Contents

Data Structures

BIT Binary Indexed Tree

```
struct BIT{//1-indexed
  int n;vector<int> t;
  BIT(){}
  BIT(int _n){n=_n;t.assign(n+5,0);}
  int qry(int i){
    int ans=0;
    for(;i>=1;i-=(i&-i))ans+=t[i];
    return ans;
  }
  void upd(int i,int val){
    if(i<=0)return;
    for(;i<=n;i+=(i&-i))t[i]+=val;
  }
  void upd(int l,int r,int val){
    upd(l,val);
    upd(r+1,-val);
  }
  int qry(int l,int r){
    return qry(r)-qry(l-1);
  }
};
```

DSU Union Find

```
struct DSU{
  vector<int> p,siz;
  DSU(int n){
    p.assign(n+1,0);
    siz.assign(n+1,1);
    iota(all(p),0);
  }
  int get(int x){
    if(p[x]==x)return x;
    return p[x]=get(p[x]);
  }
  bool Merge(int a,int b){
    a=get(a),b=get(b);
    if(a==b)return true;
    if(siz[a]<siz[b])swap(a,b);
    siz[a]+=siz[b];
    p[b]=a;
    return false;
  }
};
```

LIS Longest Increasing Subsequence

```
//Finds LIS of a vector in nlogn also returns the index of elm
vector<int> LIS(const vector<int>&elm){
  auto compare=[&](int x,int y){
    return elm[x]<elm[y];
  };
  set<int,decltype(compare)>S(compare);
  vector<int>prv(elm.size(),-1);
  for(int i=0;i<elm.size();++i){
    auto it=S.insert(i).first;
    if(it!=S.begin())
      prv[i]=*prev(it);
    if(*it==i&&next(it)!=S.end())
      S.erase(next(it));
  }
  vector<int>ans;
  ans.push_back(*S.rbegin());
  while(prv[ans.back()]!=-1)
    ans.push_back(prv[ans.back()]);
  reverse(ans.begin(),ans.end());
  return ans;
}
```

Ordered Set

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

#define ordered_set tree<int, null_type,less<int>, rb_tree_tag,tree_order_statistics_node_update>
int main()
{
 // Ordered set declared with name o_set
  ordered_set o_set;

  // insert function to insert in
  // ordered set same as SET STL
  o_set.insert(5);
  o_set.insert(1);
  o_set.insert(2);

  // Finding the second smallest element
  // in the set using * because
  //  find_by_order returns an iterator


  // Finding the number of elements
  // strictly less than k=4

}
```

Segment Tree

```
struct node{
  int mn,s,mx,lz;
  node(int x=0){
    mx=-M,mn=M,s=x,lz=0;
  }
};

struct ST{
  vector<node> t;
  int n;
  ST(int _n){
    n=_n;
    t.assign(4*n+10,node());
  }
  node Merge(node a,node b){
    node res;
    res.s=min(a.s,b.s);
    return res;
  }
  void upd(int L,int x,int i,int l,int r){
    if(l==r){
      t[i].s+=x;
      return;
    }
    int m=(l+r)/2;
    if(L<=m)upd(L,x,2*i,l,m);
    else upd(L,x,2*i+1,m+1,r);
    t[i]=Merge(t[2*i],t[2*i+1]);
  }
  void upd(int l,int val){
    upd(l,val,1,0,n-1);
  }
  node qry(int L,int R,int i,int l,int r){
    if(l>=L&&r<=R)return t[i];
    if(l>R||r<L)return node();
    int m=(l+r)/2;
    node left,right;
    if(L<=m)left=qry(L,R,2*i,l,m);
    if(m<R)right=qry(L,R,2*i+1,m+1,r);
    return Merge(left,right);
  }
  node qry(int l,int r){
    return qry(l,r,1,0,n-1);
  }
  void prop(int i,int l,int r){
    if(l==r||t[i].lz==0)return;
    t[2*i].s+=t[i].lz;
    t[2*i].lz+=t[i].lz;
    t[2*i+1].s+=t[i].lz;
    t[2*i+1].lz+=t[i].lz;
    t[i].lz=0;
  }
  void upd1(int L,int R,int x,int i,int l,int r){
    prop(i,l,r);
    if(L>r||R<l||l>r)return;
    if(L<=l&&r<=R){
      t[i].lz+=x;
      t[i].s+=x;
      return;
    }
    int m=(l+r)/2;
    upd1(L,R,x,2*i,l,m);
    upd1(L,R,x,2*i+1,m+1,r);
    t[i]=Merge(t[2*i],t[2*i+1]);
  }
  void upd1(int L,int R,int x){
    upd1(L,R,x,1,0,n-1);
  }
  node qry1(int L,int R,int i,int l,int r){
    prop(i,l,r);
    if(L>r||R<l||l>r)return node(1);
    if(L<=l&&r<=R)return t[i];
    int m=(l+r)/2;
    node left,right;
    left=qry1(L,R,2*i,l,m);
    right=qry1(L,R,2*i+1,m+1,r);
    return Merge(left,right);
  }
  node qry1(int L,int R){
    return qry1(L,R,1,0,n-1);
  }
};
```

Sparse Table

```
struct ST{
  int n;
  const int LOG=22;
  vector<int> a,pw;
  vector<vector<int>> mx,mn;

  ST(vector<int> _a){
    a=_a;
    n=(int)a.size();
    pw=vector<int>(n+10);
    mx=mn=vector<vector<int>>(n+1,vector<int>(LOG+1));

    pw[1]=0;
    for(int i=2;i<=n;i++){
      pw[i]=pw[i/2]+1;
    }
    for(int i=0;i<n;i++){
      mx[i][0]=mn[i][0]=a[i];
    }

    for(int k=1;k<LOG;k++){
      for(int i=0;(i+(1<<k)-1)<n;i++){
        mx[i][k]=max(mx[i][k-1],mx[i+(1<<(k-1))][k-1]);
        mn[i][k]=min(mn[i][k-1],mn[i+(1<<(k-1))][k-1]);
      }
    }
  }

  int qmx(int l,int r){
    int k=pw[r-l+1];
    return max(mx[l][k],mx[r-(1<<k)+1][k]);
  }
  int qmn(int l,int r){
    int k=pw[r-l+1];
    return min(mn[l][k],mn[r-(1<<k)+1][k]);
  }
};
```

Square Root Decomposition (MOs)

