Cheatsheet I See the One

Daftar isi

Template

```
#include <bits/stdc++.h>

using namespace std;
#define inf 1000000000
#define unvisited -1
#define visited 1
#define eps 1e-9
#define pb push_back
#define pi acos(-1.0)
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> ii;
typedef vector<ii> vii;

int main() {

  return 0;
}
```

Segment Tree (+lazy propagation)

```
class SegmentTree{
  private:
    vi st,lazy;
    int n;
  public:
    void build(int p, int l, int r){
      if(l==r){
        st[p] = 0;
        return;
      }
      int mid = (l+r)/2;
      build(left(p),l,mid);
      build(right(p),mid+1,r);
      st[p] = st[left(p)] + st[right(p)];
    }

    SegmentTree(int n){
      this->n = n;
      st.assign(4*n,0);
      lazy.assign(4*n,0);
      build(1,1,n);
    }

    void update(int p, int l, int r, int i,
int j,ll v){
      if(lazy[p]!=0){
        st[p]+=(r-l+1)*lazy[p];
        if(l!=r){       lazy[left(p)]
+=lazy[p];
          lazy[right(p)]+=lazy[p];
        }
        lazy[p] = 0;
      }
      if(i>r || j < l){return;}
      if(l>=i && r<=j){
        st[p] += (r-l+1)*v;
        if(l!=r){
```

```
        lazy[left(p)]+=v;
        lazy[right(p)]+=v;
      }
      return;
    }
    int mid = (l+r)/2;
    update(left(p),l,mid,i,j,v);
    update(right(p),mid+1,r,i,j,v);
    st[p] = st[left(p)] + st[right(p)];
  }

  ll query(int p, int l, int r, int i, int
j){
    if(i>r || j<l){return 0;}
    if(lazy[p]!=0){
      st[p]+=(r-l+1)*lazy[p];
      if(l!=r){
        lazy[left(p)]+=lazy[p];
        lazy[right(p)]+=lazy[p];
      }
      lazy[p] = 0;
    }
    if(l>=i && r<=j){return st[p];}
    int mid = (l+r)/2;
    ll a = query(left(p),l,mid,i,j);
    ll b = query(right(p),mid+1,r,i,j);
    return (a+b);
  }
};
```

UFDS

```
class UnionFind{
private:
  vi rank,p,setSize;
  int numset,i;
public:
  UnionFind(int n){
     numset=n; setSize.assign(n,1);
     rank.assign(n,0); p.assign(n,0);
     for(i=0;i<n;i++){p[i]=i;}
  }

  int findSet(int i){return (p[i]==i) ? i :
(p[i]=findSet(p[i]));}

  bool isSameSet(int i, int j){return
findSet(i)==findSet(j);}

  void unionSet(int i, int j){
    if(!isSameSet(i,j)){
      numset--;
      int x=findSet(i), y=findSet(j);
      if(rank[x] > rank[y]){p[y]=x;
setSize[x]+=setSize[y];}
      else{
        p[x]=y;
        setSize[y]+=setSize[x];
        if(rank[x]==rank[y]){rank[y]++;}
      }
    }
  }

  int numDisjointSet(){return numset;}

  int sizeSetOf(int i){return
```

```
setSize[findSet(i)];}
};
```

Bipartite Graph Check

```
bool isBipartiteCheck (){
  int n,e,i,j,a,b,v,vertex;
  queue <int> q;
  bool isBipartite;
  while(scanf("%d",&n),n){
    AdjList.assign(n,vi());
    vi color(n,inf);
    scanf("%d",&e);
    for(i=0;i<e;i++){
      scanf("%d %d",&a,&b);
      AdjList[a].push_back(b);
      AdjList[b].push_back(a);
    }
    q.push(0);color[0]=0;
    isBipartite=true;
    while(!q.empty() && isBipartite){
      v = q.front(); q.pop();
      for(i=0;i<(int)AdjList[v].size();i++){
        vertex=AdjList[v][i];
        if(color[vertex]==inf){
          color[vertex]=1-color[v];
          q.push(vertex);
        }else if(color[vertex]==color[v]){
          isBipartite=false;
        }
      }
    }
```

```
  return isBipartite;
}
```

Finding a Cycle in a Graph

```
void graphCheck(int u){
  if(foundCycle){return;}
  dfs_num[u] = explored;
  for(int i=0;i<AdjList[u].size();i++){
    int v = AdjList[u][i];
    if(dfs_num[v] == unvisited){
      graphCheck(v);
    }
    if(dfs_num[v] == explored){
      foundCycle = true;
    }
  }
  dfs_num[u] = visited;
}
```

Finding Articulation Points and Bridges

```
void articulationPointAndBridge(int u) {
  dfs_low[u]=dfs_num[u]=dfsNumberCounter++;
  for (int j = 0; j < AdjList[u].size(); j+
+) {
    int v = AdjList[u][j];
    if (dfs_num[v] == DFS_WHITE) { // a tree
edge
      dfs_parent[v] = u;
      if (u == dfsRoot) rootChildren++;  //
```

```
special case, count children of root

      articulationPointAndBridge(v);

      if (dfs_low[v] >= dfs_num[u])
// for articulation point
        articulation_vertex[u] = true;
// store this information first
      if (dfs_low[v] > dfs_num[u])
// for bridge
        printf(" Edge (%d, %d) is a
bridge\n", u, v);
      dfs_low[u] = min(dfs_low[u],
dfs_low[v]; // update dfs_low[u]
    }
    else if (v != dfs_parent[u]) // a back
edge and not direct cycle
      dfs_low[u] = min(dfs_low[u],
dfs_num[v]); // update dfs_low[u]
} }
```

