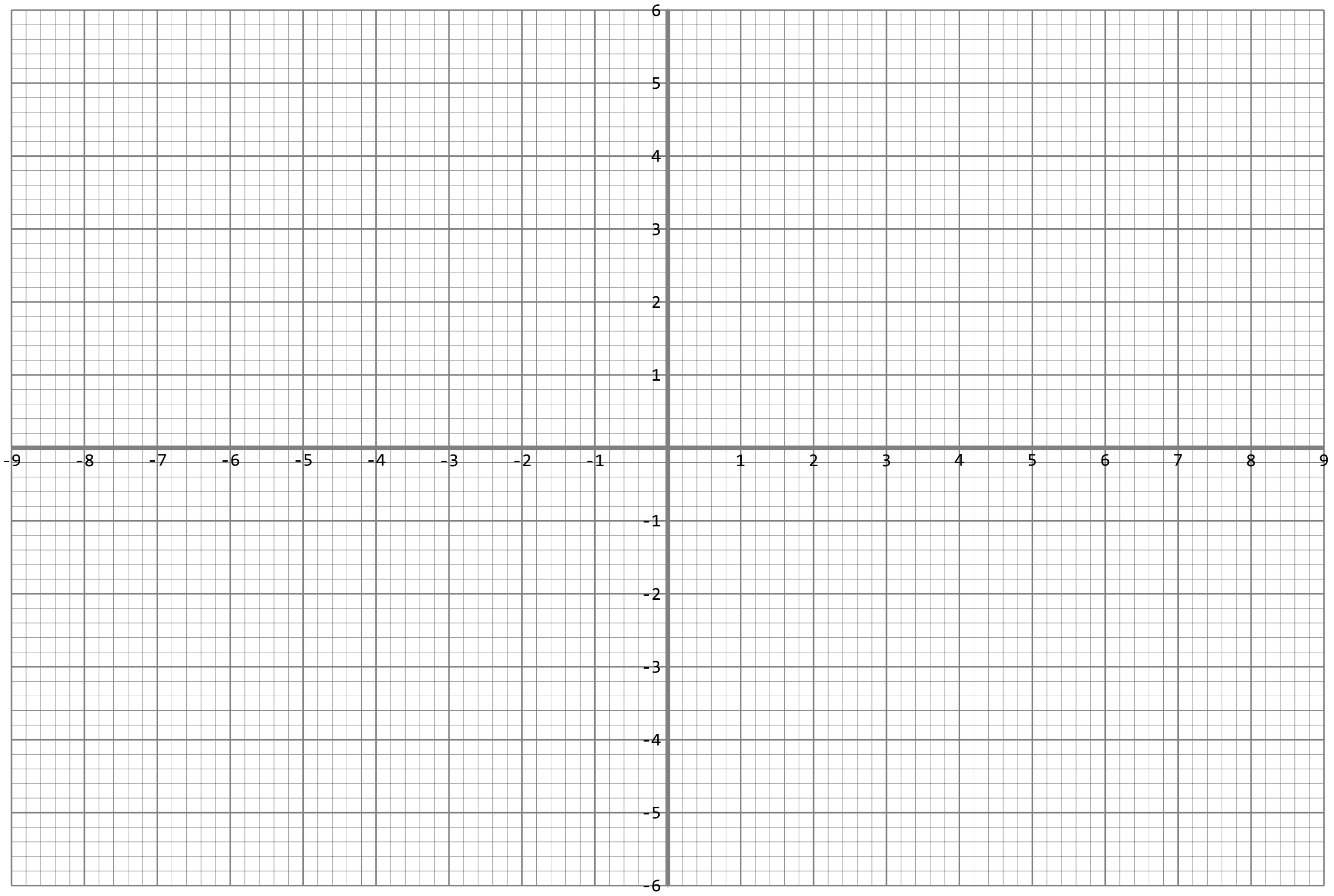
9.2 Hilbert curve

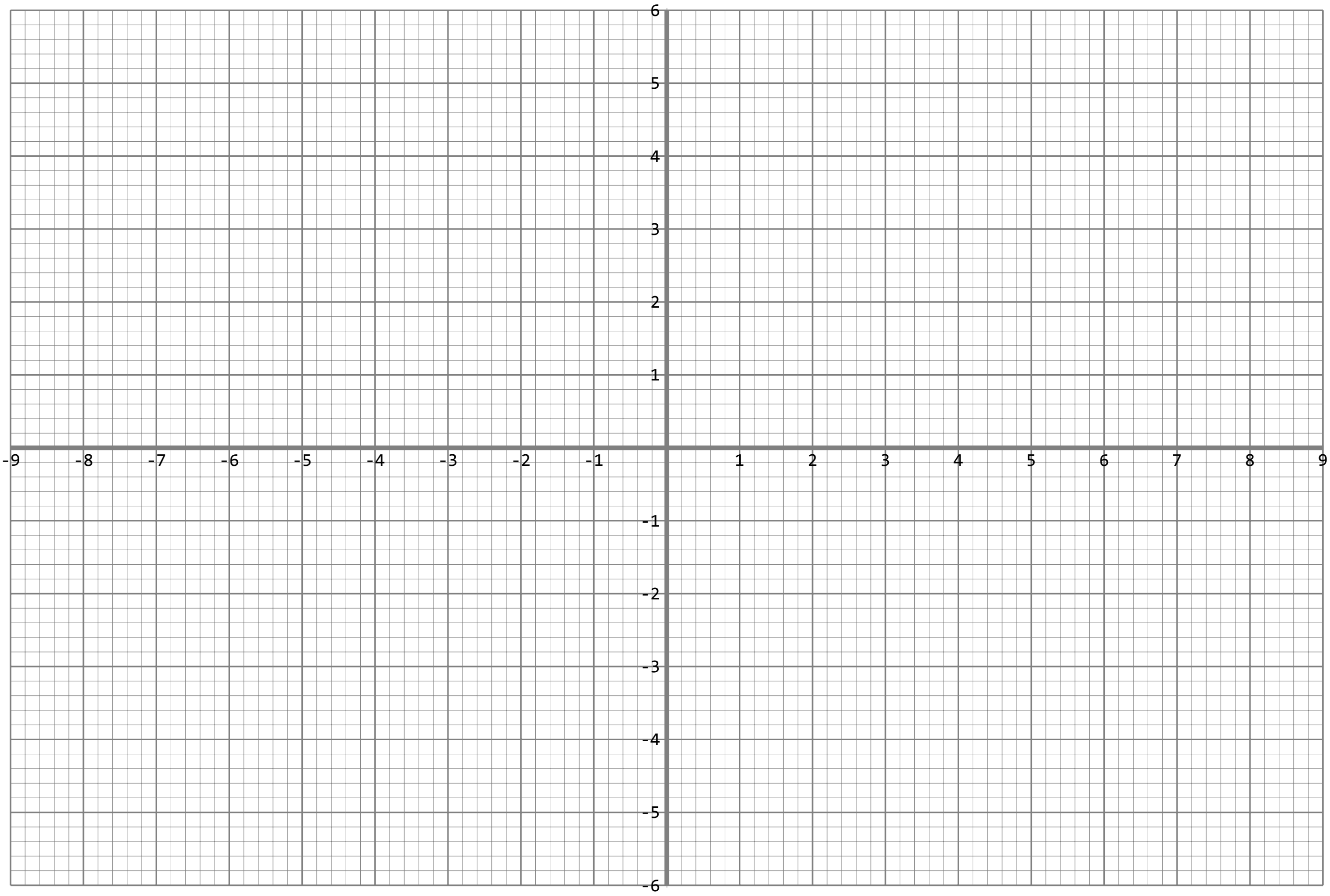
```
long long hilbert(int n,int x,int y){
  long long res=0;
  for(int s=n/2;s;s>>=1){
    int rx=(x&s)>0,ry=(y&s)>0; res+=s*1ll*s*((3*rx)^ry)
        ;
    if(ry==0){ if(rx==1) x=s-1-x,y=s-1-y; swap(x,y); }
  }
  return res;
}
```