```
struct query{
  int l,r,id;
};
int cnt[10000000];
vector<int> ans,ar;
int l=0,r=-1,sum=0,k;

bool cmp(query &a,query &b){
  int block_a=a.l/k,block_b=b.l/k;
  if(block_a!=block_b)
    return block_a<block_b;
  return block_a&1?a.r<b.r:a.r>b.r;
}

void add(int x){
  cnt[ar[x]]++;
  if(cnt[ar[x]]==1) sum++;
}
void remove(int x){
  cnt[ar[x]]--;
  if(cnt[ar[x]]==0) sum--;
}
int solve(){
  int n,Q;
  cin>>n;
  ar.resize(n);
  for(int i=0;i<n;i++){
    cin>>ar[i];
  }
  cin>>Q;
  vector<query> q;
  ans.resize(Q);
  for(int i=0;i<Q;i++){
    int li,ri;
    cin>>li>>ri;
    li--;ri--;
    q.push_back({li,ri,i});
  }
  if(Q==0)Q++;
  k=sqrt(n*n/Q);
  if(k==0)k++;
  sort(q.begin(),q.end(),cmp);

  for(auto x:q){
    while(l<x.l) remove(l++);
    while(l>x.l) add(--l);
    while(r<x.r) add(++r);
    while(r>x.r) remove(r--);
    ans[x.id]=sum;
  }
  for(auto x:ans) cout<<x<<endl;
  return 0;
}
```

Trie

```
struct Node{
  int nxt[26],cnt;
  Node(){fill(nxt,nxt+26,0),cnt=0;}
};

struct Trie{
  vector<Node>t;
  Trie(){t.push_back(Node());}
  void add(string s){
    int cur=0;
    for(auto c:s){
      int to=c-'a';
      if(!t[cur].nxt[to]){
        t[cur].nxt[to]=sz(t);
        t.push_back(Node());
      }
      cur=t[cur].nxt[to];
    }
    t[cur].cnt++;
  }
  int get(string s){
    int cur=0;
    for(int i=0;i<sz(s);i++){
      int to=s[i]-'a';
      if(!t[cur].nxt[to])return 0;
      cur=t[cur].nxt[to];
    }
    return t[cur].cnt;
  }
};
```

Geometry

Convex Hull

```
vector<Point> Convex_Hull(vector<Point>points){
  int n=points.size();
  sort(points.begin(),points.end());
  vector<Point>hull;
  for(int rep=0;rep<2;rep++){
    int s=hull.size();
    for(int i=0;i<n;i++){
      while((int)hull.size()>=s+2){
        Point A=hull.end()[-2];
        Point B=hull.end()[-1];
        Point C=points[i];
        if(A.triangle(B,C)<=0)
          break;
        hull.pop_back();
      }
      hull.push_back(points[i]);
    }
    hull.pop_back(); // comment this line if necessary
    reverse(points.begin(),points.end());
  }
  return hull;
}
```

Geo Ashik

```
struct Point{
  int x,y;
  void read(){
    cin>>x>>y;
  }
  Point operator-(const Point &other)const{
    return Point{x-other.x,y-other.y};
  }
  void operator-=(Point &other){
    x-=other.x;
    y-=other.y;
  }
  int operator*(const Point &other){
    return x*other.y-y*other.x;
  }
  int triangle(const Point &b,const Point &c)const{
    return(b-*this)*(c-*this);
  }
  bool operator<(const Point &other)const{
    return make_pair(x,y)<make_pair(other.x,other.y);
  }
};
bool intersect(Point p1,Point p2,Point p3,Point p4){
  Point x=p2-p1;
  Point y=p4-p3;
  if(x*y==0){
    y=p3-p1;
    if(x*y!=0){
      return false;
    }
    for(int rep=0;rep<2;rep++){
      if(max(p1.x,p2.x)<min(p3.x,p4.x)||max(p1.y,p2.y)<min(p3.y,p4.y)){
        return false;
      }
      swap(p1,p3);
      swap(p2,p4);
    }
    return true;
  }
  for(int rep=0;rep<2;rep++){
    x=p2-p1;
    y=p3-p1;
    int sign1=p1.triangle(p2,p3);
    y=p4-p1;
    int sign2=p1.triangle(p2,p4);
    if((sign1<0&&sign2<0)||(sign1>0&&sign2>0)){
      return false;
    }
    swap(p1,p3);
    swap(p2,p4);
  }
  return true;
}
```

Geometry Template

```
const double eps=1e-9;
const double PI=acos(-1.0);
int sign(double x){
  return(x>eps)-(x<-eps);
}
struct P{
  double x,y;
  P(){x=y=0;}
  P(double x,double y):x(x),y(y){}
  void read(){cin>>x>>y;}
  P operator+(const P &b)const{return P{x+b.x,y+b.y};}
  void operator+=(const P &b){
    x+=b.x;
    y+=b.y;
  }
  bool operator==(P a)const{return(sign(a.x-x)==0&&sign(a.y-y)==0);}
  bool operator<(P a)const{return sign(a.x-x)==0?y<a.y:x<a.x;}
  bool operator>(P a)const{return sign(a.x-x)==0?y>a.y:x>a.x;}
  bool operator!=(P a)const{return!*this==a;}
};
double norm(P p){return sqrt(p.x*p.x+p.y*p.y);}
double arg(P p){return atan2(p.y,p.x);}
// tan^-1(y/x) value of theta in radian
inline double dot(P a,P b){return a.x*b.x+a.y*b.y;}
double dist(P a,P b){return sqrt(dot(a-b,a-b));}
double cross(P a,P b){return a.x*b.y-a.y*b.x;}
double cross2(P a,P b,P c){return cross(b-a,c-a);}
int orientaion(P a,P b,P c){return sign(cross(b-a,c-a));}
P perp(P a){return P{-a.y,a.x};}
double deg_to_rad(double d){return d*PI/180.0;}
double rad_to_deg(double r){return r*180.0/PI;}
double get_angle(P a,P b){
  double costheta=dot(a,b)/norm(a)/norm(b);
  return acos(max((double)-1.0,min((double)1.0,costheta)));
}
P rotate(P p,double theta){
  // rotate p by theta degree ccw w.r.t origin(0,0)
  double rad=deg_to_rad(theta);
  P res;
  res.x=(p.x*cos(rad)-p.y*sin(rad));
  res.y=(p.x*sin(rad)+p.y*cos(rad));
  return res;
}
bool segParallel(P a,P b,P c,P d){
  return abs(cross(a-b,c-d))<eps;
}
// If point p in the segement of ab
bool pointOnSeg(P p,P a,P b){
  if(dist(p,b)<eps||dist(p,a)<eps){
    return true;
  }
  return(segParallel(p,a,p,b)&&dot(p-a,p-b)<0);
}
bool pointOnPloygon(const P &p,const vector<P> &points){
  int n=sz(points);
  for(int i=0;i<n;i++){
    if(pointOnSeg(p,points[i],points[(i+1)%n])){
      return true;
    }
  }
  return false;
}

