Team Notebook

SUST$_Londoni_Sisters$

July 25, 2023

Contents

1 Data Structure and Graph

1.1 Arti Bridge Point

```
vl adj[MAX],point; vector<pll> span_edge,back_edge,bridge;
bool arti_point[MAX],vis[MAX]; ll d[MAX],low[MAX],timer=0;
void arti_BP(ll u=1, ll par=-1){
  low[u]=d[u]=++timer; vis[u]=1; ll child=0;
  for(auto now : adj[u]) { if(now==par)continue;
    if(vis[now]){
      low[u]=min(low[u],d[now]);
      if(d[now]<d[u]) back_edge.push_back({u,now});
    } else { span_edge.push_back({u,now}); arti_BP(now,u);
      low[u]=min(low[u],low[now]);
      if(d[u]<low[now]) bridge.push_back({u,now});
      if(d[u]<=low[now] && par!=-1) arti_point[u]=1;
      child++; }
    if(child>1 && par==-1) arti_point[u]=1; } }
```

1.2 BIT

```
int BIT[MAX],N;
void update(int x, int val) {
  while(x <= N) { BIT[x]+=val; x+=x&(-x);} }
int query(int x) { int sum=0;
  while(x>0) { sum+=BIT[x]; x-=x&(-x);} return sum; }
```

1.3 DSU on Index

```
ll par[MAX],idx[MAX], arr[MAX];
ll findPar(ll v){ if(par[v] == v) return v;
  return par[v] = findPar(par[v]); }
void union_sets(ll v, ll p){ //new par=p
  v=findPar(v); p=findPar(p); par[v]=p; }
void translate(ll x, ll y){ //x-->y
  ll id_x = idx[x]; ll id_y = idx[y];
  if(x == y || id_x == -1) return;
  if(id_y == -1){ //1st occu of y
    idx[y] = id_x; arr[id_x] = y; //update value
  } else { //merge x's with old y
      union_sets(id_x, id_y); }
  idx[x]=-1; }
void init(ll n){
  memset(idx, -1, sizeof idx);
  for(int i=0; i<MAX; i++) par[i]=i;
  for(int i=1; i<=n; i++){
      int id = idx[arr[i]];//index of current element
      if(id == -1) idx[arr[i]] = i; //1st occurence
      else union_sets(i, id); //par[i] = old occu
  } }
//main: translate(x,y); arr[findPar(idx)]
```

1.4 DSU on Tree

```
//preity
void add(int cur){ cnt[col[cur]]++; }
void remove_sack(int cur){ cnt[ col[cur]]--; }
void query(){
  int i,x,y,p; cin >> m;
  for(i=0;i<m;i++){
      cin>>x>>y;
      p=parent_find(x,y); if(p>0) q[p].push_back({i,y});
  } }
void dfs(int node, bool keep){
  int i,start,end_time; start=t; sack[t]=node; t++;
  int mx=0,bigchild=-1;
  for(i=0;i<graph[node].size();i++){
    if(sz[ graph[node][i] ] > mx){
      mx=sz[ graph[node][i] ]; bigchild=graph[node][i]; }}
  for(i=0;i<graph[node].size();i++){
    if(graph[node][i] == bigchild) continue;
    dfs(graph[node][i],false); }
  end_time=t;
  if(bigchild!=-1) dfs(bigchild, true);
  for(i=start;i<end_time;i++) add(sack[i]);
  for(i=0;i<q[node].size();i++)
    ans[ q[node][i].first] = cnt[ jeta jante chacchi ];
  if(keep) return;
  for(i=start;i<t;i++) remove_sack(sack[i]); }
//kawchar
int val[MAX],cnt[MAX],sz[MAX],ans[MAX]; bool big[MAX];
void processQuery(int v){ /*ans[v] = cnt[ ?? ]; */}
void getSize(int v, int p) {
  sz[v] = 1; for(auto u : adj[v]){
    if( u == p ) continue; getSize(u,v); sz[v]+=sz[u];}}
void add(int v, int p, int x){ cnt[ val[v] ] += x;
  for(auto u : adj[v])if(u!=p && !big[u]) add(u, v, x);}
void dfs(int v, int p, bool keep){
  int mx = -1, bigChild = -1; for(auto u : adj[v])
    if(u != p && sz[u] > mx) mx = sz[u], bigChild = u;
  for(auto u : adj[v])
    if(u != p && u != bigChild) dfs(u, v, 0);
  if(bigChild!=-1) dfs(bigChild,v,1), big[bigChild] = 1;
  add(v, p, 1); processQuery(v);
  if(bigChild != -1) big[bigChild] = 0;
  if(keep == 0){ add(v, p, -1); } }
void init(int root=1){getSize(root,-1);dfs(root,-1,0);}
```