Finding SCC

```
/****** Tarjan's SCC *******/
vector< int > num, low, S, vis;
int cntr, numCC;

void tarjanSCC(int v) {
  low[v] = num[v] = ++cntr;
  vis[v] = 1;
  S.push_back(v);
  for(auto u : adj[v]) {
    if(num[u] == -1)
      tarjanSCC(u);
    if(vis[u])
      low[v] = min(low[v], low[u]);
  }
  if(low[v] == num[v]) {
    printf("SCC %d :", ++numCC);
    while(1) {
      int u = S.back(); S.pop_back(); vis[u]
= 0;
      printf(" %d", u);
      if(u == v)
        break;
    }
  }
}

// In MAIN();
  num.assign(n, -1);
  low.assign(n, 0);
  vis.assign(n, 0);
  cntr = numCC = 0;
  for(int i = 0; i<n; i++))
    if(num[i] == -1)
      tarjanSCC(i);
```

Dijkstra

```
vector <vii> AdjList;
vi dist;
```

```
int main(){
  int V,E,s,u,v,w,i,j;


  scanf("%d %d %d",&V,&E,&s);
  AdjList.assign(V,vii());
  dist.assign(V,inf);
  for(i=0;i<E;i++){
    scanf("%d %d %d",&u,&v,&w);
    AdjList[u].push_back(ii(v,w));
  }

  dist[s]=0;
  priority_queue <ii,vii,greater<ii> > pq;
pq.push(ii(dist[s],s));//coba dimodif
  while(!pq.empty()){
    ii front=pq.top(); pq.pop();
    int d=front.first,v1=front.second;
    if(d > dist[v1]){continue;}//biar nanti
otomatis ke pop sendiri -> lazy deletion
    for(i=0;i<AdjList[v1].size();i++){
      ii pair=AdjList[v1][i];
      if(dist[pair.first] >
pair.second+dist[v1]){
        dist[pair.first]=pair.second+dist[v1];

  pq.push(ii(dist[pair.first],pair.first));
      }
    }
  } }
```

Bellman Ford's Algo (negative cycle check)

```
dist.assign(n+1,inf);
AdjList.assign(n+1,vii());
  int w;
  for(i=0;i<m;i++){scanf("%d %d
%d",&a,&b,&w); AdjList[a].pb(ii(b,w));}
  dist[u] = 0;
  for(i=0;i<n-1;i++){
    for(int k = 1;k<=n;k++){
      for(j=0;j<AdjList[k].size();j++){
        ii v = AdjList[k][j];
        dist[v.first] =
min(dist[v.first],dist[k] + v.second);
      }
    }
  }
  bool hasCycle = false;
  for(int k = 1;k<=n;k++){
    for(j=0;j<AdjList[k].size();j++){
      ii v = AdjList[k][j];
      if(dist[v.first] > dist[k]+v.second)
{hasCycle = true; break;}
    }
    if(hasCycle){break;}
  }
  if(hasCycle){printf("TIDAK\n");}
  else{printf("BISA\n");}
```

Floyd Warshall's Algo (print path too)

```
for(int k=1;k<=n;k++){
   for(i=1;i<=n;i++){
     for(j=1;j<=n;j++){
       if(mat[i][j] > mat[i][k] + mat[k][j]){
```

```
        mat[i][j] = mat[i][k] + mat[k][j];
        p[i][j] = p[k][j];
      }
    }
  }
 }

//print path dr a ke b….
CatatPath(a,b);
//rekursif
void CatatPath(int i, int j){
 if(i!=j){CatatPath(i,p[i][j]);}
 //printf("yang dipush: %d\n",j);
 ans.pb(j);
}
```

Maxflow Edmond Karp

```
#define maxn 102

int s,t,f,mf;
vi p;
int res[maxn][maxn];
vector<vi> AdjList;

void augment(int v, int minEdge){
 if(v==s){f = minEdge; return;}
 else if(p[v]!=-1){
  augment(p[v],min(minEdge,res[p[v]][v]));
  res[p[v]][v]-=f; res[v][p[v]]+=f;
 }
}
```

```
int main() {
 int n,i,j,a,b,w,test=1;
 while(scanf("%d",&n),n){
  AdjList.assign(n+1,vi());
  printf("Network %d\n",test++);
  memset(res,0,sizeof res);
  int m;
  scanf("%d %d %d",&s,&t,&m); s--; t--;
  for(i=0;i<m;i++){
   scanf("%d %d %d",&a,&b,&w); a--; b--;
   res[a][b] +=w; res[b][a] += w;
   AdjList[a].pb(b); AdjList[b].pb(a);
  }
  mf = 0;
    while (1) {                    // now
a true O(VE^2) Edmonds Karp's algorithm
      f = 0;
      bitset<maxn> vis; vis[s] = true;
// we change vi dist to bitset!
      queue<int> q; q.push(s);
      p.assign(maxn, -1);
      while (!q.empty()) {
        int u = q.front(); q.pop();
        if (u == t) break;
        for (int j = 0; j <
(int)AdjList[u].size(); j++) {  // we use
AdjList here!
          int v = AdjList[u][j];
          if (res[u][v] > 0 && !vis[v])
            vis[v] = true, q.push(v), p[v]
= u;
        }
```

```
        }
        augment(t, inf);
        if (f == 0) break;
        mf += f;
      }
    printf("The bandwidth is %d.\n\n",mf);
    AdjList.clear(); p.clear();
  }

  return 0;
}
```

Eulerian Graph Check

Jumlah node derajat ganjil = 0 -> eulerian tour
Klo jumlah ganjil ada 2->semua dikunjungin tp ga euler tour
Selain itu ga eulerian graph.