bool pointInside(const P &p,const vector<P> &points){
  int n=sz(points);
  bool ok=false;
  for(int i=0;i<n;i++){
    int j=(i+1)%n;
    if((p.y<points[i].y!=p.y<points[j].y)&&(p.x<points[i].x+(points[j].x-points[i].x)*(p.y-points[i].y)/(points[j].y-points[i].y))){
      ok=!ok;
    }
  }
  return ok;
}
bool ccw(P p,P q,P r){
  return cross(q-p,r-p)<eps;
}
vector<P> convex_hull(vector<P> points){
  //--Incremental algorithm--
  // upper hull
  sort(points.begin(),points.end());
  stack<P> stk_up;
  stk_up.push(points[0]);
  stk_up.push(points[1]);
  for(int i=2;i<sz(points);i++){
    while(sz(stk_up)>=2){
      P p1,p2;
      p1=stk_up.top();
      stk_up.pop();
      p2=stk_up.top();
      if(ccw(points[i],p1,p2)){
        stk_up.push(p1);
        break;
      }
    }
    stk_up.push(points[i]);
  }
  // lower hull
  for(int i=0;i<sz(points);i++){
    points[i].x=-points[i].x;
    points[i].y=-points[i].y;
  }
  sort(all(points));
  stack<P> stk_low;
  stk_low.push(points[0]);
  stk_low.push(points[1]);

  for(int i=2;i<sz(points);i++){
    while(sz(stk_low)>=2){
      P p1,p2;
      p1=stk_low.top();
      stk_low.pop();
      p2=stk_low.top();
      if(ccw(points[i],p1,p2))
      {
        stk_low.push(p1);
        break;
      }
    }
    stk_low.push(points[i]);
  }
  // Print ch cw order from leftmost point
  vector<P> CH;
  stk_low.pop();
  P p;
  while(!stk_low.empty()){
    p=stk_low.top();
    p*= -1.0;
    CH.push_back(p);
    stk_low.pop();
  }
  stk_up.pop();

  while(!stk_up.empty()){
    CH.push_back(stk_up.top());
    stk_up.pop();
  }
  reverse(all(CH));
  return CH;
}

struct Line{ // ax + by + c = 0
  double a,b,c;
};

Line pointToLine(P a,P b){
  Line l;
  if(fabs(a.x-b.x)<eps){
    l.a=1.0,l.b=0.0,l.c=-a.x;
  }
  else{
    l.a=-(a.y-b.y)/(a.x-b.x);
    l.b=1.0;
    l.c=-(l.a*a.x)-a.y;
  }
  return l;
}

Line pointSlopeToLine(P p,double
 ↪  m){
  Line l;
  l.a=-m,l.b=1;
  l.c=-((l.a*p.x)+(l.b*p.y));
  return l;
}
// Two line are parallel or not
bool areParallel(Line l1,Line l2){
  return(fabs(l1.a-l2.a)<eps)&&(f⌋
   ↪  abs(l1.b-l2.b)<eps);
}
// Two line same or not
bool areSame(Line l1,Line l2){
  return areParallel(l1,l2)&&(fab⌋
   ↪  s(l1.c-l2.c)<eps);
}

bool areIntersect(Line l1,Line
 ↪  l2,P &p){
  if(areParallel(l1,l2))
    return false;
  // solve system of 2 linear
   ↪  algebric
  //  eqn with 2 unknowns
  p.x=(l2.b*l1.c-l1.b*l2.c)/(l2.a⌋
   ↪  *l1.b-l1.a*l2.b);
  // special case: test for
   ↪  vertical
  //  line to avoid divison by
   ↪  zero
  if(fabs(l1.b)>eps)
    p.y=-(l1.a*p.x+l1.c);
  else
    p.y=-(l2.a*p.x+l2.c);
  return true;
}
// retrurn true if r in the same
 ↪  line of pq
bool collinear(P p,P q,P r){
  return fabs(cross(q-p,r-p))<eps;
}
// Perpendicular to l and pass
 ↪  through p
P closestPoint(Line l,P p){
  Line perp;
  P r;
  if(fabs(l.b)<eps){
    r.x=-l.c,r.y=p.y;
    return r;
  }
  if(fabs(l.a)<eps){
    r.x=p.x,r.y=-l.c;
    return r;
  }
  // normal line
  perp=pointSlopeToLine(p,1/l.a);
  // intersect lien l with this
   ↪  perp line
  //  the intersection point is
   ↪  the closest point
  areIntersect(l,perp,r);
  return r;
}
```

Graphs

Cycle Finding Directed Graph

```
bool isCycDG(int u) {
  vis[u]=1;
  for(int v:g[u]) {
    if(!vis[v]) { par[v]=u;
      if(isCycDG(v))
        return true;
    } else if(vis[v]==1) {
      b=v,e=u;
      return true;
    }
  }
  vis[u]=2; return false;
}
```

Cycle Finding Undirected Graph

```
bool isCycUG(int u,int p=-1) {
  vis[u]=1;
  for(auto v:g[u]) {
    if(!vis[v]) {
      par[v]=u;
      if(isCycUG(v,u))return true;
    } else if(v!=p) {
      b=v,e=u;
      return true;
    }
  }
  return false;
}
vector<int> find_cycle(int n) {
  b=-1;
  for(int v=1;v<=n;v++) {
    if(vis[v]==0&&isCycUG(v))brea⌋
     ↪  k;
  }
  if(b==-1) {
    return vector<int>();
  } else {
    vector<int> cycle;
    for(int v=e;v!=b;v=par[v]) {
      cycle.push_back(v);
    }
    cycle.push_back(b);
    reverse(all(cycle));
    return cycle;
  }
}
```

Finding Articulation Points

```
int in[N],low[N],timer;
vector<int> g[N], cut_vertex;
void dfs(int u,int p=-1){
  in[u] = low[u] = ++timer;
  for(auto v:g[u]){
    if(v == p)continue;
    if(in[v] == 0){ dfs(v,u);
      low[u]=min(low[v],low[u]);
      if(low[v]>=in[u]&&p!=-1)
        cut_vertex.push_back(u);
    } else
      low[u]=min(low[u],in[v]);
  }
  if(p==-1&&sz(g[u])>1){
    cut_vertex.push_back(u);
  }
}
```

Finding Bridges