1.5 Flat Tree

```
FT[t]=u;in[u]=t++; ... out[u]=t-1;(in[x],out[x]) child of x
v2: FT[t]=u;in[u]=t++; ... FT[t]=u;out[u]=t++
(in[x],out[x]) child of x & each node will appear twice.
```

1.6 Flow Max

```
/*Directed: cap[x][y]=c; cap[y][x]=0;
  Undirected: cap[x][y]=c; cap[y][x]=c;
  Multiple Edge: cap[x][y]+=c; */
int bfs(int s, int t){
  memset(par,-1,sizeof par);
  par[s] = s; queue<pair<int, int>> q; q.push({s, 1e9});
  while (!q.empty()) {
    int cur=q.front().first; int flow=q.front().second;
    q.pop(); if (cur == t) return flow;
    for (int v : g[cur]) {
      if (par[v] == -1 && capacity[cur][v]) {
          par[v] = cur;
          int new_flow = min(flow, capacity[cur][v]);
          q.push({v, new_flow}); } } } return 0; }
//O(V*E*E)
int maxflow(int s, int t) {
  int flow = 0,new_flow;
  while (1){
    new_flow = bfs(s, t); if(new_flow<=0) break;
    flow += new_flow; int cur = t;
    while (cur != s) {
      int prev=parent[cur]; capacity[prev][cur]-=new_flow;
      capacity[cur][prev] += new_flow; cur = prev;
    } } return flow; }
struct FlowEdge{
  int v,u; ll cap,flow=0LL;
  FlowEdge(int vv, int uu, ll c) {
    v=vv; u=uu; cap=c; /*Edge v to u*/ } };
//O(V*V*E)
struct Dinic{
  const ll flow_inf=LLONG_MAX; vector<int> level, par;
  vector<FlowEdge> edges; vector< vector<int> > adj;
  int n,m=0,s,t; queue<int> q;
  Dinic(int nn, int ss, int tt){
    n=nn+5; s=ss; t=tt;
    adj.resize(n); par.resize(n); level.resize(n); }
  void add_edge(int v, int u, ll cap){
    edges.emplace_back(v,u,cap);
    edges.emplace_back(u,v,0LL);//Ulto edge
    adj[v].push_back(m); adj[u].push_back(m+1); m+=2; }
  bool bfs(){
    while(!q.empty()){
      int v=q.front(); q.pop();
      for(int id : adj[v]) {
        if(level[edges[id].u]!=-1 || edges[id].cap - edges[id
            ].flow < 1) continue;
        level[edges[id].u]=level[v]+1;q.push(edges[id].u);
      } } return level[t]!=-1; }
  ll dfs(int v, ll pushed) {
    if(pushed == 0LL) return 0LL; if(v == t) return pushed;
    for(int& cid=par[v];cid<(int)adj[v].size();cid++){
      int id = adj[v][cid]; int u = edges[id].u;
      if(level[v]+1 != level[u] || edges[id].cap - edges[id].
          flow < 1) continue;
      ll tr=dfs(u,min(pushed,edges[id].cap-edges[id].flow));
      if(tr == 0) continue;
      edges[id].flow += tr; edges[id ^ 1].flow -= tr;
      return tr; } return 0LL; }
  ll maxFlow() {
    ll f = 0LL,pushed;
    while(1) {
      fill(level.begin(), level.end(), -1);
      level[s] = 0; q.push(s); if(!bfs()) break;
      fill(par.begin(), par.end(), 0);
      while(pushed = dfs(s, flow_inf)) f += pushed;
    } return f; }  };
```

1.7 Flow Maximum Bipartite Matching Kuhn

```
//bipartite graph maximum pair matching kuhn's O(n*m)
vector<int>g[sz]; int t,n,m,k,timer,from[sz]; bool used[sz];
bool go(int u){
  used[u] = 1;
  for(auto x:g[u]){
    if(used[x]) continue;
    if(!from[x] or go(from[x])){ from[x] = u; return 1; }
  } return 0; }
main(){ int ans = 0;
  for(int i = 1;i<=n;i++){
      memset(used,0,sizeof used); go(i); }
  for(int i = 1;i<=n;i++){
      if(from[i] == 0) continue;
      cout<<"match "<<i<<" "<<from[i]<<endl; } }
```