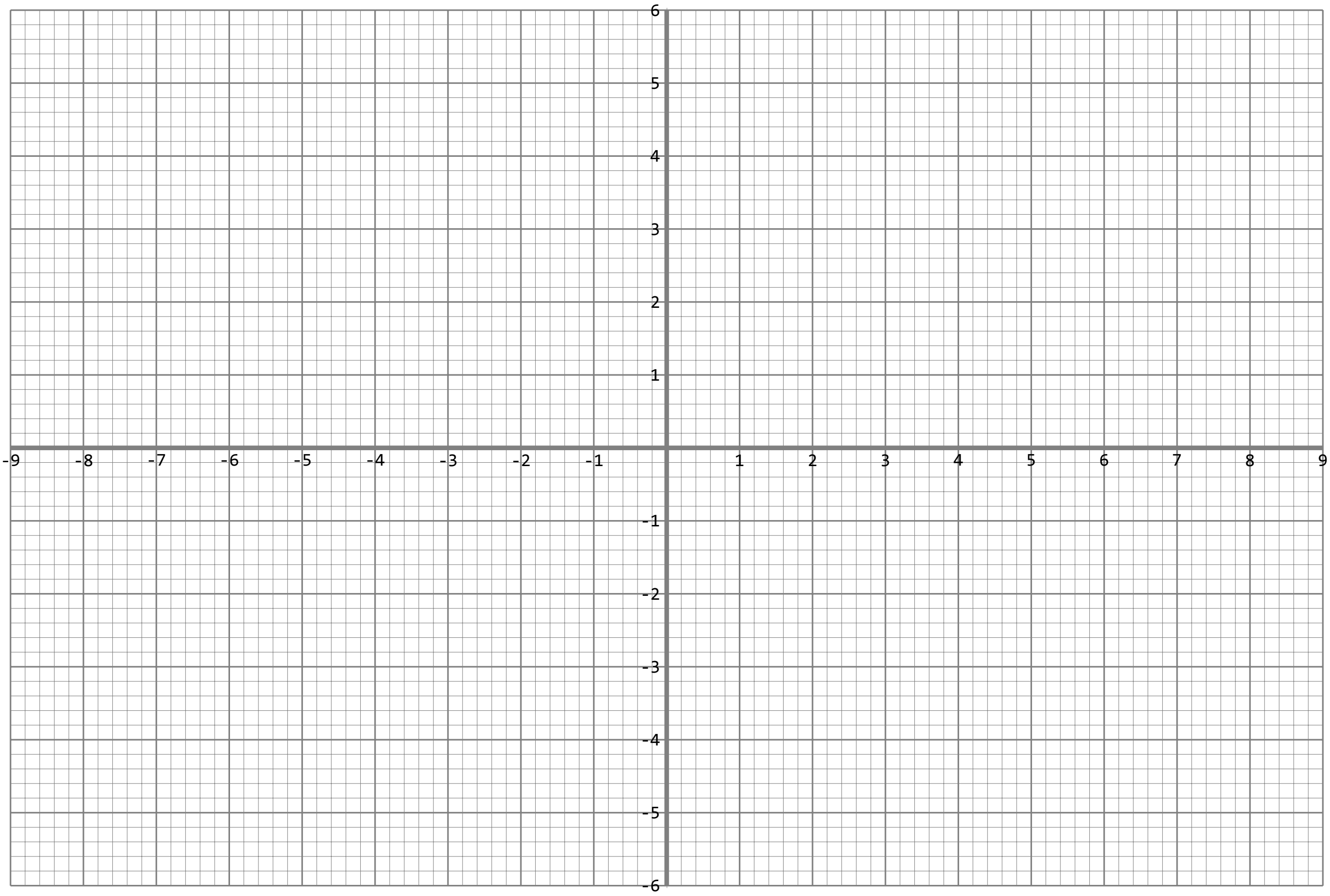

6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6

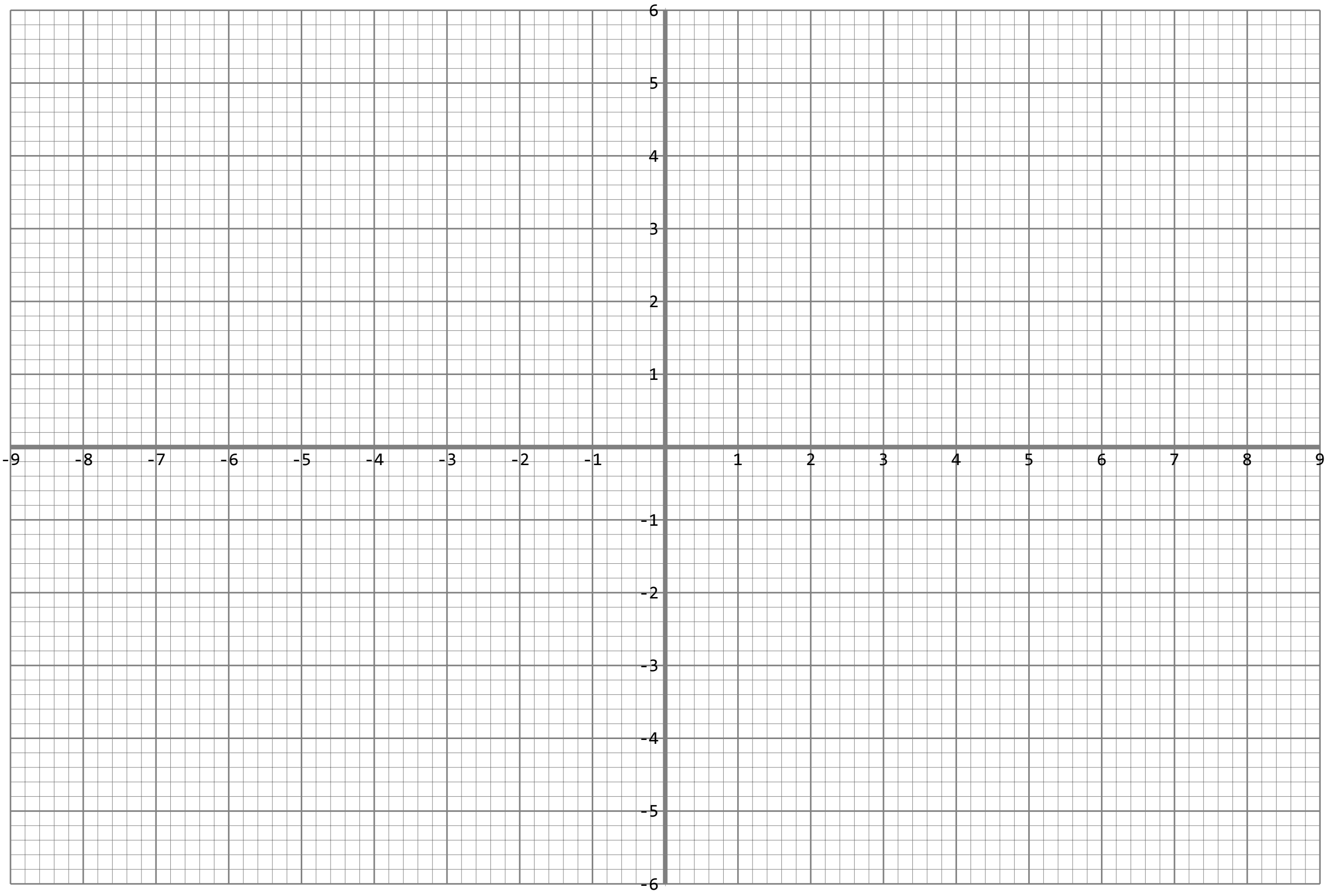

6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6

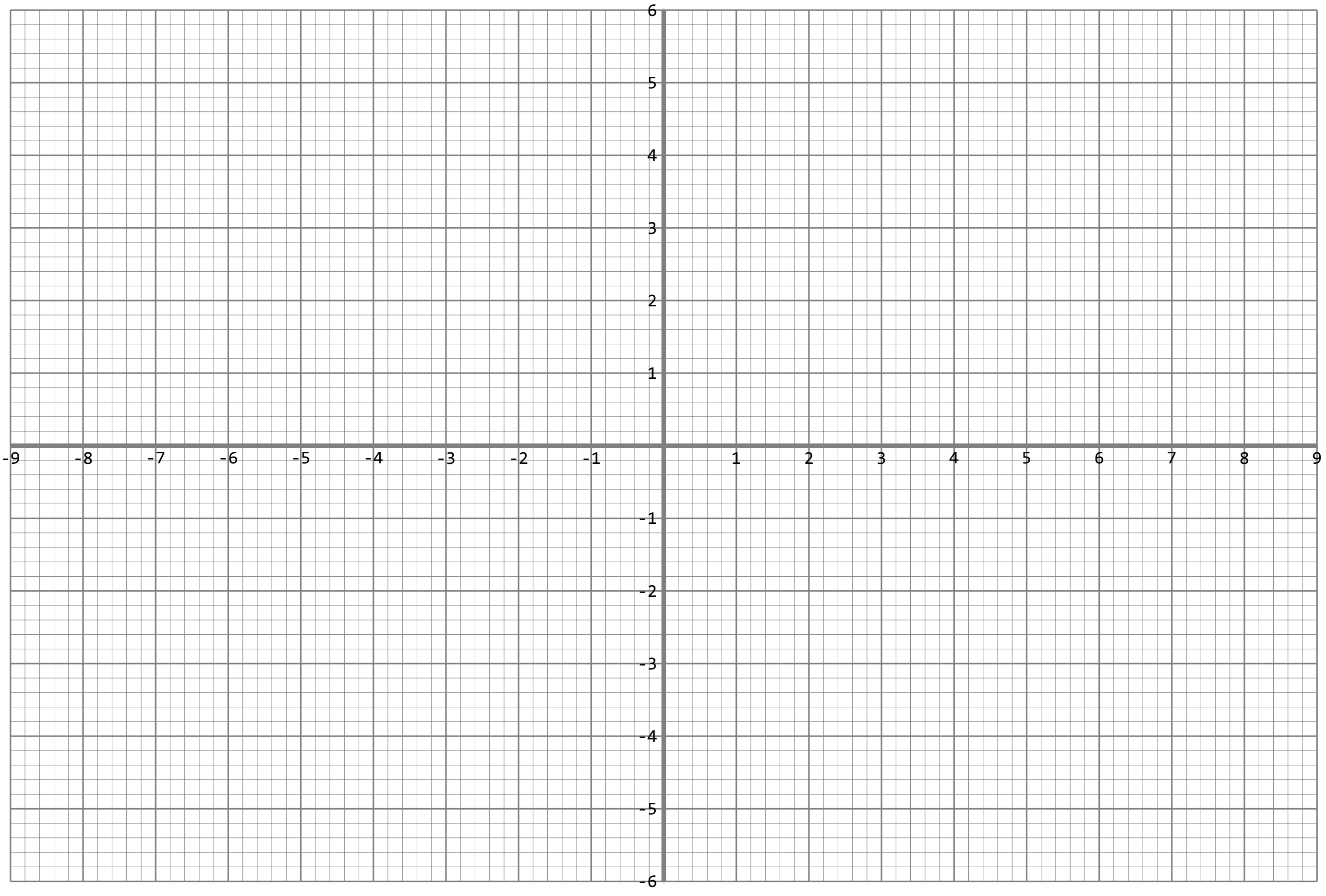