```
int in[N],low[N],timer;
vector<int>g[N];
vector<ar<int,2>>bridges;

void dfs(int u,int p=-1){
  in[u]=low[u]=++timer;
  for(auto v:g[u]){
    if(v==p)continue;
    if(in[v]==0){
      dfs(v,u);
      low[u]=min(low[v],low[u]);
      if(low[v]>in[u])
        bridges.push_back({u,v});
    }else{
      low[u]=min(low[u],in[v]);
    }
  }
}
```

LCA Lowest Common Ancestor

```
int anc[N][25],d[N];
vector<int>g[N];
void dfs(int u=0,int p=-1){
  anc[u][0]=p;
  for(int i=1;i<19;i++)
    anc[u][i]=~anc[u][i-1]?anc[an⌋
     ↪  c[u][i-1]][i-1]:-1;
  for(int v:g[u]){
    if(v==p)continue;
    d[v]=d[u]+1;
    dfs(v,u);
  }
}
int lca(int u,int v){
  if(d[u]<d[v])swap(u,v);
  for(int i=18;~i;i--)
    if(d[u]-(1<<i)>=d[v])
      u=anc[u][i];
  if(u==v)return u;
  for(int i=18;~i;i--)
    if(anc[u][i]^anc[v][i])
      u=anc[u][i],v=anc[v][i];
  return anc[u][0];
}
int dia(int u,int v){
  return d[u]+d[v]-2*d[lca(u,v)];
}
```

Strongly Connected Component (Kosaraju)

```
vector<int> g[N],gr[N];
vector<int> vis,order,cmp;
//For topological order
void dfs1(int u){
  vis[u] = 1;
  for(auto v: g[u]){
    if(!vis[v])dfs1(v);
  }
  order.push_back(u);
}
void dfs2(int u){
  vis[u] = 1;
  cmp.push_back(u);
  for(auto v: gr[u]){
    if(!vis[v])dfs2(v);
  }
}
void ssc(int n){
  vis.resize(n+1,0);
  for(int i = 1;i <= n;i++){
    if(!vis[i])dfs1(i);
  }

  reverse(all(order));
  vis = vector<int>(n+1,0);

  for(auto v: order){
    if(!vis[v]){
      cmp.clear();
      dfs2(v);
      print(cmp);
    }
  }
}
```

Flow

Max Flow Min Cut Edmonds-Karp

```
// cap[a][b] = Capacity left from
↪   a to b
// iflow = initial flow, icap =
↪   initial capacity
// pathMinCap = capacity
↪   bottleneck for a path (s->t)

typedef int T;
vector<int> level;
vector<vector<int>> adj, cap;
T inf = 1 << 30;

void init(int N) {
  adj.assign(N, vector<int>());
  cap.assign(N, vector<int>(N));
}

void addEdge(int u, int v, T
↪   icap, T iflow = 0) {
  if (!cap[u][v])
    adj[u].push_back(v),
    ↪   adj[v].push_back(u);
  cap[u][v] = icap - iflow;
  // cap[v][u] = cap[u][v]; // if
  ↪   graph is undirected
}
// O(N)
T bfs(int s, int t, vector<int>
↪   &dad) {
  dad.assign(adj.size(), -1);
  queue<pair<int, T>> q;
  dad[s] = s, q.push(s);
  while (q.size()) {
    int u = q.front().first;
    T pathMinCap =
    ↪   q.front().second;
    q.pop();
    for (int v : adj[u])
      if (dad[v] == -1 &&
      ↪   cap[u][v]) {
        dad[v] = u;
        T flow = min(pathMinCap,
        ↪   cap[u][v]);
        if (v == t) return flow;
        q.push({v, flow});
      }
  }
  return 0;
}
// O(E^2 * V)
T maxFlowMinCut(int s, int t) {
  T maxFlow = 0;
  vector<int> dad;
  while (T flow = bfs(s, t, dad))
  ↪   {
    maxFlow += flow;
    int u = t;
    while (u != s) {
      cap[dad[u]][u] -= flow,
      ↪   cap[u][dad[u]] += flow;
      u = dad[u];
    }
  }
  return maxFlow;
}
```

Max Flow Min Cut Dinic

```
// cap[a][b] = Capacity from a to
↪   b
// flow[a][b] = flow occupied from
↪   a to b
// level[a] = level in graph of
↪   node a
// iflow = initial flow, icap =
↪   initial capacity
// pathMinCap = capacity
↪   bottleneck for a path (s->t)

typedef int T;
vector<int> level;
vector<vector<int>> adj;
vector<vector<T>> cap, flow;
T inf = 1 << 30;

void init(int N) {
  adj.assign(N, vector<int>());
  cap.assign(N, vector<int>(N));
  flow.assign(N, vector<int>(N));
}

void addEdge(int u, int v, T
↪   icap, T iflow = 0) {
  if (!cap[u][v])
    adj[u].push_back(v),
    ↪   adj[v].push_back(u);
  cap[u][v] += icap;
  // cap[v][u] = cap[u][v]; // if
  ↪   graph is undirected
  flow[u][v] += iflow, flow[v][u]
  ↪   -= iflow;
}
bool levelGraph(int s, int t) {
  level.assign(adj.size(), 0);
  level[s] = 1;
  queue<int> q;
  q.push(s);
  while (!q.empty()) {
    int u = q.front();
    q.pop();
    for (int &v : adj[u]) {
      if (!level[v] && flow[u][v]
      ↪   < cap[u][v]) {
        q.push(v);
        level[v] = level[u] + 1;
      }
    }
  }
  return level[t];
}
T blockingFlow(int u, int t, T
↪   pathMinCap) {
  if (u == t) return pathMinCap;
  for (int v : adj[u]) {
    T capLeft = cap[u][v] -
    ↪   flow[u][v];
    if (level[v] == (level[u] +
    ↪   1) && capLeft > 0)
      if (T pathMaxFlow =
      ↪   blockingFlow(
          v, t, min(pathMinCap,
          ↪   capLeft))) {
        flow[u][v] += pathMaxFlow;
        flow[v][u] -= pathMaxFlow;
        return pathMaxFlow;
      }
  }
  return 0;
}
// O(E * V^2)
T maxFlowMinCut(int s, int t) {
  if (s == t) return inf;
  T maxFlow = 0;
  while (levelGraph(s, t))
    while (T flow =
    ↪   blockingFlow(s, t, inf))
      maxFlow += flow;
  return maxFlow;
}
```

Shortest Paths

Dijkstra