1.8 Flow Min Cost Max Flow

```
int cost[MAX][MAX],capacity[MAX][MAX],d[MAX],p[MAX];
bool inq[MAX];
void add_edge(int u, int v, int cap, int c){
  //u--->v : directed , capacity cap, cost c
  adj[u].push_back(v); adj[v].push_back(u);
  cost[u][v]=c; cost[v][u]=-c;
  capacity[u][v]=cap; capacity[v][u]=0; }
void shortest_paths(int n, int s) {
  for(int i=1;i<=n;i++)d[i]=inf, inq[i]=0, p[i]=-1;
  queue<int> q; d[s] = 0; q.push(s); inq[s] = true;
  while(!q.empty()) {
    int u = q.front(); q.pop(); inq[u] = false;
    for(int v : adj[u]) {
      if(capacity[u][v]>0 && d[v]>d[u]+cost[u][v]){
        d[v] = d[u] + cost[u][v]; p[v] = u;
        if(!inq[v]) { inq[v] = 1; q.push(v);}}}}}
//O(V*V*E*E)
int min_cost_flow(int N,int s,int t,int maxFlow=inf){
  int flow = 0,minCost = 0,cur,f;
  while(flow < maxFlow) {
    shortest_paths(N, s); if(d[t] == inf) break;
    f = maxFlow - flow; cur = t;
    while(cur != s) {
    f = min(f, capacity[p[cur]][cur]); cur = p[cur];}
    flow += f; minCost += f * d[t]; cur = t;
    while (cur != s) {
      capacity[p[cur]][cur] -= f;
      capacity[cur][p[cur]] += f; cur = p[cur];} }
  return minCost; }
main(){add_edge(u,v,cap,c);cout<<min_cost_flow(n,1,n);}
```

1.9 Heavy-Light Decomposition

```
// z[i]=longest common prefix between s[1...n], s[i...n]
int sz[MX],par[MX],dep[MX],val[MX],id[MX],tp[MX],ct;
int tree[3*MX], arr[MX], N, lazy[3*MX];
// here seg tree //
void dfs_sz(int u, int p) {
  sz[u] = 1; par[u] = p; for (int v : adj[u]) {
    if (v == p) continue; dep[v] = dep[u] + 1;
    par[v] = u; dfs_sz(v, u); sz[u] += sz[v]; } }
void dfs_hld(int u, int p, int top) {
  id[u] = ct++; tp[u] = top;
  arr[ id[u] ] = val[u]; int h_chi = -1, h_sz = -1;
  for (int v : adj[u]) { if (v == p) continue;
    if (sz[v] > h_sz) { h_sz = sz[v]; h_chi = v; } }
  if (h_chi == -1) return; dfs_hld(h_chi, u, top);
  for(int v : adj[u]) {
    if(v==p || v==h_chi) continue; dfs_hld(v,u, v);}}
int pathQuery(int x, int y) { int ret = INT_MIN;
  while (tp[x] != tp[y]) {
    if (dep[tp[x]] < dep[tp[y]]) swap(x, y);
    ret = merge(ret, query(id[tp[x]], id[x]));
    x = par[tp[x]];
  } if (dep[x] > dep[y]) swap(x, y);
  ret = merge(ret, query(id[x], id[y])); return ret; }
void updateNode(int u, int val) {update(id[u], val); }
void updateSubTree(int u, int val) {
  update(id[u], id[u]+sz[u]-1, val);}
void init_hld(int n) {
  ct = 0; dep[1] = 0; dfs_sz(1, 1); dfs_hld(1, 1, 1);
  buildSegTree(ct); /*arr[0 ... ct-1]*/ }
```

1.10 LCA

```
#define maxN 17 #define MAX 100001
vector<int> adj[MAX]; int lvl[MAX],LCA[MAX][maxN+1];
void DFS(int v, int p) {
  LCA[v][0]=p;
  for(auto x : adj[v]){
      if(x==p) continue; lvl[x]=lvl[v]+1; DFS(x, v); } }
void init(int N) {
  memset(LCA, -1, sizeof LCA);
  lvl[1]=0; DFS(1, -1);
  for(int j=1; j<=maxN; j++) {
    for(int i=1; i<=N; i++) {
      if(~LCA[i][j-1]) {
       int p = LCA[i][j-1]; LCA[i][j]=LCA[p][j-1]; } } }}
int find_lca(int a, int b) {
  if(lvl[a]>lvl[b]) swap(a,b);
  int d=lvl[b]-lvl[a];
  while(d>0) { int j = log2(d);
    b=LCA[b][j]; d-=(1<<j); }
  if(a==b) return a;
  for(int j=maxN; ~j; j--) {
    if(~LCA[a][j] && (LCA[a][j]!=LCA[b][j])) {
      a=LCA[a][j]; b=LCA[b][j]; } } return LCA[a][0]; }
int RootedLCA(int r, int u, int v) {
  int ur=find_lca(u,r); int rv=find_lca(r,v);
  int uv=find_lca(u,v); if(ur==rv) return uv;
  if(rv==uv) return ur; if(uv==ur) return rv; }
```