6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6

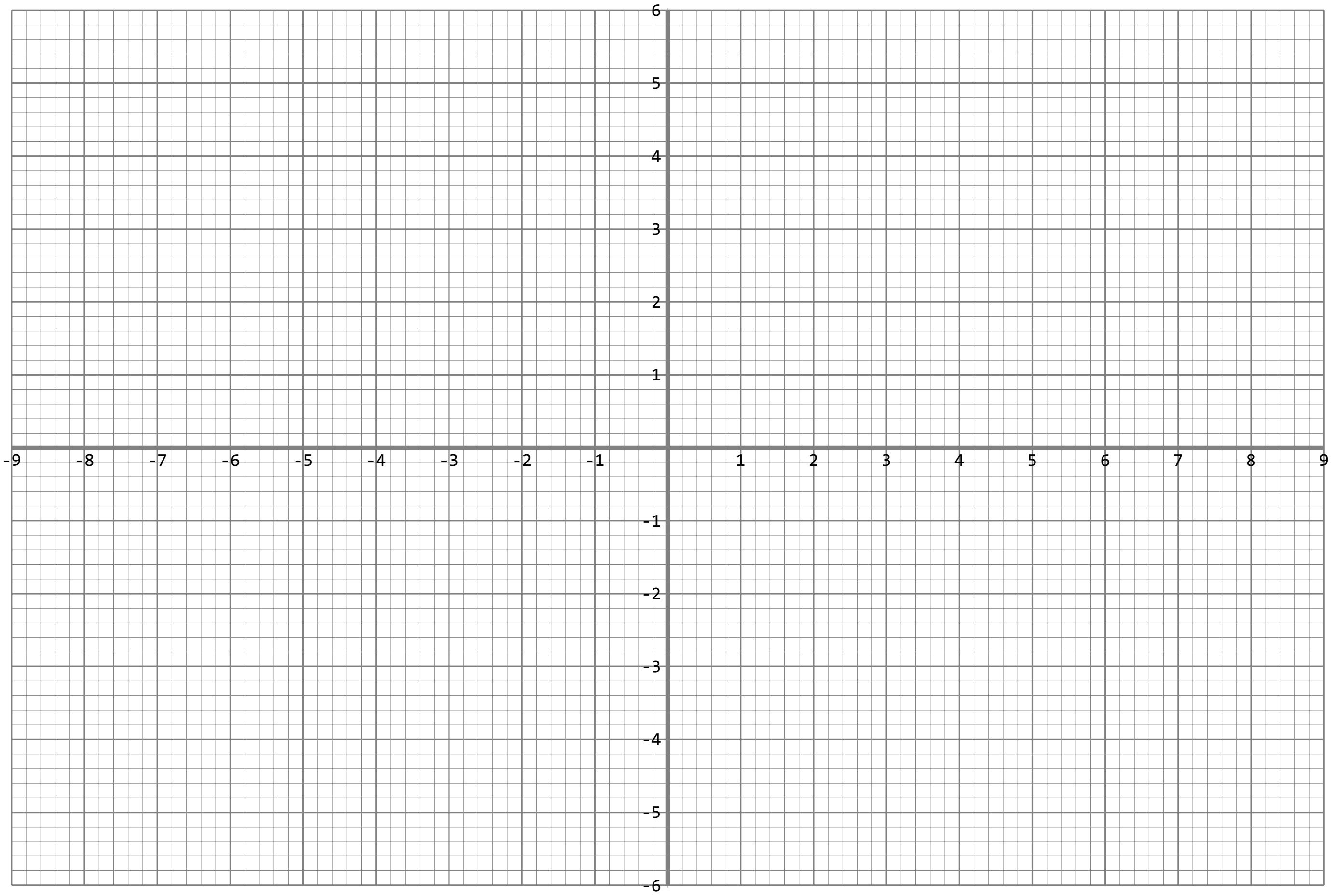

6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6

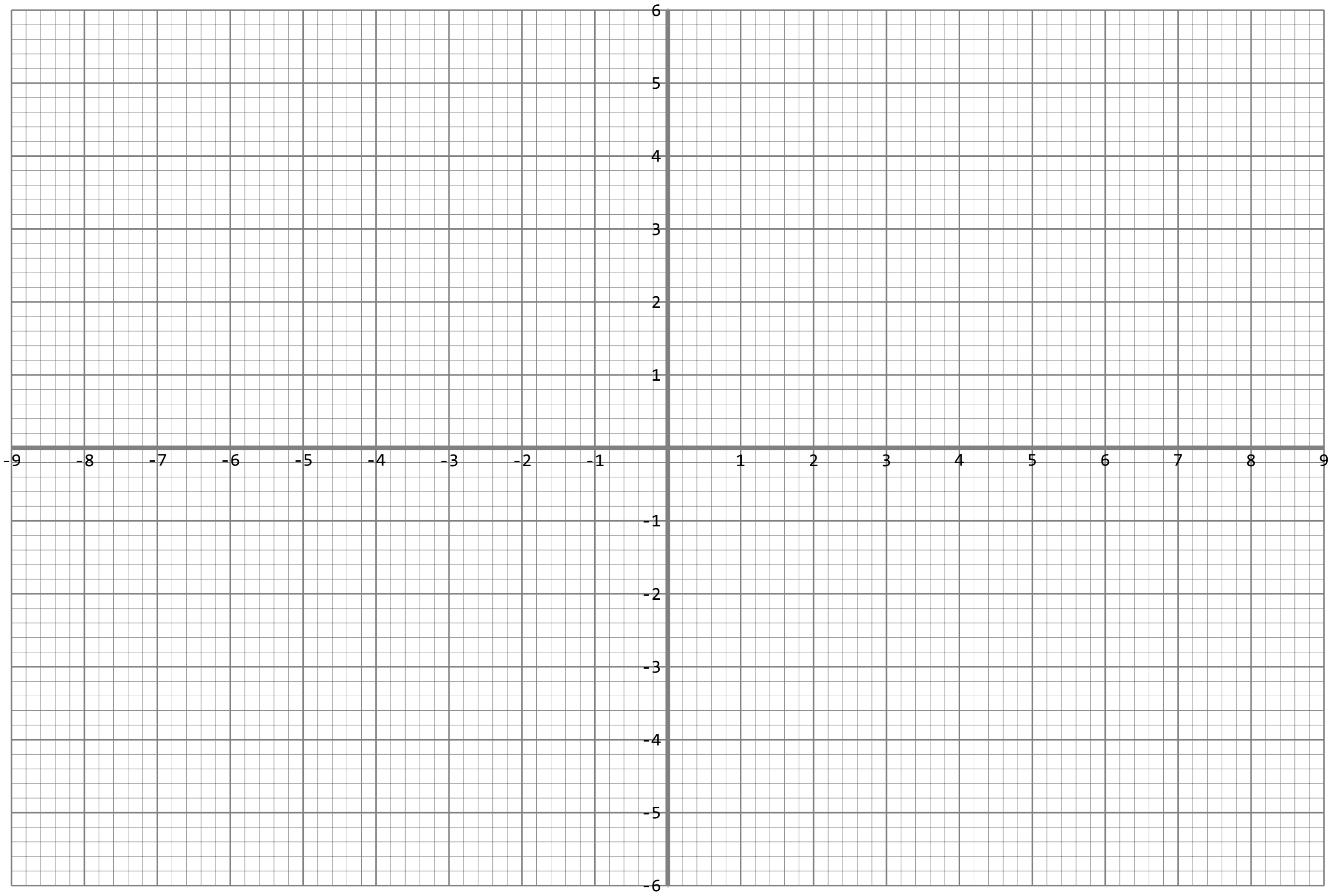

6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6