Printing Euler Tour

```
list<int> cyc;
  void EulerTour(list<int>::iterator i, int u){
    for(int j=0;j<AdjList[u].size();j++){
      ii v = AdjList[u][j];
      if(v.second){
        v.second = 0;
        for(int k=0;k<AdjList[v.first].size();k++){
          ii uu = AdjList[v.first][k];
          if(uu.first == u && uu.second){
            uu.second = 0;
            break;
          }
        }
      }
      EulerTour(cyc.insert(i,u),v.first);
    }
  }

  cyc.clear();
  EulerTour(cyc.begin(),A);
  for(list<int>::iterator it = cyc.begin();it!=cyc.begin();it++){
    printf("%d\n",*it);
  }
```

Binomial Coefficient

```
C(n,0) = C(n,n) = 1;
C(n,k) = C(n-1,k-1) + c(n-1,k) //n>k>0
```

Catalan Numbers

```
Cat(0) = 1;
Cat(m) = (2m*(2m-1)/((m+1)*m))*cat(m-1);
```

Factorization

```
vi primefactor(ll n){
  vi factors;
  ll idx = 0, pf = prime[idx];
  while(pf*pf<=n){
```

```
   while(n%pf==0){n/=pf;
factors.push_back(pf); }
   pf = prime[++idx];
  }
  if(n!=1){factors.push_back(n);}
  return factors; }
```

Sum of Divisors of N

```
Ll sumDiv(ll n){
  ll idx = 0, pf = prime[idx]; ans=1;
  while(pf*pf<=n){
          ll power = 0;
    while(n%pf==0){n/=pf; power++;}
    pf = prime[++idx];
     ans*=((ll)pow((double)pf,power+1.0)-1)/
(pf-1);
     pf = prime[++idx];
  }
  if(n!=1){ans*=((ll)pow((double)n,2.0)-1)/
(n-1);}
      return ans;
}
```

Euler Phi

```
ll eulerPhi(ll n){
  ll idx = 0, pf = prime[idx], ans = n;
  while(pf*pf<=n){
    if(n%pf==0){ans-=ans/pf;}
    while(n%pf==0){n/=pf;}
    pf = prime[++idx];
  }
  if(n!=1){ans-=ans/n;}
  return ans;
}
```

Extended Euclid Algorithm

```
long long x, y, d; // ax + by = d
void extendedEuclidean(long long a, long
long b) {
  if(b == 0) { x = 1; y = 0; d = a;
return; }
  extendedEuclidean(b, a % b);
  long long xx, yy;
  xx = y;
  yy = x - (a/b)*y;
  x = xx; y = yy;
}
```

Cycle Finding

```
ll z,i,m;

ll f(ll x){
  return ((z*x)+i)%m;
}

ii floydCycleFinding(ll x0){
  //cari k*mu
  ll tortoise = f(x0), hare = f(f(x0));
  while(tortoise!=hare){tortoise =
f(tortoise); hare = f(f(hare));}
  //cari mu
  int mu = 0; hare = x0;
```

```
  while(tortoise!=hare){tortoise =
f(tortoise); hare = f(f(hare)); mu++;}
  //finding lambda
  int lambda = 1;hare = f(tortoise);
  while(tortoise!=hare){hare=f(hare);
lambda++;}
  return ii(mu,lambda); }
```

KMP

```
void kmpPreprocess() {
  int i = 0, j = -1; b[0] = -1;
  while(i<m) {
    while(j >= 0 && pattern[i]!=pattern[j]) j
= b[j];
    i++; j++;
    b[i] = j;
  }
}

void kmpSearch() {
  int i = 0, j = 0;
  while(i<n) {
    while(j >= 0 && text[i]!=pattern[j]) j =
b[j];
    i++; j++;
    if(j == m) {
      printf("pattern found in index %d\n",i-
j);
      j = b[j];
    }
  }
}
```

Edit Distance

```
int solve(string kata1,string kata2){
  int panjang1 = kata1.length(),panjang2 =
kata2.length();
  int i,j;
  //buat base case
  for(i=0;i<=panjang1;i++){a[i][0] = i;}
  for(j=0;j<=panjang2;j++){a[0][j] = j;}
  for(i=1;i<=panjang1;i++){
    for(j=1;j<=panjang2;j++){
      if(kata1[i-1] == kata2[j-1]){a[i][j] =
a[i-1][j-1];}
      else{
        a[i][j] = min(a[i-1][j],min(a[i][j-
1],a[i-1][j-1])) + 1;
      }
    }
  }
  return a[panjang1][panjang2];
}
```

Longest Palindrome

```
int solve(int l, int r){
  //if(l>r){return 0;}
  if(l==r){return 1;}
  if(l+1==r){
    if(kata[l]==kata[r]){return 2;}
    else{return 1;}
  }
  if(memo[l][r]!=-1){return memo[l][r];}
  if(kata[l]==kata[r]){return memo[l][r] = 2
```

```
+ solve(l+1,r-1);}
  return memo[l][r] = max(solve(l,r-
1),solve(l+1,r));
}
```

LCS

```
  for i=0 to n {
   a[i][0] = 0;//base case
  }
  for j=0 to m {
   a[0][j] = 0;//base case
  }
  for i=1 to n {
   for j=1 to m {
    if(word1[i-1] = word2[j-1]) then
      a[i][j]  =  a[i-1][j-1]  +  1;  //same
characters
    } else {
      a[i][j] = max (a[i-1][j], a[i][j-1]);
      //different characters
    }
   }
  }
  print a[n][m]; //printing the answer
```