1.11 MOs algo

```
//count unique
#define B_SIZE 550 ///sqrt(MAX)
struct query{ int l,r,id; };
query Q[MAX]; int ar[MAX], ans[MAX], fre[1000006],cnt=0;
bool cmp(query &a, query &b){
  if(a.l/B_SIZE != b.l/B_SIZE) return a.l<b.l;
  return a.l/B_SIZE%2?a.r>b.r:a.r<b.r; }
void add(int pos) {
  fre[ar[pos]]++; if(fre[ar[pos]]==1) cnt++; }
void sub(int pos) {
  fre[ar[pos]]--; if(!fre[ar[pos]]) cnt--; }
int get_ans() { return cnt; }
void MO(int q) {
  sort(Q, Q+q, cmp); int cur_l=0, cur_r=-1;
  for(int i=0; i<q; i++) {
    int L=Q[i].l; int R=Q[i].r;
    while(cur_l>L) add(--cur_l);
    while(cur_r<R) add(++cur_r);
    while(cur_l<L) sub(cur_l++);
    while(cur_r>R) sub(cur_r--);
    ans[Q[i].id]=get_ans(); } }
//most frequent value
void add(int pos){
  if(cnt[ar[pos]] != 0) fre[cnt[ar[pos]]]--;
  cnt[ar[pos]]++; fre[cnt[ar[pos]]]++;
  max_cnt=max(max_cnt, cnt[ar[pos]]);}
void sub(int pos){
  if(cnt[ar[pos]] != 0) fre[cnt[ar[pos]]]--;
  if(fre[max_cnt]==0) max_cnt--;
  if(max_cnt<1) max_cnt=1; cnt[ar[pos]]--;
  if(cnt[ar[pos]] != 0) fre[cnt[ar[pos]]]++; }
//cnt sub-array of xor K
void add(int pos){cnt+=fre[ar[pos]^k];fre[ar[pos]]++;}
void sub(int pos){fre[ar[pos]]--;cnt-=fre[ar[pos]^k];}
main(){ ar[i]^=ar[i-1]; Q[i].r=R;
Q[i].l=L-1; //cum[1...x]^cum[1...L-1]=cum[1...L]
```

1.12 MST

```
struct Edge{
  int u,v,w; Edge(){}
  Edge(int x,int y,int z){ u=x;v=y;w=z; } };
bool operator<(Edge a,Edge b){ return a.w<b.w; }
vector<Edge>E; int par[MAX],Size[MAX];
int find_par(int x){
  return par[x]==x?x:par[x]=find_par(par[x]); }
void union_sets(int a, int b){
  a=find_set(a); b=find_set(b);
  if(a!=b) { if(Size[a]<Size[b]) swap(a,b);
    parent[b]=a; Size[a]+=Size[b]; } }
int MST(int n) {
  int i,cnt=0,u,v,total=0;
  for(i=1; i<=n; i++) par[i]=i;
  sort(E.begin(),E.end());
  for(auto now : E){
    u=find_par(now.u); v=find_par(now.v);
    if(u==v) continue; union_sets(u,v);
    total+=now.w; cnt++; }
  if(cnt!=n-1) total=-1; return total; }
```

1.13 SCC

```
vector<int>g[siz],gin[siz],comps[siz];
int t,n,m,cur,h,k,sz,mark[siz],a[siz]; bool bad,used[siz];
vector<int>component; stack<int>order;
void dfs1(int v) {
  used[v] = true;
  for(auto u : g[v]){ if(!used[u]) dfs1(u); }
  order.push(v); }
void dfs2(int v) {
  used[v] = 1; component.push_back(v);
  for(auto u : gin[v]){ if(!used[u]) dfs2(u); } }
int32_t main(){ cin>>n>>m;
  for(int i = 1;i<=m;i++){
    int x,y; cin>>x>>y;
    g[x].pb(y); gin[y].pb(x); }
  for(int i=1; i<=n; i++) if(!used[i]) dfs1(i);
  memset(used,0,sizeof used);
  while(order.size()){
    int v = order.top(); order.pop();
    if(!used[v]){
        component.clear(); dfs2(v);
        //component will have all node of this strongly
             connected component
    } } }
```

1.14 SPFA

```
//SPFA is a improvement of the Bellman-Ford algorithm
vector< pair<int,int> > adj[MAX]; int d[MAX], cnt[MAX];
bool inqueue[MAX];
bool spfa(int n, int s){
  for(int i=1; i<=n; i++){
    d[i]=inf; inqueue[i]=0; cnt[i]=0; }
  queue<int> q; q.push(s); inqueue[s] = 1; d[s] = 0;
  while(!q.empty()){
    int u = q.front(); q.pop(); inqueue[u] = 0;
    for(auto edge : adj[u]){
      int v = edge.first; int w = edge.second;
      if(d[u] + w < d[v]){
        d[v] = d[u] + w;
        if(!inqueue[v]){
          q.push(v); inqueue[v] = 1; cnt[v]++;
          if(cnt[v] > n) return 0;/*negative cycle*/
        } } } } return 1; }
```