```
vector<ar<int,2>>g[N];
void dijkstra(int src) {
  vector<int>d(N,M);
  priority_queue<ar<int,2>,
  ↪   vector<ar<int,2>>,greater<a⌋
  ↪   r<int,2>>>q;
  d[src]=0;
  q.push({0,src});
  while(q.size()) {
    array<int,2> u=q.top();
    q.pop();
    if(u[0]>d[u[1]])continue;
    for(array<int,2>v:g[u[1]]) {
      if(d[v[1]]>u[0]+v[0]) {
        d[v[1]]=u[0]+v[0];
        q.push({d[v[1]],v[1]});
      }
    }
  }
}
```

Floyed Warshal

```
for(int k=0;k<n;++k)
  for(int i=0;i<n;++i)
    for(int j=0;j<n;++j)
      d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
```

Math

Combination NCR

```
// calculate nCr in O( max(n) )
// pre-calculation and O(1) query
class NCR{
public:
  vector<int>fact;
  vector<int>inv_fact;
  NCR(int n){
    fact.resize(n+1);
    inv_fact.resize(n+1);
    fact[0]=1;
    for(int i=1;i<=n;i++){
      fact[i]=(fact[i-1]*i)%Mod;
    }
    // inv[(x - 1)!] = x * inv[x!]
    inv_fact[n]=bigmod(fact[n],Mod-2);
    for(int i=n-1;i>=0;i--){
      inv_fact[i]=(inv_fact[i+1]*(i+1))%Mod;
    }
  }
  int ncr(int n,int r){
    return (((fact[n]*inv_fact[r])%Mod)*inv_fact[n-r])%Mod;
  }
private:
  const int Mod=1e9+7;
};

//    NCR cal(1000000);
//    cout << cal.ncr(x, y) << endl;
```

CRT Chines Reminder Theorem

```
int ext_gcd(int a,int b,int &x,int &y)
{
  x=1ll,y=0ll;
  int x1=0,y1=1,a1=a,b1=b;
  while(b1){
    int q=a1/b1;
    tie(x,x1)=make_tuple(x1,x-q*x1);
    tie(y,y1)=make_tuple(y1,y-q*y1);
    tie(a1,b1)=make_tuple(b1,a1-q*b1);
  }
  return a1;
}

class ChineseRemainderTheorem
{
  typedef long long vlong;
  typedef pair<vlong,vlong> pll;

  vector<pll> equations;

public:
  void clear() {
    equations.clear();
  }
  void addEquation(vlong r,vlong m){
    equations.push_back({r,m});
  }
  pll solve()
  {
    if (equations.size() == 0)
      return {-1,-1};
    vlong a1 = equations[0].first;
    vlong m1 = equations[0].second;
    a1 %= m1;

    for(int i=1;i<equations.size();i++)
    {
      vlong a2 = equations[i].first;
      vlong m2 = equations[i].second;

      vlong g = __gcd(m1,m2);
      if (a1 % g != a2 % g)
        return {-1,-1};

      vlong p,q;
      ext_gcd(m1/g,m2/g,p,q);

      vlong mod = m1/g*m2;
      vlong x = ((__int128)a1*(m2/g)%mod*q%mod+(__int128)a2*(m1/g)%mod*p%mod)%mod;
      a1 = x;
      if (a1<0)
        a1+=mod;
      m1=mod;
    }
    return {a1,m1};
  }
};
```

Diophantine Equation

```
int gcd(int a,int b,int& x,int& y) {
  if(!b) {x=1,y=0;return a;}
  int x1,y1;
  int d=gcd(b,a%b,x1,y1);
  x=y1;
  y=x1-y1*(a/b);
  return d;
}

bool any_solution(int a,int b,int c,int &x0,int &y0) {
  int g=gcd(abs(a),abs(b),x0,y0);
  if(c%g)return false;
  x0*=c/g;
  y0*=c/g;
  if(a<0)x0=-x0;
  if(b<0)y0=-y0;
  return true;
}
```

```
void shift(int& x,int& y,int a,int b,int cnt) {
  x+=cnt*b;
  y-=cnt*a;
}

int all_solutions(int a,int b,int c,int minx,int maxx,int miny,int maxy) {
  int x,y,g;
  if(!any_solution(a,b,c,x,y))return 0;
  a/=g;
  b/=g;
  int sign_a=a>0?+1:-1;
  int sign_b=b>0?+1:-1;
  shift(x,y,a,b,(minx-x)/b);
  if(x<minx)shift(x,y,a,b,sign_b);
  if(x>maxx)return 0;

  int lx1=x;
  shift(x,y,a,b,(maxx-x)/b);
  if(x>maxx)shift(x,y,a,b,-sign_b);

  int rx1=x;
  shift(x,y,a,b,-(miny-y)/a);
  if(y<miny)shift(x,y,a,b,-sign_a);
  if(y>maxy)return 0;

  int lx2=x;
  shift(x,y,a,b,-(maxy-y)/a);
  if(y>maxy)shift(x,y,a,b,sign_a);

  int rx2=x;
  if(lx2>rx2)swap(lx2,rx2);
  int lx=max(lx1,lx2);
  int rx=min(rx1,rx2);
  if(lx>rx)return 0;
  return (rx-lx)/abs(b)+1;
}
```

Euler Totient Function

```
int phi(int n)
{
  int ans = n;
  for(int i=2;i<=sqrt(n);i++)
  {
    if(n%i==0)
    {
      while(n%i==0)
        n/=i;
      ans -= ans/i;
    }
  }
  if(n>1) ans -= ans/n;
  return ans;
}
```

Euler Totient with sieve

```
vector<int> seive_phi(int n)
{
  vector<int>phi(n+1);
  for(int i=2;i<=n;i++)
    phi[i] = i-1;
  for(int i=2;i<=n;i++)
  {
    for(int j=2*i;j<=n;j+=i)
    {
      phi[j]-=phi[i];
    }
  }
  return phi;
}
```

Extended Euclide Algorithm

```
int extEuclid(int a,int b,int& x,int& y) {
  int x1=y=0,y1=x=1;
  while(b) {
    int q=a/b,t=b;b=a%b;a=t;
    t=x1;x1=x-q*x1;x=t;
    t=y1;y1=y-q*y1;y=t;
  }
  return a;
}

int deqn(int a,int b,int c,int& x,int& y,int& g) {
  g=extEuclid(a,b,x,y);
  if(c%g)return 0;
  x*=c/g;y*=c/g;
  return 1;
}

vector<pair<int,int>> deqn_sol(int a,int b,int c,int x,int y,int g) {
  if(!deqn(a,b,c,x,y,g)) {
    cout<<"NO SOLUTION"<<endl;
  }
  vector<pair<int,int>> ans;
  if(c%g)return ans;
  while(x>0) {
    x-=b/g;
    y+=a/g;
  }
  while(x<0) {
    x+=b/g;
    y-=a/g;
  }
  while(y>=0) {
    ans.push_back({x,y});
    x+=b/g;
    y-=a/g;
  }
  return ans;
}
```

Lucas Theorem