Template Geometri

Points and Lines

```
double degToRad(double a){return
a*pi/180.0;}
double radToDeg(double a){return
a*180.0/pi;}

struct point_i{
  int x,y;
  point_i(){x = 0; y=0;}
  point_i(int _x, int _y){x = _x; y = _y;}
};

struct point{
  double x,y;
  point(){x = y = 0.0;}
  point(double _x, double _y): x(_x), y(_y)
{}
  bool operator < (point other) const {
    if(fabs(x-other.x)>eps){return x <
other.x;}
    return y<other.y;
  }
  bool operator == (point other){return
((fabs(x-other.x)<eps) && (fabs(y-
other.y)<eps));}
};
double dist(point p1, point p2){return
hypot(p1.x-p2.x,p1.y-p2.y);}
point rotate(point p, double theta){
  double rad = degToRad(theta);
  return point(p.x*cos(rad) - p.y*sin(rad),
p.x*sin(rad) + p.y*cos(rad));
}

struct line{double a,b,c;};
```

```
void pointsToLine(point p1, point p2, line
&l){
  if(fabs(p1.x-p2.x) < eps){
    l.a = 1.0; l.b = 0.0; l.c = -p1.x;
  }else{
    l.a = -(double)(p1.y-p2.y) / (p1.x-p2.x);
    l.b = 1.0;
    l.c = -(double)(l.a*p1.x) - p1.y;
  }
}

bool areParallel(line l1, line l2){return
((fabs(l1.a-l2.a)<eps) && (fabs(l1.b-
l2.b)<eps));}
bool areSame(line l1, line l2){
  if(areParallel(l1,l2)){return fabs(l1.c-
l2.c)<eps;}
  return false;
}

bool areIntersect(line l1, line l2, point
&p){
  if(areParallel(l1,l2)){return false;}
  p.x = (l2.b*l1.c - l1.b*l2.c) / (l2.a*l1.b
- l1.a*l2.b);
  if(fabs(l1.b) > eps){p.y = -(l1.a*p.x +
l1.c);}
  else{p.y = -(l2.a*p.x + l2.c);}
  return true;
}

struct vec{
  double x,y;
  vec(double _x, double _y): x(_x), y(_y){}
};

vec toVec(point a, point b){
  return vec(b.x-a.x, b.y-a.y);
}
vec scale(vec v, double s){
  return vec(v.x*s,v.y*s);
}
point translate(point p, vec v){//translate
p sebanyak v
  return point(p.x+v.x, p.y+v.y);
}
double dot(vec a, vec b){return (a.x*b.x +
a.y*b.y);}
double norm_sq(vec v){return (v.x*v.x +
v.y*v.y);}

double distToLine(point p, point a, point b,
point &c){
  vec ap = toVec(a,p), ab = toVec(a,b);
  double u = dot(ap,ab) / norm_sq(ab);
  c = translate(a,scale(ab,u));
  return dist(p,c);
}
double distToLineSegment(point p, point a,
point b, point &c){
  vec ap = toVec(a,p), ab = toVec(a,b);
  double u = dot(ap,ab) / norm_sq(ab);
  if(u<0.0){c = point(a.x,a.y); //closer to
a
    return dist(p,a);
```

```
  }
  if(u>1.0){
    c = point(b.x,b.y);
    return dist(p,b);
  }
  return distToLine(p,a,b,c);
}

double angle(point a, point o, point b)
{//return in rad
  vec oa = toVec(o,a), ob = toVec(o,b);
  return acos(dot(oa,ob) / sqrt(norm_sq(oa)
* norm_sq(ob)));
}

double cross(vec a, vec b){return a.x*b.y -
a.y*b.x;}
bool ccw(point p, point q, point r){
  return cross(toVec(p,q), toVec(p,r)) > 0;
}
bool collinear(point p, point q, point r){
  return fabs(cross(toVec(p,q), toVec(p,r)))
< eps;
}
```

Circle

```
int insideCirlce(point p, point center,
double r){
  double dx = p.x - center.x, dy = p.y-
center.y;
  double Euc = dx*dx + dy*dy, rSq = r*r;
  if(fabs(Euc-rSq)<eps){return 1;}//in
border
  if(Euc<rSq){return 0;}//inside
  if(Euc>rSq){return 2;}//outside
}

bool circle2PtsRad(point p1, point p2,
double r, point& c){
  double d2 = (p1.x-p2.x)*(p1.x-p2.x) +
(p1.y-p2.y)*(p1.y-p2.y);
  double det = r*r/d2 - 0.25;
  if(det<0.0){return false;}
  double h = sqrt(det);
  c.x = (p1.x+p2.x)*0.5 + (p1.y-p2.y)*h;
  c.y = (p1.y+p2.y)*0.5 + (p2.x-p1.x)*h;
  return true;
}
```