1.15 Segment Tree 2D

```
struct info {
  int sum,mn,mx;
  info(){ mn=INT_MAX/100; mx=INT_MIN/100; sum=0; }
  info(int x, int y, int z){ mn=x; mx=y; sum=z; }
  info(int x) { mn=mx=sum=x; }
}tree[3*MAXN][3*MAXM];
info leaf(int i, int j) { return info(arr[i][j]); }
info Set(ll x){ return info(x); }
info outOfRange(){ return info(); }
info merge(info x, info y){
  info ans; ans.mn=min(x.mn, y.mn);
  ans.mx=max(x.mx, y.mx); ans.sum=x.sum+y.sum;
  return ans; }
void build_y(int vx,int lx,int rx,int vy,int ly,int ry){
  if(ly == ry){
    if(lx == rx) tree[vx][vy] = leaf(lx,ly);
  else tree[vx][vy]=merge(tree[vx*2][vy],tree[vx*2+1][vy]);
  } else {
    int my = (ly + ry) / 2;
    build_y(vx, lx, rx, vy*2, ly, my);
    build_y(vx, lx, rx, vy*2+1, my+1, ry);
    tree[vx][vy]=merge(tree[vx][vy*2],tree[vx][vy*2+1]); } }
void build_x(int vx, int lx, int rx){
  if(lx != rx) {
    int mx = (lx + rx) / 2;
    build_x(vx*2,lx,mx); build_x(vx*2+1,mx+1,rx); }
  build_y(vx, lx, rx, 1, 0, M-1); }
info query_y(int vx,int vy,int tly,int try_,int ly,int ry){
  if(ly > ry) return outOfRange();
  if(ly == tly && try_ == ry) return tree[vx][vy];
  int tmy = (tly + try_) / 2;
  auto x=query_y(vx,vy*2,tly,tmy,ly,min(ry, tmy));
  auto y=query_y(vx,vy*2+1,tmy+1,try_,max(ly,tmy+1),ry);
  return merge(x,y); }
info query_x(int vx,int tlx,int trx,int lx,int rx,int ly,int
      ry){
```

```
  if(lx > rx) return outOfRange();
  if(lx == tlx && trx == rx)
      return query_y(vx, 1, 0, M-1, ly, ry);
  int tmx = (tlx + trx) / 2;
  auto x=query_x(vx*2,tlx,tmx,lx,min(rx, tmx),ly,ry);
  auto y=query_x(vx*2+1,tmx+1,trx, max(lx,tmx+1),rx,ly,ry);
  return merge(x,y); }
void update_y(int vx, int lx, int rx, int vy, int ly, int ry
    , int x, int y, int val){
  if(ly == ry){
    if(lx == rx) tree[vx][vy] = val;
   else tree[vx][vy]=merge(tree[vx*2][vy],tree[vx*2+1][vy]);
  } else {
      int my = (ly + ry) / 2;
      if(y<=my) update_y(vx,lx,rx,vy*2,ly,my,x,y,val);
      else update_y(vx,lx,rx,vy*2+1,my+1,ry,x,y,val);
      tree[vx][vy]=merge(tree[vx][vy*2],tree[vx][vy*2+1]);}}
void update_x(int vx,int lx,int rx,int x,int y,int val){
  if(lx != rx){
    int mx = (lx + rx) / 2;
    if(x<=mx) update_x(vx*2, lx, mx, x, y, val);
    else update_x(vx*2+1, mx+1, rx, x, y, val); }
  update_y(vx, lx, rx, 1, 0, M-1, x, y, val); }
void build(int n, int m){ N=n;M=m; build_x(1,0,N-1);}
void update(int i, int j, int val){ //arr[i][j]=val
  update_x(1,0,N-1,i,j,val); }
info query(int x1, int y1, int x2, int y2){
  return query_x(1, 0, N-1, x1, x2, y1, y2); }
```