```
ll C(ll n, ll k) {
  if (n < k) return 0;
  if (n >= MOD) return (C(n%MOD, k%MOD) * C(n/MOD, k/MOD)) % MOD;
  return (((fact[n] * inv(fact[k]))%MOD) * inv(fact[n-k])) % MOD;
}
```

Moduler Inverse

```
bigmod(a, M - 2, M);
```

Segmented Sieve

```
int lpf[N];
vector<int>pfs;
vector<int> sieve() {
  for(int i=2; i<N; i++) {
    if(!lpf[i]) {
      pfs.push_back(i);
      lpf[i]=i;
    }
    for(int j=0; j<pfs.size() && pfs[j]<=lpf[i] && i*pfs[j]<N; j++)
      lpf[i*pfs[j]]=pfs[j];
  }
  return pfs;
}

vector<int> segSieve(int l,int r) {
  bool isPrime[r-l+1];
  vector<int>prime=sieve(),p;
  for(auto &a:isPrime)a=true;

  for(int i=0; prime[i]*prime[i]<=r; i++) {
    int cp=prime[i];
    int base=(l/cp)*cp;
    if(base<cp)base+=cp;
    for(int j=base; j<=r; j+=cp)
      isPrime[j-l]=false;
    if(base==cp)isPrime[base-l]=true;
  }
  for(int i=0; i<r-l+1; i++) {
    if(isPrime[i]==true)p.push_back(i+l);
  }
  return p;
}
```

Big Integer

Big Integer Addition

```
string Add(string str1, string str2){
  if(str1.length()>str2.length())
    swap(str1,str2);
  string str="";
  int n1=str1.length(),n2=str2.length();
  reverse(str1.begin(),str1.end());
  reverse(str2.begin(),str2.end());
  int carry=0;
  for(int i=0;i<n1;i++){
    int sum=((str1[i]-'0')+(str2[i]-'0')+carry);
    str.push_back(sum%10+'0');
    carry=sum/10;
  }
  for(int i=n1;i<n2;i++){
    int sum=((str2[i]-'0')+carry);
    str.push_back(sum%10+'0');
    carry=sum/10;
  }
  if(carry)
    str.push_back(carry+'0');
  reverse(str.begin(),str.end());
  return str;
}
```

Big Integer Division

```
string bigIntegerDivision(string num1,string num2,int precision){
  if(num2=="0")return"Error: Division by zero!";
  string result="";
  bool negative=(num1[0]=='-')^(num2[0]=='-');
  if(num1[0]=='-')num1=num1.substr(1);
  if(num2[0]=='-')num2=num2.substr(1);
  int len1=num1.size(),len2=num2.size();
  int carry=0,i=0;
  string quotient;
  while(i<len1){
    carry=carry*10+(num1[i]-'0');
    int quotientDigit=carry/stoi(num2);
    carry=carry%stoi(num2);
    quotient+=to_string(quotientDigit);
    i++;
  }
  size_t start=quotient.find_first_not_of('0');
  if(start!=string::npos)quotient=quotient.substr(start);
  else quotient="0";
  if(quotient.empty())quotient="0";
  if(precision<=0)return quotient;
  result=quotient+".";
  while(precision>0){
    carry=carry*10;
    int fractionalDigit=carry/stoi(num2);
    carry=carry%stoi(num2);
    result+=to_string(fractionalDigit);
    precision--;
  }
  if(negative&&result!="0.")result="-"+result;
  return result;
}
```

Big Integer Library in Java

```
General Arithmetic :
BigInteger.intValue();
BigInteger.add(BigInteger b); // a + b
BigInteger.subtract(BigInteger b); // a - b
BigInteger.multiply(BigInteger b); // a * b
BigInteger.divide(BigInteger b); // a / b
BigInteger.pow(int p); // a ^ p
BigInteger.remainder(BigInteger m); // a % m
BigInteger.mod(BigInteger m); // a % m
BigInteger.modInverse(BigInteger m); // a^-1 % m
BigInteger.modPow(BigInteger p, BigInteger m); // a^p % m
BigInteger.negate(); // -a
BigInteger.not(); // ~a
BigInteger.and(BigInteger b); // a & b
BigInteger.andNot(BigInteger b); // a & ~b
BigInteger.or(BigInteger b); // a | b
BigInteger.xor(BigInteger b);
BigInteger.shiftLeft(int n); // a << n
BigInteger.shiftRight(int n); // a >> n
BigInteger.max(BigInteger b); // max(a, b)
BigInteger.min(BigInteger b); // min(a, b)
BigInteger.toString(int b); // to base convertor
You can also check the large number
is prime or not using isProbablePrime() method
BigInteger num = new BigInteger("121020010201001039");
System.out.println(num.isProbablePrime(100)); // true
// Also if you want next prime
// after given number you can use
// nextProabablePrime() method:

Library : import java.math.BigInteger;
Input from stdin : BigInteger bi = sc.nextBigInteger();
Constants: BigInteger.ZERO BigInteger.ONE BigInteger.TEN
```

Big Integer Multiply

```
string Multiply(string num1, string num2){
  int len1=num1.size(),len2=num2.size();
  if(len1==0||len2==0)return"0";
  vector<int>result(len1+len2,0);
  int i_n1=0,i_n2=0;
  for(int i=len1-1;i>=0;i--){
    int carry=0,n1=num1[i]-'0';
    i_n2=0;
    for(int j=len2-1;j>=0;j--){
      int n2=num2[j]-'0';
      sum=n1*n2+result[i_n1+i_n2]+carry;
      carry=sum/10;
      result[i_n1+i_n2]=sum%10;
      i_n2++;
    }
    if(carry>0)result[i_n1+i_n2]+=carry;
    i_n1++;
  }
  int i=result.size()-1;
  while(i>=0&&result[i]==0)i--;
  if(i==-1)return"0";
  string s="";
  while(i>=0)s+=to_string(result[i--]);
  return s;
}
```

Big Integer Substraction