Triangle

```
/*Sudah ditambahkan library point*/
/*TAMBAHAN LIBRARY DARI TRIANGLE*/
double perimeter(double ab, double bc,
double ac){
  return ab+bc+ac;
}
double perimeter(point a, point b, point c){
  return
perimeter(dist(a,b),dist(b,c),dist(a,c));
}

double area(double ab, double bc, double ac)
{
```

```
  double s = perimeter(ab,bc,ac)*0.5;
  return sqrt(s)*sqrt(s-ab)*sqrt(s-
bc)*sqrt(s-ac);
}

double area(point a, point b, point c){
  return area(dist(a,b), dist(b,c),
dist(a,c));
}

double rInCircle(double ab, double bc,
double ac){//panjang jari"lingkaran dalam
  return(area(ab,bc,ac)/
(0.5*perimeter(ab,bc,ac)));
}
double rInCircle(point a, point b, point c){
  return
rInCircle(dist(a,b),dist(b,c),dist(a,c));
}
//cari titik tengah inscribed Circle dan
radiusnya
int inCircle(point p1, point p2, point p3,
point& ctr, double& r){//return 0..ga ada
lingkaran dalam segitiga, otherwise return 1
  r = rInCircle(p1,p2,p3);
  if(fabs(r)<eps){return 0;}//3 point
collinear
  line l1,l2;
  double ratio = dist(p1,p2)/dist(p1,p3);
  point p =
translate(p2,scale(toVec(p2,p3),ratio/
(1+ratio)));
  pointsToLine(p1,p,l1);
  ratio = dist(p2,p1)/dist(p2,p3);
  p = translate(p1,scale(toVec(p1,p3),ratio/
(1+ratio)));
  pointsToLine(p2,p,l2);

  areIntersect(l1,l2,ctr);
  return 1;
}
double rCircumCircle(double ab, double bc,
double ac){
  return ab*bc*ac / (4.0*area(ab,bc,ac));
}
double rCircumCircle(point a, point b, point
c){
  return rCircumCircle(dist(a,b), dist(b,c),
dist(a,c));
}

//cari titik tengah circumCircle dan
radiusnya
int circumCircle(point p1, point p2, point
p3, point &ctr, double &r){
  double a = p2.x - p1.x, b = p2.y - p1.y;
  double c = p3.x - p1.x, d = p3.y - p1.y;
  double e = a * (p1.x + p2.x) + b * (p1.y +
p2.y);
  double f = c * (p1.x + p3.x) + d * (p1.y +
p3.y);
  double g = 2.0 * (a * (p3.y - p2.y) - b *
(p3.x - p2.x));
  if (fabs(g) < eps) return 0;
```

```
  ctr.x = (d*e - b*f) / g;
  ctr.y = (a*f - c*e) / g;
  r = dist(p1, ctr);  // r = distance from
center to 1 of the 3 points
  return 1; }

// returns true if point d is inside the
circumCircle defined by a,b,c
int inCircumCircle(point a, point b, point
c, point d) {
  return (a.x - d.x) * (b.y - d.y) * ((c.x -
d.x) * (c.x - d.x) + (c.y - d.y) * (c.y -
d.y)) +
         (a.y - d.y) * ((b.x - d.x) * (b.x -
d.x) + (b.y - d.y) * (b.y - d.y)) * (c.x -
d.x) +
         ((a.x - d.x) * (a.x - d.x) + (a.y -
d.y) * (a.y - d.y)) * (b.x - d.x) * (c.y -
d.y) -
         ((a.x - d.x) * (a.x - d.x) + (a.y -
d.y) * (a.y - d.y)) * (b.y - d.y) * (c.x -
d.x) -
         (a.y - d.y) * (b.x - d.x) * ((c.x -
d.x) * (c.x - d.x) + (c.y - d.y) * (c.y -
d.y)) -
         (a.x - d.x) * ((b.x - d.x) * (b.x -
d.x) + (b.y - d.y) * (b.y - d.y)) * (c.y -
d.y) > 0 ? 1 : 0;
}

bool canFormTriangle(double a, double b,
double c){
  return (a+b>c) && (a+c>b) && (b+c>a);
}
```

Polygon

```
//sudah ditambahkan library point
double perimeter(const vector<point> &P){
  double result = 0.0;
  for(int i=0;i<P.size()-1;i++){
    result+=dist(P[i],P[i+1]);
  }
  return result;
}

double area(const vector<point> &P){
  double result = 0.0, x1,x2,y1,y2;
  for(int i=0;i<P.size()-1;i++){
    x1 = P[i].x; x2 = P[i+1].x;
    y1 = P[i].y; y2 = P[i+1].y;
    result+=(x1*y2 - x2*y1);
  }
  return fabs(result)/2.0;
}

bool isConvex(const vector<point> &P){
  int sz = P.size();
  if(sz <= 3){return false;}
  bool isLeft = ccw(P[0],P[1],P[2]);
  for(int i=1;i<sz-1;i++){
    if(ccw(P[i],P[i+1],P[(i+2)==sz? 1:i+2]) !
= isLeft){return false;}
  }
  return true;
}
```

```
bool inPolygon(point pt, const vector<point>
&P){
  if(P.size()==0){return false;}
  double sum = 0.0;
  for(int i=0;i<P.size()-1;i++){
    if(ccw(pt,P[i],P[i+1]))
{sum+=angle(P[i],pt,P[i+1]);}
    else{sum-=angle(P[i],pt,P[i+1]);}
  }
  return fabs(fabs(sum)-2*pi) <eps;
}

point pivot;
bool angleCmp(point a, point b){
  if(collinear(pivot, a, b)){return
dist(pivot,a) < dist(pivot,b);}
  point d1,d2;
  d1.x = a.x - pivot.x, d1.y = a.y -
pivot.y;
  d2.x = b.x - pivot.x, d2.y = b.y -
pivot.y;
  return (atan2(d1.y,d1.x) -
atan2(d2.y,d2.x)) < 0;
}

vector<point> CH(vector<point> P) {
  int i,j, n =P.size();
  if(n<=3){
    if(!(P[0]==P[n-1])){P.pb(P[0]);}
    return P;
  }
  //find index so that P[idx] has lowest Y,
if tie..the rightmost X
  int idx = 0;
  for(i=1;i<n;i++){
    if(P[i].y < P[idx].y || (P[i].y==P[idx].y
&& P[i].x > P[idx].x)){
      idx = i;
    }
  }
  //swap routine
  point temp = P[0]; P[0] = P[idx]; P[idx] =
temp;
  //sort
  pivotf = P[0];
  sort(++P.begin(),P.end(),angleCmp);

  vector<point> S;
  S.pb(P[n-1]); S.pb(P[0]); S.pb(P[1]);
  i = 2;
  while(i<n){
    j = S.size()-1;
    if(ccw(S[j-1],S[j],P[i])){S.pb(P[i++]);}
    else{S.pop_back();}
  }
  return S;
}

// line segment p-q intersect with line A-B.
point lineIntersectSeg(point p, point q,
point A, point B) {
  double a = B.y - A.y;
  double b = A.x - B.x;
  double c = B.x * A.y - A.x * B.y;
  double u = fabs(a * p.x + b * p.y + c);
```

```
  double v = fabs(a * q.x + b * q.y + c);
  return point((p.x * v + q.x * u) / (u+v),
(p.y * v + q.y * u) / (u+v)); }

// cuts polygon Q along the line formed by
point a -> point b
// (note: the last point must be the same as
the first point)
vector<point> cutPolygon(point a, point b,
const vector<point> &Q) {
  vector<point> P;
  for (int i = 0; i < (int)Q.size(); i++) {
    double left1 = cross(toVec(a, b),
toVec(a, Q[i])), left2 = 0;
    if (i != (int)Q.size()-1) left2 =
cross(toVec(a, b), toVec(a, Q[i+1]));
    if (left1 > -eps) P.push_back(Q[i]);
// Q[i] is on the left of ab
    if (left1 * left2 < -eps)        // edge
(Q[i], Q[i+1]) crosses line ab
      P.push_back(lineIntersectSeg(Q[i],
Q[i+1], a, b));
  }
  if (!P.empty() && !(P.back() ==
P.front()))
    P.push_back(P.front());        // make
P's first point = P's last point
  return P; }
```