1.16 Segment Tree Lazy

```
ll arr[MAX]; int N;
struct info{
  ll lazy,val; info() {lazy=0LL; val=0LL; }
}tree[3*MAX];
void propagate(int node, int l, int r) {
  tree[node].val+=tree[node].lazy*(r-l+1);
  if(l!=r){
    tree[2*node].lazy+=tree[node].lazy;
    tree[2*node+1].lazy+=tree[node].lazy;
  } tree[node].lazy=0; }
ll merge(ll x, ll y) {return x+y;}
void build(int node, int l,int r) {
  if(l==r){
    tree[node].lazy=0;tree[node].val=arr[l]; return;}
  int mid=(l+r)/2;
  build(node*2,l,mid);build(node*2+1,mid+1,r);
  tree[node].lazy=0LL;
  tree[node].val=merge(tree[node*2].val,tree[2*node+1].val)
      ;}
ll query(int node,int l,int r,int i,int j){
  propagate(node,l,r); if(i>r || j<l) return 0LL;
  if(l>=i && r<=j) return tree[node].val;
  int mid=(l+r)/2;
  auto x=query(node*2,l,mid,i,j);
  auto y=query(node*2+1,mid+1,r,i,j);return merge(x,y);}
int searchQuery(int node, int l, int r, ll sum) {
  propagate(node,l,r); int mid=(l+r)/2;
  if(l==r) return l; ll x=tree[2*node].val;
  if(sum<=x) return searchQuery(2*node, l, mid, sum);
  else return searchQuery(2*node+1, mid+1, r, sum-x);}
void update(int node,int l,int r,int i,int j,ll val){
  propagate(node,l,r); if(i>r || j<l) return;
  if(l>=i && r<=j) {
    tree[node].lazy+=val;propagate(node,l,r);return;}
  int mid=(l+r)/2; update(node*2,l,mid,i,j,val);
  update(node*2+1,mid+1,r,i,j,val);
  tree[node].val=merge(tree[node*2].val,tree[node*2+1].val)
      ;}
void build(int n){ N=n; build(1,0,N-1);}
void update(int i, int j, ll val){
  update(1,0,N-1,i,j,val); }
ll query(int x, int y){ return query(1,0,N-1,x,y); }
```

1.17 Segment Tree Persistent

```
#define LOG 17
struct info{ int L,R,val; };
info tree[LOG*MAX];
int root[MAX],arr[MAX],N,NEXT_FREE_INDEX=1,v_id=1;
int merge(int x, int y){ return x+y; }
void build(int node, int l, int r) {
  if(l==r){ tree[node].val=arr[l]; return; }
  int mid=(l+r)/2;
  int x=NEXT_FREE_INDEX++; int y=NEXT_FREE_INDEX++;
  tree[node].L=x; tree[node].R=y;
  build(tree[node].L, l, mid);
  build(tree[node].R, mid+1, r);
  tree[node].val=merge(tree[x].val, tree[y].val); }
int update(int node, int l, int r, int pos, int val){
  if(pos<l || r<pos) return node;
  if(l==r) {
      int x=NEXT_FREE_INDEX++;
      tree[x].val=tree[node].val+val; return x; }
  int mid=(l+r)/2; int z=NEXT_FREE_INDEX++;
  int x=update(tree[node].L, l, mid, pos, val);
  int y=update(tree[node].R, mid+1, r, pos, val);
  tree[z].val=merge(tree[x].val, tree[y].val);
  tree[z].L=x; tree[z].R=y; return z; }
int query(int node, int l, int r, int i, int j){
  if(j<l || r<i) return 0;
  if(l>=i && r<=j) return tree[node].val;
  int mid=(l+r)/2; int x=query(tree[node].L, l, mid, i, j);
  int y=query(tree[node].R, mid+1, r, i, j);
  return merge(x,y); }
int k_th(int cur, int pre, int l, int r, int k){
  if(l==r) return l; int mid=(l+r)/2;
  int l_val=tree[tree[cur].L].val-tree[tree[pre].L].val;
  if(k<=l_val)
    return k_th(tree[cur].L, tree[pre].L, l, mid, k);
  else return k_th(tree[cur].R,tree[pre].R,mid+1,r,k-l_val);
}
void build(int n){
  N=n; NEXT_FREE_INDEX=1; v_id=1;
  root[0]=NEXT_FREE_INDEX++; build(root[0], 0, N-1); }
void update(int version, int pos, int x){
  root[v_id++]=update(root[version], 0, N-1, pos, x); }
int query(int version, int i, int j){
  return query(root[version], 0, N-1, i, j); }
```