```
bool isSmaller(string str1,string str2){
  int n1=str1.length(),n2=str2.length();
  if(n1<n2)return true;
  if(n2<n1)return false;
  for(int i=0;i<n1;i++)
    if(str1[i]<str2[i])return true;
    else if(str1[i]>str2[i])return false;
  return false;
}
string findDiff(string str1,string str2){
  if(isSmaller(str1,str2))swap(str1,str2);
  string str="";
  int n1=str1.length(),n2=str2.length();
  reverse(str1.begin(),str1.end());
  reverse(str2.begin(),str2.end());
  int carry=0;
  for(int i=0;i<n2;i++){
    int sub=((str1[i]-'0')-(str2[i]-'0')-carry);
    if(sub<0){sub=sub+10;carry=1;}
    else carry=0;
    str.push_back(sub+'0');
  }
  for(int i=n2;i<n1;i++){
    int sub=((str1[i]-'0')-carry);
    if(sub<0){sub=sub+10;carry=1;}
    else carry=0;
    str.push_back(sub+'0');
  }
  reverse(str.begin(),str.end());
  return str;
}
```

String Algorithms

Aho Corasick

```
string t;
int n,node,par[N],d[N],pl[N],sl[N],trie[N][150];
int nxt[N][150],ans[N],vis[N],a[N],lev[N],mn[N];
vector<int>tr[N],qr;
int cnt[N];
void ins(string &s) {
  int cur=0;
  for(auto it: s) {
    int c=it;
    if(!trie[cur][c]) {
      trie[cur][c]=++node;
      d[node]=d[cur]+1;
      par[node]=cur;
      pl[node]=c;
    }
    cur=trie[cur][c];
  }
  qr.push_back(cur);
}
void push_link() {
  queue<int>q;
  q.push(0);
  while(sz(q)) {
    int v=q.front();
    q.pop();
    if(d[v]<=1)sl[v]=0;
    else {
      int u=sl[par[v]];
      int l=pl[v];
      while(u>0 && !trie[u][l])
        u=sl[u];
      if(trie[u][l])u=trie[u][l];
      sl[v]=u;
    }
    if(v!=0)tr[sl[v]].push_back(v);

    for(int i=0; i<150; i++)
      if(trie[v][i])
        q.push(trie[v][i]);
  }
}

int jump(int cur, int id) {
  if(nxt[cur][id])
    return nxt[cur][id];
  int u=cur;
  while(cur>0 && !trie[cur][id])
    cur=sl[cur];
  if(trie[cur][id])
    cur=trie[cur][id];
  return nxt[u][id]=cur;
}

void Search() {
  int cur=0;
  for(int i=0; i<sz(t); i++) {
    int c=t[i];
    while(cur>0 && !trie[cur][c])
      cur=sl[cur];
    cur=trie[cur][c];
    cnt[cur]++;
  }
}
void dfs(int u) {
  vis[u]=1;
  for(auto v: tr[u]) {
    if(!vis[v])dfs(v);
    cnt[u]+=cnt[v];
  }
}
void solve() {
  push_link(); Search();
  for(int i=0; i<n; i++) {
    if(!vis[qr[i]])dfs(qr[i]);
    cout<<cnt[qr[i]]<<endl;
  }
}
```

Hashing

```
struct Hash{
  string s;
  const int p=397,p1=313;
  int len;
  vector<int> pw1,pw,hF,hF1,hR,hR1;

  Hash(string s1){
    s=s1;
    this->len=sz(s);
    pw=hF=hR=vector<int>(len+5,0);
    pw1=hF1=hR1=vector<int>(len+5,0);
  }

  void Calc(){
    pw[0]=1;
    hF[0]=hR[len+1]=0;
    for(int i=1;i<=len;i++){
      pw[i]=(pw[i-1]*p)%M;
    }
    for(int i=0;i<len;i++){
      hF[i+1]=(hF[i]*p+(s[i]))%M;
      hR[len-i]=(hR[len-i+1]*p+(s[len-i-1]))%M;
    }
  }

  int hashF(int l,int r){
    int val=hF[r]-(hF[l-1]*pw[r-l+1])%M;
    if(val<0)val+=M;
    return val;
  }

  int hashR(int l,int r){
    int val=hR[l]-(hR[r+1]*pw[r-l+1])%M;
    if(val<0)val+=M;
    return val;
  }

  bool isPalin(int l,int r){
    if(r<l)return false;
    return (hashF(l,r)==hashR(l,r));
  }

  void Calc1(){
    pw1[0]=1;
    hF1[0]=hR1[len+1]=0;
    pw[0]=1;
    hF[0]=hR[len+1]=0;
    for(int i=1;i<=len;i++){
      pw1[i]=(pw1[i-1]*p1)%M;
      pw[i]=(pw[i-1]*p)%M;
    }
    for(int i=0;i<len;i++){
      hF1[i+1]=(hF1[i]*p1+(s[i]))%M;
      hF[i+1]=(hF[i]*p+s[i])%M;
      hR1[len-i]=(hR1[len-i+1]*p1+(s[len-i-1]))%M;
      hR[len-i]=(hR[len-i+1]*p+(s[len-i-1]))%M;
    }
  }

  ar<int,2> hashF1(int l,int r){
    int val1=hF1[r]-(hF1[l-1]*pw1[r-l+1])%M;
    int val2=hF[r]-(hF[l-1]*pw[r-l+1])%M;
    if(val1<0)val1+=M;
    if(val2<0)val2+=M;
    return {val1,val2};
  }

  ar<int,2> hashR1(int l,int r){
    int val1=hR1[l]-(hR1[r+1]*pw1[r-l+1])%M;
    int val2=hR[l]-(hR[r+1]*pw[r-l+1])%M;
    if(val1<0)val1+=M;
    if(val2<0)val2+=M;
    return {val1,val2};
  }

  bool isPalin1(int l,int r){
    if(r<l)return false;
    return (hashF1(l,r)==hashR1(l,r));
  }
};
```

KMP Knuth Morris Pratt

```
vector<int> prefix_function(string s){
  int n=(int)s.length();
  vector<int> pi(n);
  for(int i=1;i<n;i++){
    int j=pi[i-1];
    while(j>0&&s[i]!=s[j]) j=pi[j-1];
    if(s[i]==s[j]) j++;
    pi[i]=j;
  }
  return pi;
}
```

Manachers Algorithm

```
vector<int> pal_array(string s){
  string t=s; s="#";
  for(auto c:t)s+=c,s+="#";
  s="@"+s+"$";
  int n=s.size();
  vector<int> len(n+1);
  int l=1,r=1;
  for(int i=1;i<=n;i++){
    len[i]=min(r-i,len[l+(r-i)]);
    while(s[i-len[i]]==s[i+len[i]]) len[i]++;
    if(i+len[i]>r){
      l=i-len[i];
      r=i+len[i];
    }
  }
  return len;
}
```

String Automation