LCA

```
int L[2*maxn], H[2*maxn], E[2*maxn], idx;

void dfs(int cur, int parent, int depth) {
  H[cur] = idx;
  E[idx] = cur;
  L[idx++] = depth;
  for (int i = 0; i < AdjList[cur].size();
i++) {
   int v = AdjList[cur][i];
   if(v!=parent){
     dfs(v, cur, depth+1);
      E[idx] = cur;
// backtrack to current node
      L[idx++] = depth;
   }

  }
}

void buildRMQ() {
  idx = 0;
  memset(H, -1, sizeof H);
  dfs(0, -1, 0);                          // we
assume that the root is at index 0
}

//Code Segtree RMQ

int main() {
  int n,i,j,a,b;
```

```
  scanf("%d",&n);
  AdjList.assign(n,vi());
  for(i=0;i<n-1;i++){
    scanf("%d %d",&a,&b); a--; b--;
    AdjList[a].pb(b); AdjList[b].pb(a);
  }

  buildRMQ();

  vi A;
  for(i=0;i<2*n;i++){A.pb(L[i]);}
  SegmentTree s(A);

  int q;
  scanf("%d",&q);
  while(q--){
    scanf("%d %d",&a,&b);
    a--; b--;
    if(a==b){printf("TIDAK\n"); continue;}
    int nilaiAcuan = a;
    if(H[a] > H[b]){
      swap(a,b);
      nilaiAcuan = b;
    }

    int idx1 = H[a], idx2 = H[b];
    int idx = s.rmq(idx1,idx2);
    int ans = E[idx]; //hasil LCA
    if(ans==nilaiAcuan){printf("TIDAK\n");}
    else{printf("YA\n");}
  }
  return 0;
}
```

Gcd-extended Algorithm

```
int gcd_extended(int x,int y,int *cx,int
*cy) //cx.x + cy.y = gcd(x,y)
{
  if(y==0)
  {
    *cx = 1;
    *cy = 0;
    return x;
  }
  else
  {
    int dx,dy; //dx.y + dy.(x%y) = gcd(y,x%y)
= gcd(x,y)
    int result = gcd_extended(y,x%y,&dx,&dy);

    *cx = dy;
    *cy = dx - dy*(x/y);

    return result;
  }
}

/* Explanation :
cx.x + cy.y = dx.y + dy.(x%y)
          = dx.y + dy (x - (x/y)*y)
          = dx.y + dy.x - dy.y.(x/y)
```

```
          = dy.x + (dx - dy(x/y)).y
cx.x = dy.x -> cx = dy
cy.y = (dx - dy(x/y)).y -> cy = dx - dy(x/y)

Base case (y=0) :
cx.x + cy.0 = gcd(x,0) = x
cx = 1, cy = 0
*/

int main()
{
  int a,b;
  int c1,c2; //c1*a + c2*b = gcd(a,b)

  printf("Enter 2 positive numbers : ");
  scanf("%d %d",&a,&b);

  if(a>b) //make a<=b, swap
  {
    a ^= b;
    b ^= a;
    a ^= b;
  }

  /* Case : a^-1 (mod b) = .. */

  if(gcd_extended(a,b,&c1,&c2)!=1)
printf("Modular multiplicative inverse does
not exists.\n");
  else
  {
    if(c1<0) c1 = b - (abs(c1)%b); //c1 may
be negative, better make it positive
    printf("Modular multiplicative inverse of
%d (mod %d) is = %d.\n",a,b,c1);
  }

  /* c1*a + c2*b = gcd(a,b) may be used to
solve linear diophantine equation */

  return 0;
}
```