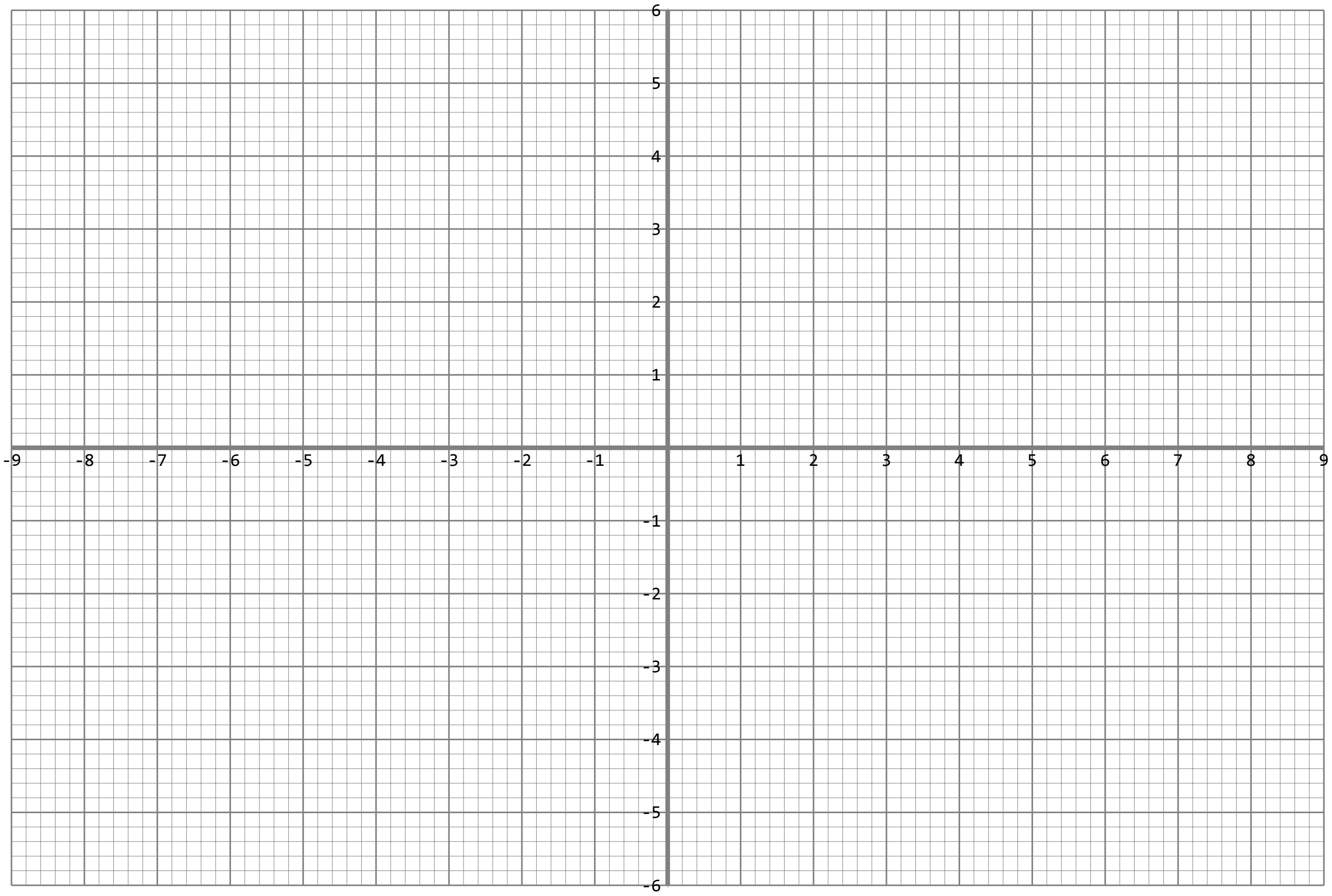

6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6

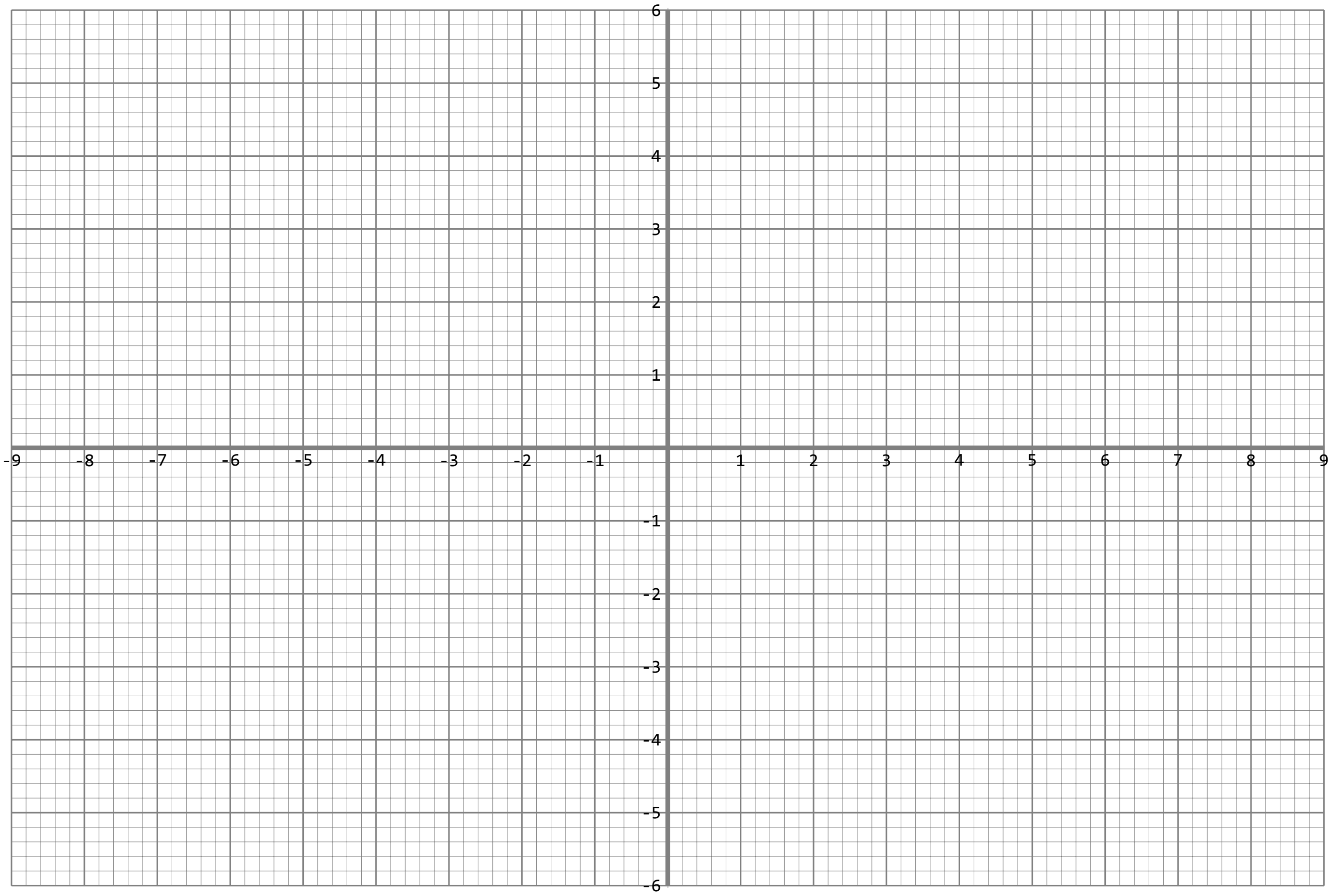

6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6