1.18 Sparse Table

```
#define LVL 20 //ceil(log_2(MAXN))+1
int arr[MAX], dst[MAX][LVL];
int merge(int x, int y) { return __gcd(x,y); }
//0 based
void build(int n){
  for(int i=0; i<n; ++i) dst[i][0] = arr[i];
  for(int k=1; k<LVL; ++k)
    for(int i=0; (i+(1<<k)-1)<n; ++i)
      dst[i][k]=merge(dst[i][k-1], dst[i+(1<<(k-1))][k-1]);}
int query(int l, int r){
  int k=31-__builtin_clz(r-l+1);
  return merge(dst[l][k], dst[r-(1<<k)+1][k]); }
//2D Sparse Table: O(n^2 (logn)^2
int A[MAXN][MAXN]; int M[MAXN][MAXN][LOGN][LOGN];
void Build2DSparse(int N){ //1 based
  for(int i = 1; i <= N; i++){
    for(int j = 1; j <= N; j++)
        M[i][j][0][0] = A[i][j];
    for(int q = 1; (1<<q) <= N; q++){
      int add = 1<<(q-1);
      for(int j=1; j+add <= N; j++){
        M[i][j][0][q] = merge(M[i][j][0][q-1], M[i][j+add
            ][0][q-1]); } } }
    for(int p=1; (1<<p)<=N; p++){
```

```
    int add = 1<<(p-1);
    for(int i=1; i+add <= N; i++){
      for(int q=0; (1<<q) <= N; q++){
        for(int j=1; j<= N; j++)
          M[i][j][p][q] = merge(M[i][j][p-1][q], M[i+add][j
               ][p-1][q]); } } } }
//merge() of all A[i][j], where x1<=i<=x2 and y1<=j<=y2
int Query(int x1,int y1,int x2,int y2){
  int kX = log2(x2-x1+1); int kY = log2(y2-y1+1);
  int addX = 1<<kX; int addY = 1<<kY;
  int ret1 = merge(M[x1][y1][kX][kY], M[x1][y2-addY+1][kX][
       kY]);
  int ret2 = merge(M[x2-addX+1][y1][kX][kY], M[x2-addX+1][y2
       -addY+1][kX][kY]);
  return max(ret1, ret2); }
```

1.19 Sum of Distance

```
sum of the distances from the node to all other nodes:
in[u]: from u to each node in subtree rooted at u
out[u]: excluding the subtree rooted at u
in[u] = sum of in[child] + size[child]
out[u] = contribution of par (out[par]) +
  contribution of edge u <--> par (n - size[par] + 1) +
 contribution of each siblings (in[sib] + 2 * size[sib])
```