1. Rumus-rumus kombin

$$\sum_{k=0}^{n} k\binom{n}{k} = n2^{n-1}$$

$$\sum_{k=0}^{n} k^2\binom{n}{k} = (n+n^2)2^{n-2}$$

$$\sum_{j=0}^{k} \binom{m}{j}\binom{n-m}{k-j} = \binom{n}{k}$$

$$\sum_{m=0}^{n} \binom{m}{j}\binom{n-m}{k-j} = \binom{n+1}{k+1},$$

$$\sum_{m=0}^{m} \binom{m}{k} = \binom{n+1}{k+1}.$$

$$\sum_{j=0} \binom{m}{j}^2 = \binom{2m}{m}.$$

$$\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n-k}{k} = F(n+1).$$

$$\sum_{i=0}^{n} i \binom{n}{i}^2 = \frac{n}{2}\binom{2n}{n}$$

$$\sum_{i=0}^{n} i^2 \binom{n}{i}^2 = n^2 \binom{2n-2}{n-1}.$$

$$\sum_{k=q}^{n} \binom{n}{k}\binom{k}{q} = 2^{n-q}\binom{n}{q}$$

Dixon Identity:

$$\sum_{k=-a}^{a} (-1)^k \binom{2a}{k+a}^3 = \frac{(3a)!}{(a!)^3}$$

$$\sum_{k=-a}^{a} (-1)^k \binom{a+b}{a+k}\binom{b+c}{b+k}\binom{c+a}{c+k} = \frac{(a+b+c)!}{a!\,b!\,c!},$$

where *a*, *b*, and *c* are non-negative integers

$$\binom{n}{k_1, k_2, \ldots, k_r} = \frac{n!}{k_1! k_2! \cdots k_r!}$$

$$\binom{z}{m}\binom{z}{n} = \sum_{k=0}^{m} \binom{m+n-k}{k, m-k, n-k}\binom{z}{m+n-k}$$

Lucas' Theorem :

For non-negative integers *m* and *n* and a prime *p*, the following congruence relation holds:

$$\binom{m}{n} \equiv \prod_{i=0}^{k} \binom{m_i}{n_i} \pmod{p},$$

where

$$m = m_k p^k + m_{k-1} p^{k-1} + \cdots + m_1 p + m_0,$$

and

$$n = n_k p^k + n_{k-1} p^{k-1} + \cdots + n_1 p + n_0$$

are the base *p* expansions of *m* and *n* respectively. This uses the convention that $\binom{m}{n} = 0$ if $m < n$.

Example : (combinatrics in small mod wheren mod < n && mod < k)

```
int comb[mod][mod];
int c(int n, int k) {
  return n == 0? 1 : comb[n%mod][k%mod] * c(n/mod, k/mod) % mod;
}
```

Faulhaber's Formula

$$(n+1)^{k+1} - 1 = \sum_{m=1}^{n} \left((m+1)^{k+1} - m^{k+1}\right) = \sum_{p=0}^{k} \binom{k+1}{p} (1^p + 2^p + \cdots + n^p)$$

Examples:

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = \frac{n^2+n}{2}$$ (the triangular numbers)

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = \frac{2n^3+3n^2+}{6}$$

(the square pyramidal numbers)

$$1^3 + 2^3 + 3^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^4+2n^3+}{4}$$

(the squared triangular numbers)

$$1^4 + 2^4 + 3^4 + \cdots + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-}{30} = \frac{6n^5+15n^4+10n^3-n}{30}$$

$$1^5 + 2^5 + 3^5 + \cdots + n^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{2n^6+6n^5+5n^4-n^2}{12}$$

$$1^6 + 2^6 + 3^6 + \cdots + n^6 = \frac{n(n+1)(2n+1)(3n^4+6n^3-3n+}{42} = \frac{6n^7+21n^6+21n^5-7n^3+n}{42}$$

2. Suffix Array + LCP

```
// suffix array
const int N = 1e5 + 5;
string s;
int sa[N], pos[N], lcp[N], tmp[N], gap, n;

bool cmp_sa(int a, int b) {
  if(pos[a] - pos[b])
    return pos[a] < pos[b];
  a += gap; b += gap;
  return (a < n && b < n) ? pos[a] < pos[b] : a > b;
}

void build_sa() {
  n = s.size();
  for(int i = 0; i<n; i++)
    sa[i] = i, pos[i] = s[i];
  for(gap = 1;; gap <<= 1) {
    sort(sa, sa + n, cmp_sa);
    for(int i = 1; i<n; i++) tmp[i] = tmp[i-1] + cmp_sa(sa[i-1],
sa[i]);
    for(int i = 0; i<n; i++) pos[sa[i]] = tmp[i];
    if(tmp[n-1] == n-1) break;
  }
}

void build_lcp() {
  for(int i = 0, k = 0; i<n; i++) if(pos[i] - n + 1) {
    for(int j = sa[pos[i] + 1]; s[j + k] == s[i + k]; k++);
    lcp[pos[i]] = k;
    if(k) k--;
  }
}
```