```
const int N=105,M=11,mod=1e9+7;
vector<int> prefix_function(string&s){
  int n=(int)s.size();
  vector<int> pi(n,0);
  for(int i=1;i<n;i++){
    int j=pi[i-1];
    while(j>0&&s[i]!=s[j])j=pi[j-1];
    if(s[i]==s[j])j++;
    pi[i]=j;
  }
  return pi;
}
int aut[N][26];
void compute_automaton(string s){
  s+='#';
  int n=(int)s.size();
  vector<int> pi=prefix_function(s);
  for(int i=0;i<n;i++){
    for(int c=0;c<26;c++){
      if(i>0&&'a'+c!=s[i])aut[i][c]=aut[pi[i-1]][c];
      else aut[i][c]=i+('a'+c==s[i]);
    }
  }
}
```

Suffix and LCP array

```
struct suffixArray{
  int n;
  string s;
  vector<int> sa,lcp;
  const int sigma=300;

  void cnt_sort(int k,const vector<int>& rnk){
    vector<int>cnt(max(sigma,n),0);
    for(int i=0;i<n;i++){
      cnt[(i+k<n)?rnk[i+k]:0]+=1;
    }
    int sum=0;
    for(int i=0;i<sz(cnt);i++){
      int ci=cnt[i];
      cnt[i]=sum;
      sum+=ci;
    }

    vector<int>tmp_sa(n);
    for(int i=0;i<n;i++){
      int pos=(sa[i]+k<n)?rnk[sa[i]+k]:0;
      tmp_sa[cnt[pos]++]=sa[i];
    }
    sa.swap(tmp_sa);
  }

  void construct_sa(){
    sa.resize(n);
    iota(all(sa),0);
    vector<int>rnk(n,0);
    for(int i=0;i<n;i++)rnk[i]=s[i];
    for(int k=1;k<n;k<<=1){
      cnt_sort(k,rnk);
      cnt_sort(0,rnk);
      int r=0;
      vector<int>tmp_rnk(n);
      tmp_rnk[sa[0]]=r;
      for(int i=1;i<n;i++){
        if(rnk[sa[i]]==rnk[sa[i-1]] && rnk[sa[i]+k]==rnk[sa[i-1]+k])
          tmp_rnk[sa[i]]=r;
        else tmp_rnk[sa[i]]=++r;
      }
      rnk.swap(tmp_rnk);
      if(rnk[sa[n-1]]==n-1)break;
    }
  }

  pair<int,int> find(const string& p){
    pair<int,int>ret;
    {
      int l=0,h=n-1;
      while(l!=h){
        int m=(l+h)/2;
        if(s.compare(sa[m],sz(p),p)>=0)
          h=m;
        else l=m+1;
      }
      if(s.compare(sa[l],sz(p),p)!=0)
        return {-1,-2};
      ret.first=l;
    }
    {
      int k=0;
      while((1<<k)<n)k+=1;
      int h=ret.first;
      for(int bit=k-1;bit>=0;bit--){
        if(h+(1<<bit)<n && s.compare(sa[h+(1<<bit)],sz(p),p)==0)
          h+=(1<<bit);
      }
      ret.second=h;
    }
    return ret;
  }

  void construct_lcp(){
    vector<int>rnk(n,0);
    for(int i=0;i<n;i++)rnk[sa[i]]=i;
    int k=0;
    lcp.resize(n-1,0);
    for(int i=0;i<n;i++){
      if(rnk[i]==n-1){
        k=0;
        continue;
      }
      int j=sa[rnk[i]+1];
      while(max(i,j)+k<n && s[i+k]==s[j+k])k++;
      lcp[rnk[i]]=k;
      k=max(0ll,k-1);
    }
  }

  suffixArray(const string& ss){
    s=ss;
    s+='!';
    n=sz(s);
    construct_sa();
    construct_lcp();
  }
};
```

Z Algorithm

```
vector<int> z_function(string s){
  int n=(int)s.length();
  vector<int> z(n);
  int l=0,r=0;
  for(int i=1;i<n;++i){
    if(i<=r)
      z[i]=min(r-i+1,z[i-l]);
    while(i+z[i]<n && s[z[i]]==s[i+z[i]])++z[i];
    if(i+z[i]-1>r)l=i,r=i+z[i]-1;
  }
  return z;
}
```

Extras

File input

```
//prime numbers
8229526783,494446522307,773954423953
91728987130369859,1454029476811019783
//fast
ios_base::sync_with_stdio(false);
cin.tie(NULL);
//file read
freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
```

Stress Testing

CommandLine Script

```
//save as name.bat
//run using run/start name.bat

@echo off
setlocal enabledelayedexpansion
g++ -o a a.cpp
g++ -o brute brute.cpp
g++ -o gen gen.cpp
if errorlevel 1 (
    echo Compilation error
    goto end
)
set i=1

:loop
echo !i!
gen.exe !i! > in
a.exe < in > out1
brute.exe < in > out2
fc out1 out2 > nul
if errorlevel 1 goto compare_end
:: Increment the counter
set /a i+=1
goto loop
:compare_end
echo "Your Output: "
type out1
echo "Correct Output: "
type out2
:end
```

Printing Grid Using Vim

```
Vim grid.txt
i+<esc>25A---+<esc>
o|<esc>25A   |<esc>
ggVGyG400pGdd
:wq<enter>
```

Random Generator

```
//For random number generation
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
inline int random(int l,int r) {
  return uniform_int_distribution<int>(l,r)(rng);
}
```

Shell Script

```
//a.cpp is my sol & brute.cpp is correct sol
//gen.cpp for input generation, "in" is text input text fil
//run using command: bash filename.sh
g++ -o a a.cpp
g++ -o brute brute.cpp
g++ -o gen gen.cpp
for((i = 1; ; ++i)); do
    echo $i
    ./gen $i > in
    # ./a < int > out1
    # ./brute < int > out2
    # diff -w out1 out2 || break
    diff -w <(./a < in) <(./brute < in) || break
done
```

Tree Generator

```
int main(int argc,char* argv[]){
  srand(atoi(argv[1]));
  int n=rand(2,20);
  printf("%d\n",n);
  vector<pair<int,int>>edges;
  for(int i=2;i<=n;++i){
    edges.emplace_back(rand(1,i-1),i);
  }
  // re-naming vertices
  vector<int>perm(n+1);
  for(int i=1;i<=n;++i){
    perm[i]=i;
  }
  random_shuffle(perm.begin()+1,perm.end());
  // random order of edges
  random_shuffle(edges.begin(),edges.end());
  for(pair<int,int>edge:edges){
    int a=edge.first,b=edge.second;
    if(rand()%2){
      // random order of two vertices
      swap(a,b);
    }
    printf("%d %d\n",perm[a],perm[b]);
  }
}
for(int i=2;i<=n;++i){
  printf("%d %d\n",rand(1,i-1),i);
}
```