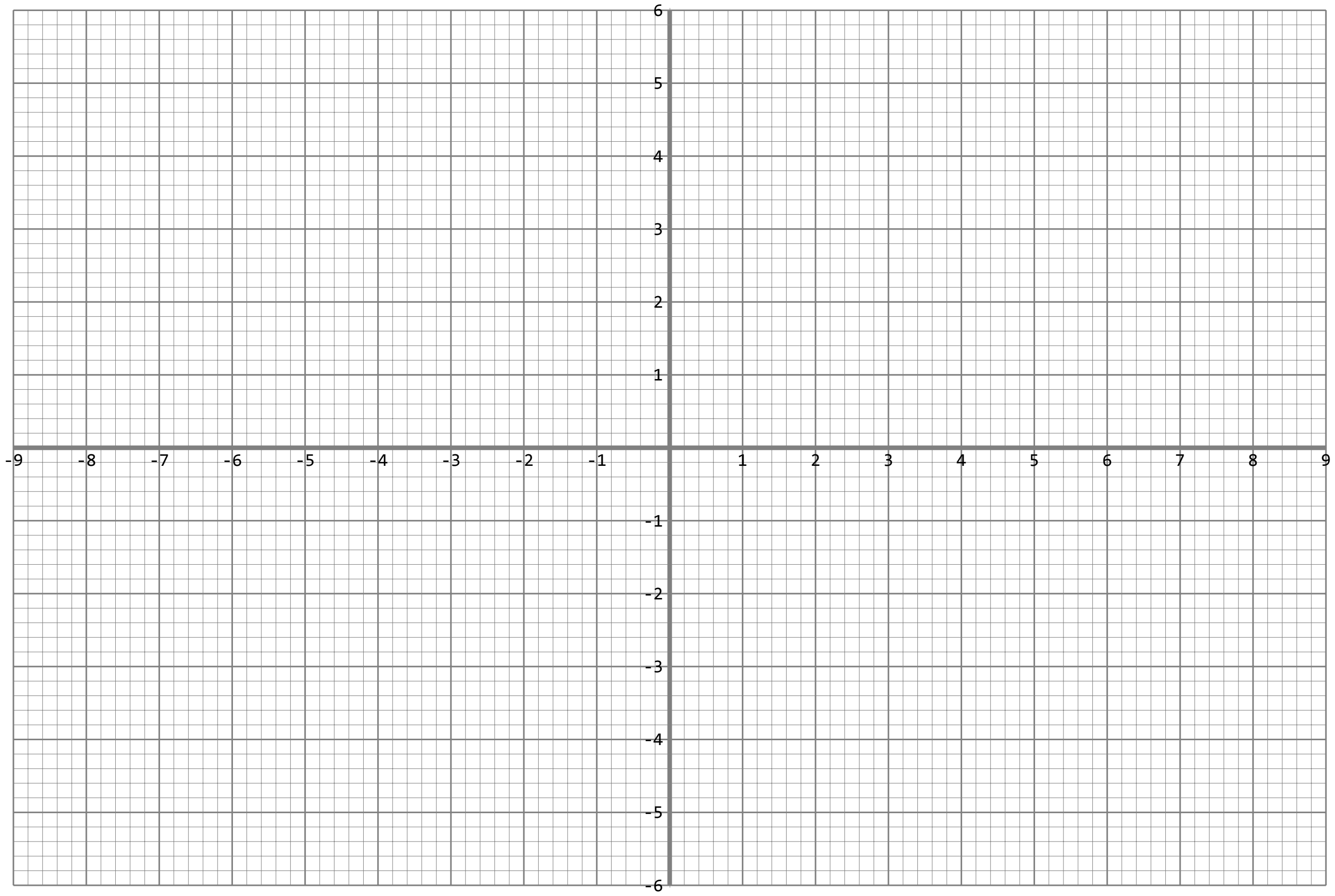

6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6

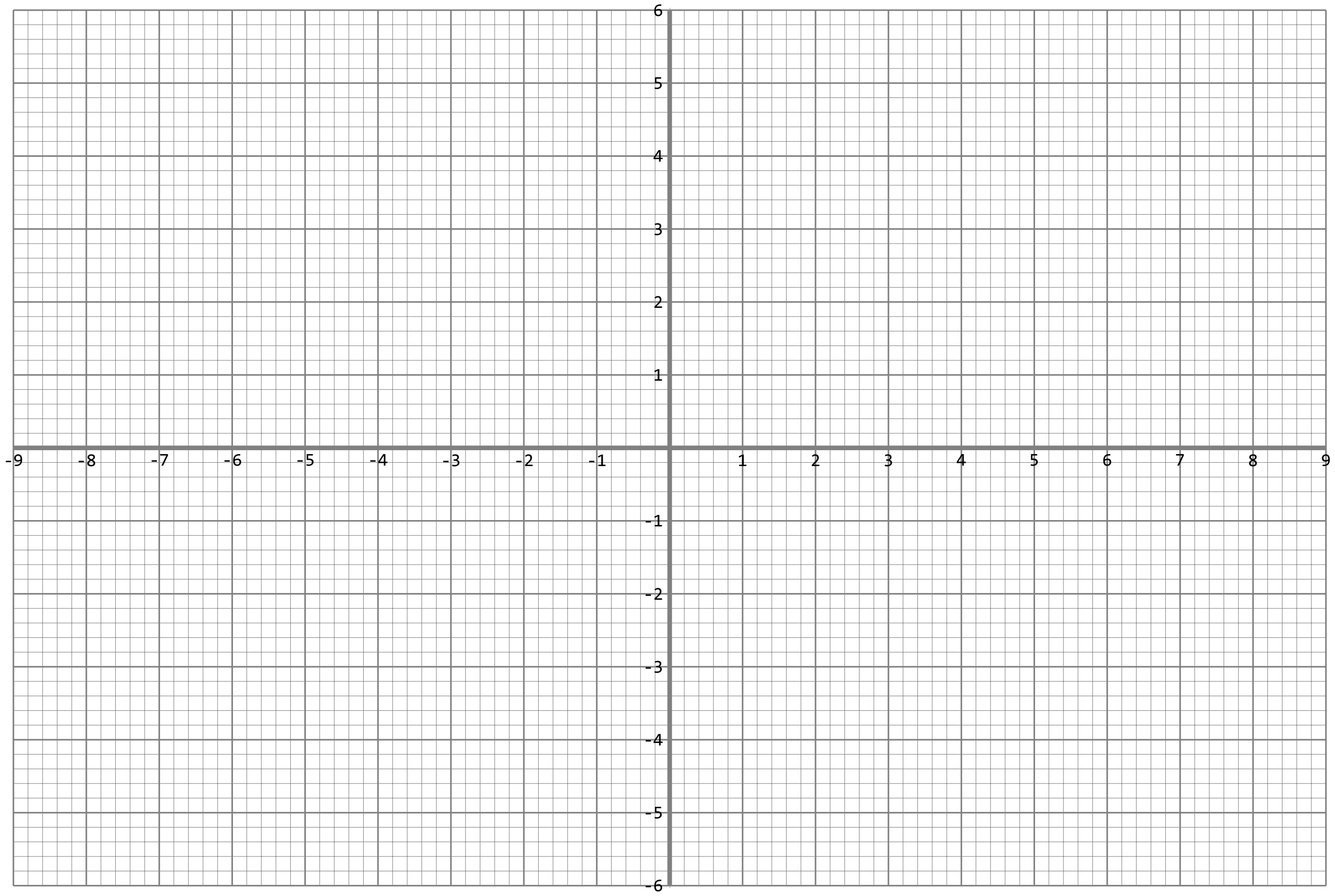

6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6

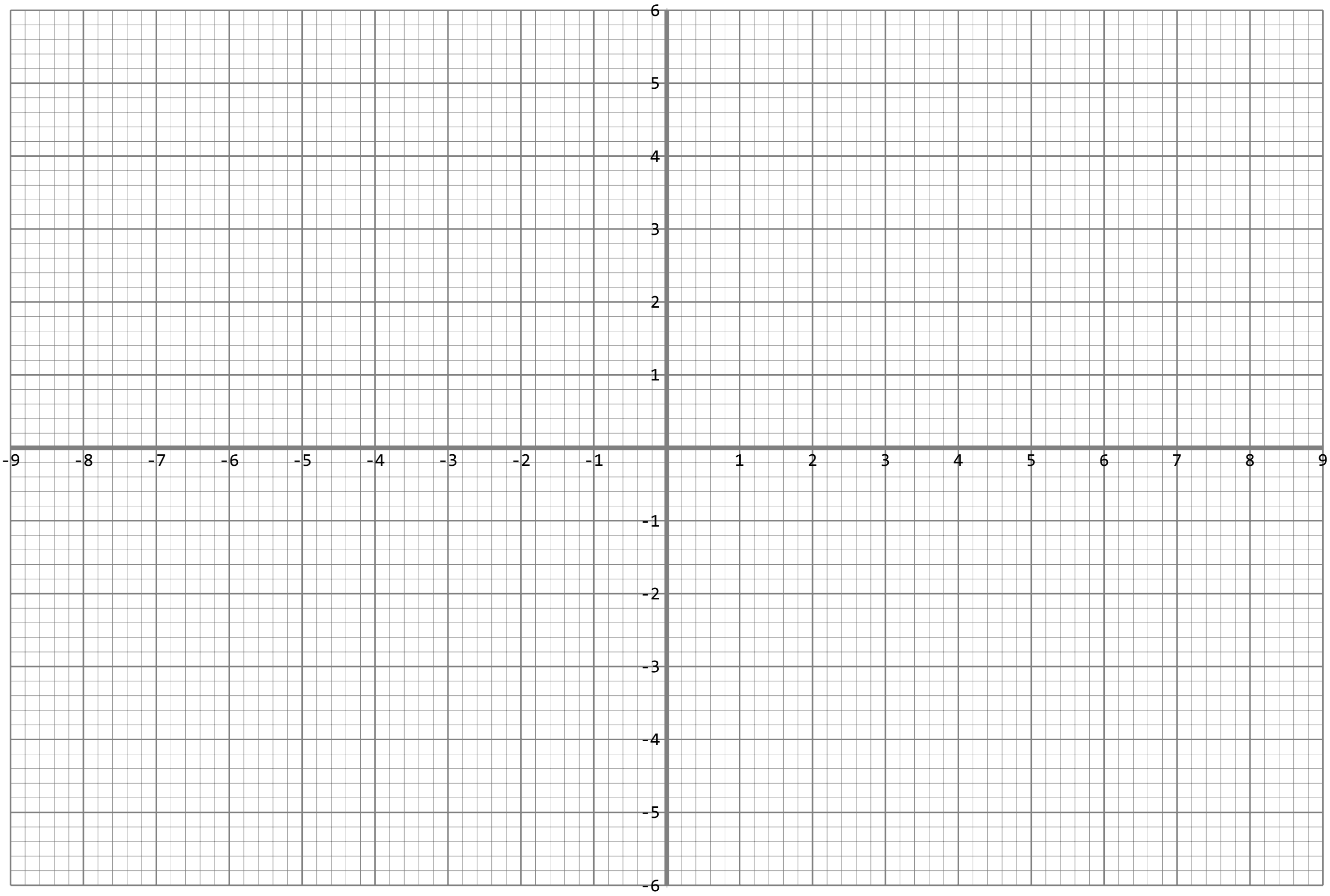