3. FFT biasa & FFT versi modular arithmetic (perkalian polinom)

```
/***************** FFT dengan complex ******************/
typedef complex<double> cd;
typedef vector< cd > vcd;
```

```
// asumsi ukuran as = 2^k, dengan k bilangan bulat positif
vcd fft(const vcd &as) {
  int n = (int)as.size();
  int k = 0;
  while((1<<k) < n) k++;
  vector< int > r(n);
  r[0] = 0;
  int h = -1;
  for(int i = 1; i<n; i++) {
    if((i & (i-1)) == 0)
      h++;
    r[i] = r[i ^ (1 << h)];
    r[i] |= (1<<(k-h-1));
  }
  vcd root(n);
  for(int i = 0; i<n; i++) {
    double ang = 2.0*M_PI*i/n;
    root[i] = cd(cos(ang), sin(ang));
  }

  vcd cur(n);
  for(int i = 0; i<n; i++)
    cur[i] = as[r[i]];

  for(int len = 1; len < n; len <<= 1 ) {
    vcd ncur(n);
    int step = n/(len << 1);
    for(int pdest = 0; pdest <n;) {
      for(int i = 0; i<len; i++) {
        cd val = root[i*step]*cur[pdest + len];
        ncur[pdest] = cur[pdest] + val;
        ncur[pdest + len] = cur[pdest] - val;
        pdest++;
      }
      pdest += len;
    }
    cur.swap(ncur);
  }
  return cur;
}

vcd inv_fft(const vcd& fa) {
  vcd res = fft(fa);
  for(int i = 0; i<nn; i++) {
    res[i] /= nn;
  }
  reverse(res.begin() + 1, res.end());
  return res;
}
```

```
/*************** FFT dengan Modular Aritmetic ***************/
const int mod = 7340033;
const int root = 5;
const int root_1 = 4404020;
const int root_pw = 1<<20;

void fft (vector<int> & a, bool invert) {
  int n = (int) a.size();

  for (int i=1, j=0; i<n; ++i) {
    int bit = n >> 1;
    for (; j>=bit; bit>>=1)
      j -= bit;
    j += bit;
    if (i < j)
      swap (a[i], a[j]);
  }

  for (int len=2; len<=n; len<<=1) {
    int wlen = invert ? root_1 : root;
    for (int i=len; i<root_pw; i<<=1)
      wlen = int (wlen * 1ll * wlen % mod);
    for (int i=0; i<n; i+=len) {
      int w = 1;
      for (int j=0; j<len/2; ++j) {
        int u = a[i+j],  v = int (a[i+j+len/2] * 1ll * w % mod);
        a[i+j] = u+v < mod ? u+v : u+v-mod;
        a[i+j+len/2] = u-v >= 0 ? u-v : u-v+mod;
        w = int (w * 1ll * wlen % mod);
      }
    }
  }
  if (invert) {
    int nrev = reverse (n, mod);
    for (int i=0; i<n; ++i)
      a[i] = int (a[i] * 1ll * nrev % mod);
  }
```

Trie

```
const int ALPHABET_SIZE = 26;
struct TrieNode {
  struct TrieNode *children[ALPHABET_SIZE];
  int maks;
};

struct TrieNode *getNode() {
  struct TrieNode *pNode = new TrieNode;
  for(int i=0;i<ALPHABET_SIZE;i++){
    pNode->children[i] = NULL;
    pNode->maks = -1;
  }
  return pNode;
}

void insert(struct TrieNode *root, string key, int
nilai) {
  struct TrieNode *pCrawl = root;
  for(int i=0;i<key.length();i++){
    int idx = key[i] - 'a';
    if(!pCrawl->children[idx]){
      pCrawl->children[idx] = getNode();
    }
    pCrawl = pCrawl->children[idx];
    pCrawl->maks = max(pCrawl->maks, nilai);
  }
}

int getMax(struct TrieNode *root, string kata) {
  struct TrieNode *pCrawl = root;
  for(int i=0;i<kata.length();i++){
    int idx = kata[i] - 'a';
    if(!pCrawl->children[idx]){
      return -1;
    }
    pCrawl = pCrawl->children[idx];
  }
  return pCrawl->maks;
}

int main(){
  int n,i,j,q;
  scanf("%d %d",&n,&q);
  struct TrieNode *root = getNode();

  for(i=0;i<n;i++){
    string kata;
    int nilai;
    cin>>kata;
    scanf("%d",&nilai);
    insert(root, kata, nilai);
  }
  while(q--){
    string kata;
    cin>>kata;
    int ans = getMax(root, kata);
    printf("%d\n",ans);
  }
  return 0;
};
```

4. Convex hull (Graham's Scan & Andrew's Monotone Chain)

```
typedef pair<long long,long long> point;
#define x first
#define y second
// (p-q) x (r-q)
long long cross(point p, point q, point r) {
  return (p.x - q.x) * (r.y - q.y) - (p.y - q.y) * (r.x - q.x);
}

bool collinear(point a, point o, point b) {
    return cross(a, o, b) == 0;
}

// true if point r is on the left side of line pq
bool ccw(point p, point q, point r) {
    return cross(p, q, r) > 0;
}

point pivot;

long long dist2(point a, point b) {
    return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
}

bool angle_cmp(point a, point b) {
    if(collinear(pivot, a, b)) {
      return dist2(a, pivot) < dist2(b, pivot);
    }
    return ccw(pivot, a, b);
}

bool cmp(point a, point b) {
    return a.y < b.y || (a.y == b.y && a.x < b.x);
}

// P tidak siklik, P[0] tidak mengulang di P.back()
// return convex hull siklik, P[0] mengulang di P.back()
vector<point> ConvexHull(vector<point> P) {
  int i, j, n = (int) P.size();
  if(n < 3)
    return P;
  int PO = 0;
  for(i = 1; i < n; i++) {
  if(cmp(P[P0], P[P[i]])) {
     PO = i;
  }
  }
  swap(P[0], P[PO]);
  pivot = P[0];
  if(collinear(P.back(), P[0], P[1])) {
    vector< point > S;
    S.push_back(P[0]);
    S.push_back(P.back());
    return S;
  }
  sort(++P.begin(), P.end(), angle_cmp);
  int k = P.size() - 1;
  while(k && collinear(P[0], P[k-1], P[k])) k--;
  reverse(P.begin() + k, P.end());
  vector<point> S;
  S.push_back(P[n-1]);
  S.push_back(P[0]);
  S.push_back(P[1]);
  i = 2;
  while(i < n) {
   j = (int) S.size() - 1;
   if(ccw(S[j-1], S[j], P[i])) S.push_back(P[i++]);
   else S.pop_back();
  }
  S.pop_back();
  return S;
}

int main(void)
{
  int n;
  scanf("%d", &n);
  vector<point> p;
  for(int i = 0; i < n; i++) {
   int a, b;
   scanf("%d %d", &a, &b);
   p.push_back(point(a, b));
  }
  vector<point> ch = ConvexHull(p);
  cout << ch.size() << endl;
  for(auto it : ch) {
   printf("%I64d %I64d\n", it.x, it.y);
  }
    return 0;
```