6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6

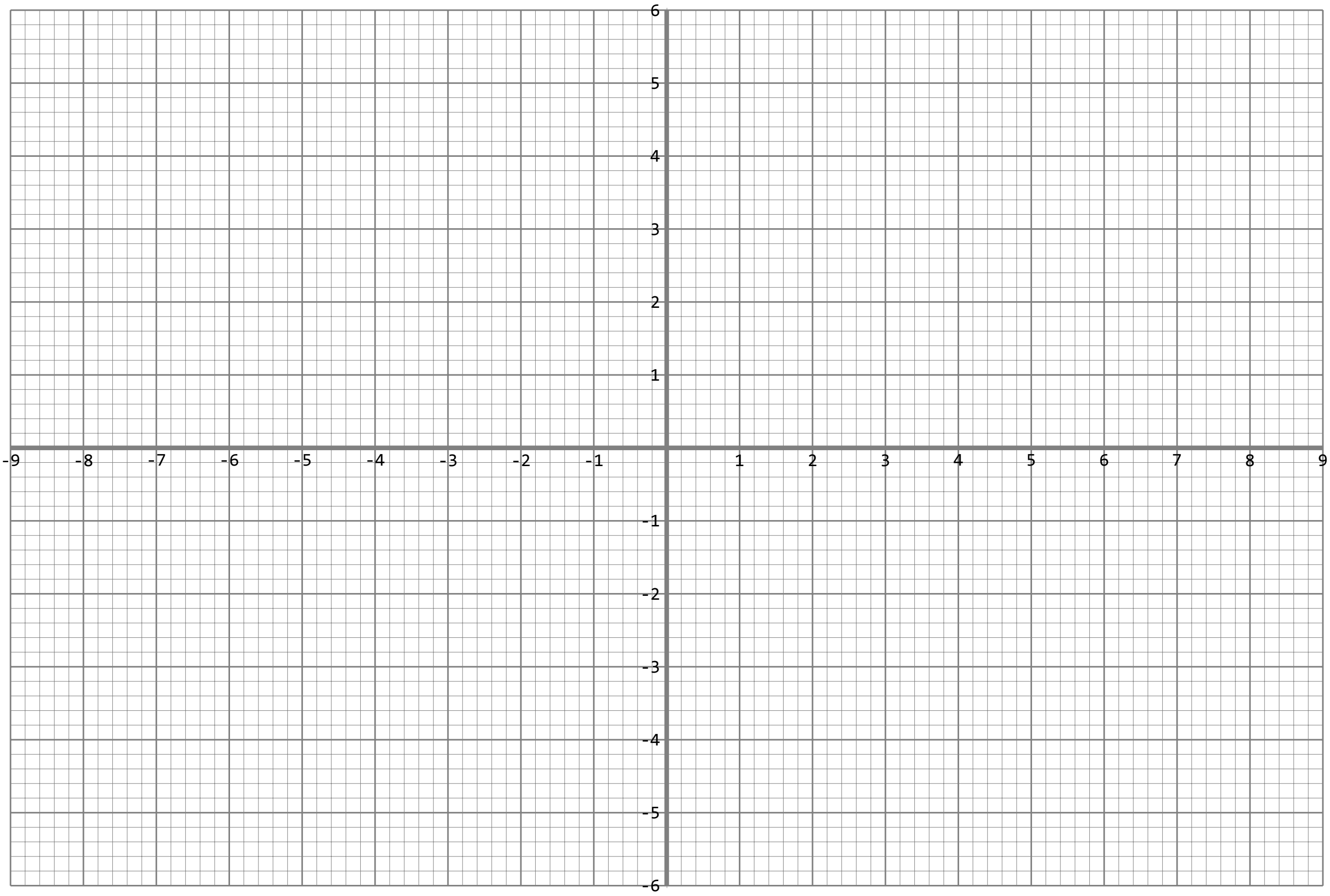

6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6

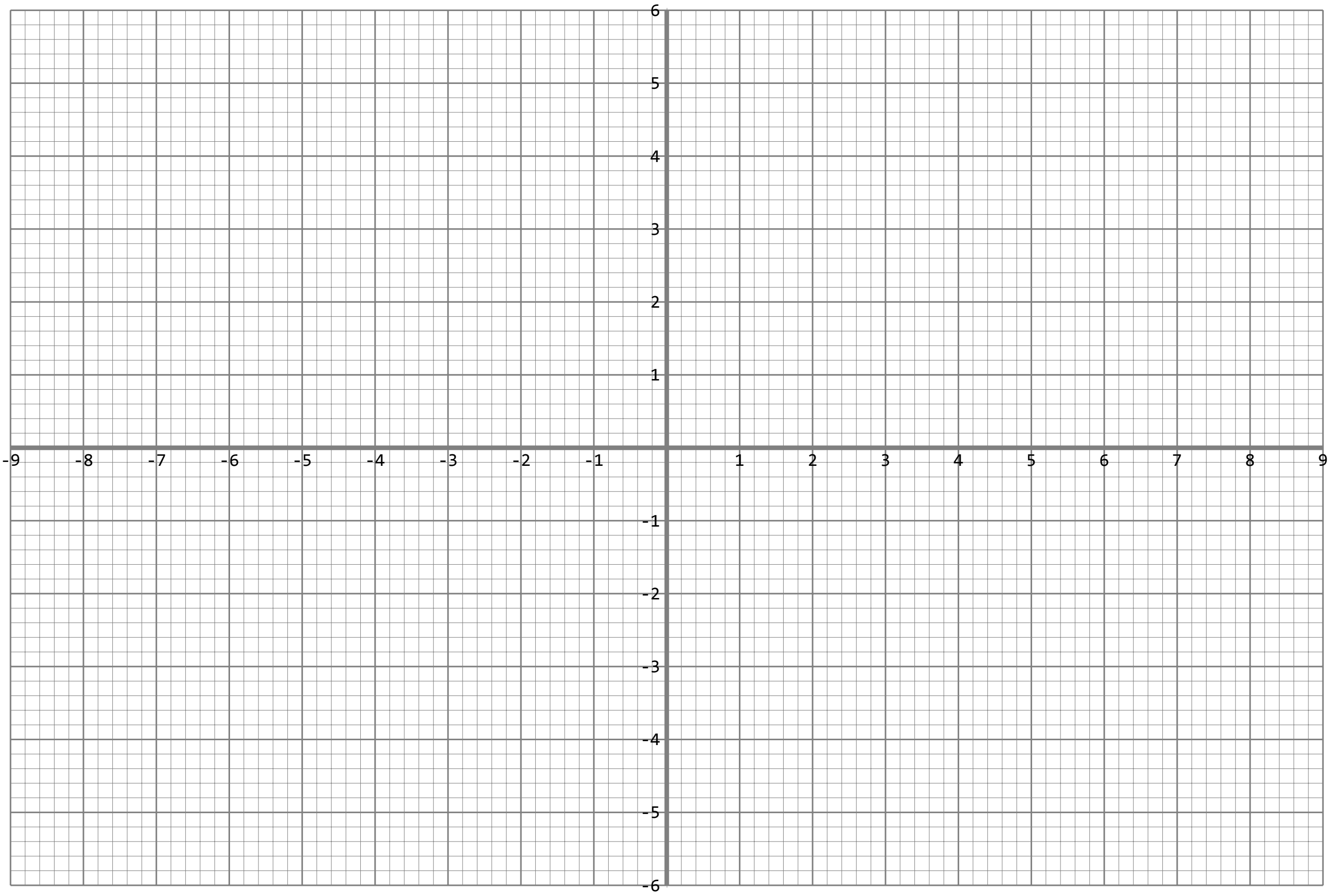

6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6

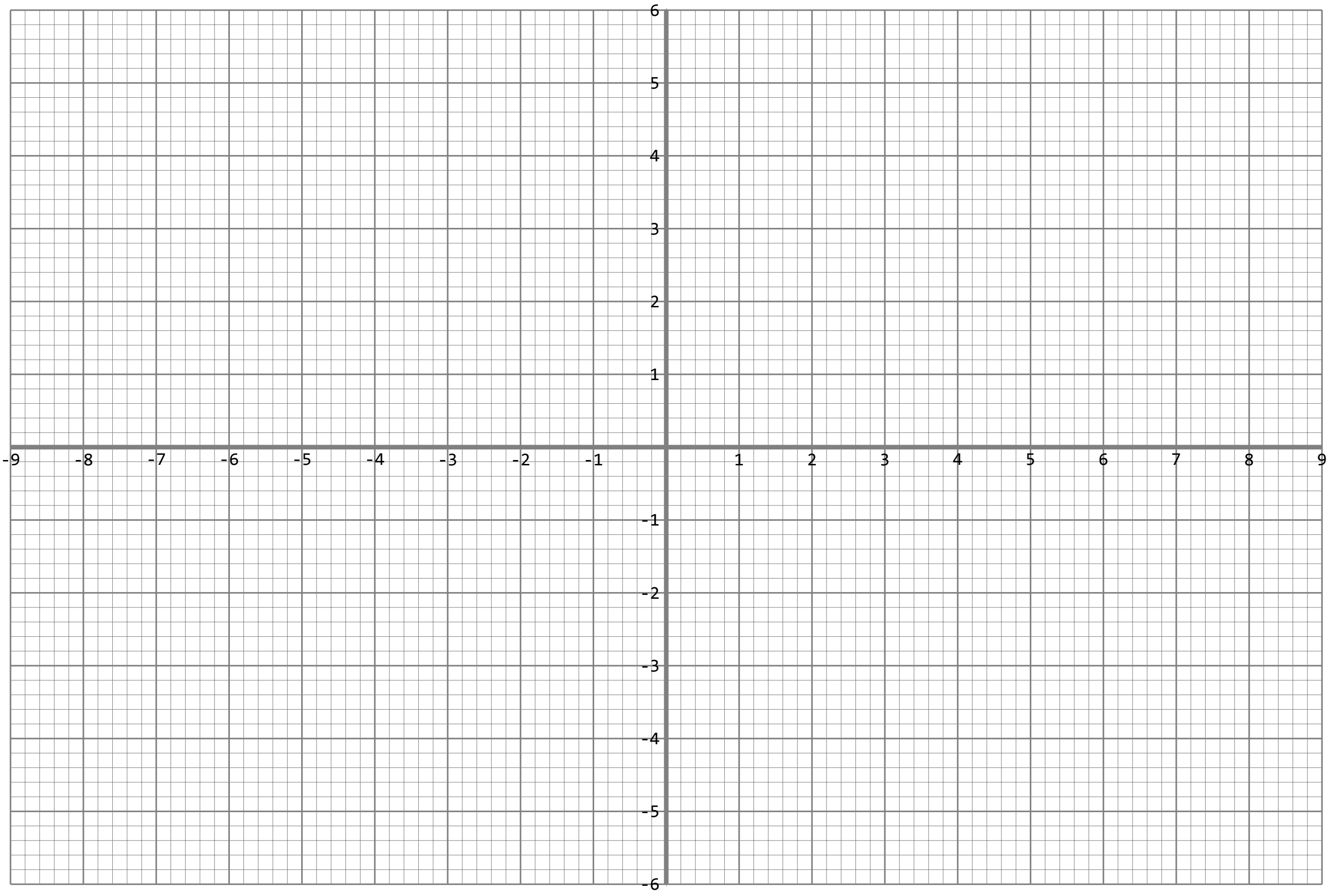

6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6


6
5
4
3
2
1
-9 -8 -7 -6 -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9
-1
-2
-3
-4
-5
-6


6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6


6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6


6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6


6
5
4
3
2
1
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
-1
-2
-3
-4
-5
-6