1.20 Treap

```
struct info{
  int l, r, prt, sz, val, ans, rev; };
//1-Based index
int N, treap; info tree[MAX];
void init(){ N = 0; treap = -1; srand(time(0));}
int get_node(int val){
  tree[N].val = val; tree[N].ans = val;
  tree[N].rev = 0; tree[N].sz = 1;
  tree[N].l = -1; tree[N].r = -1;
  tree[N].prt = rand() % 1000000000;
  return N++; }
void propagate(int t){
  //push down: T-->L,R
  if(t == -1 || !tree[t].rev) return;
  int l = tree[t].l, r = tree[t].r;
  if(~l) tree[l].rev ^= 1; if(~r) tree[r].rev ^= 1;
  swap(tree[t].l, tree[t].r); tree[t].rev = 0; }
void calibrate(int t){
  //push up: L,R-->Par
  propagate(t); //need?
  if(t == -1) return; tree[t].sz = 1;
  int l = tree[t].l, r = tree[t].r;
  //update values
  tree[t].ans = tree[t].val;
  if(~l){ //Combine Lft,Par
    tree[t].sz += tree[l].sz;
    tree[t].ans = tree[l].ans + tree[t].ans; }
  if(~r){ //Combine Par,Rgt
    tree[t].sz += tree[r].sz;
    tree[t].ans = tree[t].ans + tree[r].ans; } }
pair<int,int> split(int t, int key){
  propagate(t); if(t == -1) return {-1, -1};
  if(key <= 0) return {-1, t}; pair<int,int> ret;
  int l = tree[t].l, r = tree[t].r;
  if(l == -1 || tree[l].sz < key){
    ret = split(r, key - (~l ? tree[l].sz : 0) - 1);
    tree[t].r = ret.first; ret.first = t;
  }else{
    ret = split(l, key); tree[t].l = ret.second;
    ret.second = t; }
  calibrate(t); return ret; }
//only for sorted list
pair<int,int> split_by_value(int t, int val){
  propagate(t); if(t == -1) return {-1, -1};
  pair<int,int> ret; int l = tree[t].l, r = tree[t].r;
  if(tree[t].val <= val){
    ret = split_by_value(r, val); tree[t].r = ret.first;
    ret.first = t;
  }else{
    ret = split_by_value(l, val); tree[t].l = ret.second;
    ret.second = t; }
  calibrate(t); return ret; }
//v[0] = [1, l-1], v[1] = [l, r], v[2] = [r+1, N]
vector<int> parts(int l, int r){
  vector<int> v(3); auto cur = split(treap, l - 1);
  v[0] = cur.first; cur = split(cur.second, r - l + 1);
  v[1] = cur.first; v[2] = cur.second; return v; }
int merge(int a,int b){
  propagate(a); propagate(b);
  if(min(a, b) == -1) return max(a, b);
  if(tree[a].prt > tree[b].prt){
    tree[a].r = merge(tree[a].r, b);
    calibrate(a); return a;
  }else{
    tree[b].l = merge(a, tree[b].l);
    calibrate(b); return b; } }
int merge(int a, int b, int c){
  return merge(merge(a, b), c); }
int query(int l, int r){
  auto p = parts(l, r); int ret = tree[ p[1] ].ans;
  treap = merge(p[0], p[1], p[2]); return ret ; }
void erase(int l, int r){
  auto p = parts(l, r); treap = merge(p[0], p[2]); }
void update(int i, int val){
  auto p = parts(i, i); tree[ p[1] ].val = val;
  calibrate(p[1]); treap = merge(p[0], p[1], p[2]); }
void reverse(int l, int r){
  auto p = parts(l, r); tree[ p[1] ].rev ^= 1;
  treap = merge(p[0], p[1], p[2]); }
void cyclicShift(int l,int r,int cnt=1,int left_shift=0){
  auto p = parts(l, r);
  if(left_shift) cnt = r - l + 1 - cnt;
  auto cur = split(p[1], cnt);
  p[1] = merge(cur.second, cur.first);
  treap = merge(p[0], p[1], p[2]); }
void insert(int i, int val){
  int x = get_node(val); auto p = parts(i, i);
  p[1] = merge(x, p[1]); treap = merge(p[0], p[1], p[2]); }
void push_back(int val){
  treap = merge(treap, get_node(val)); }
void print(int t = treap, int last = 1){
  propagate(t); if(t == -1) return;
  print(tree[t].l, 0); cout << tree[t].val << ' ';
  print(tree[t].r, 0); if(last) cout << "\n"; }
//for sorted list
int lower_bound(int val){
  //[<val][val][>val]
  auto p = split_by_value(treap, val);
  auto p2 = split_by_value(p.first, val-1);
  int pos = 1; if(~p2.first) pos += tree[p2.first].sz;
  treap = merge(p2.first, p2.second, p.second);
  return pos; }
int find_pos(int val){
  //[<val][val][>val]
  auto p = split_by_value(treap, val);
  auto p2 = split_by_value(p.first, val-1);
  int pos = 1; if(p2.second == -1) pos = -1;
  else if(~p2.first) pos += tree[p2.first].sz;
  treap = merge(p2.first, p2.second, p.second);
  return pos; }
```

1.21 Warshall

```
void warshall(ll n){
  for(k=1; k<=n; k++)
    for(i=1; i<=n; i++)
      for(j=1; j<=n; j++){
        ll x=v[k].second;
        dp[k][i][j]=min(dp[k-1][i][j], dp[k-1][i][x]+dp[k-1][
            x][j]); dp[k][j][i]=dp[k][i][j]; } }
```

1.22 WavLet Tree

```
#define pii pair<int,int>
#define x first
#define y second
int n;
struct wavelet_tree {
  static const int unfound = INT_MIN;
  static const int maxn = 1e5 + 5;
  static const int mlog = 20;
  typedef int Int; typedef long long Long;
  Int s[maxn], tree[mlog][maxn];
  int L[mlog][maxn];
  Long ls[mlog][maxn], sl;
  Int & operator[](int x) {return tree[0][x];}
  void build(int l = 1, int r = n, int d = 0) {
    if (l == r)return;
    int m=(l + r)>>1, cnt=0, lc=l, rc=m+1, ec=0;
    for(int i=l; i<=r; i++) if(tree[d][i]<s[m])cnt++;
    for (int i = l; i <= r; i++) {
      if ( (tree[d][i] < s[m]) || (tree[d][i] == s[m] && ec <
           (m - l + 1 - cnt) ) ) {
        tree[d + 1][lc++] = tree[d][i];
        ls[d + 1][i] = ls[d+1][i-1]+tree[d][i];
        if (tree[d][i] == s[m]) ec++;
      } else {
        tree[d + 1][rc++] = tree[d][i];
        ls[d + 1][i] = ls[d + 1][i - 1];
    } L[d][i] = L[d][l - 1] + lc - l;
    } build(l, m, d + 1); build(m + 1, r, d + 1);
  }
  void init(Int *arr, int n) {
    for(int i=1; i<=n; i++) tree[0][i] = arr[i]; init(n);
  }
  void init(int n) {
    for(int i=1; i<=n; i++)s[i]=tree[0][i], ls[0][i]=ls[0][i
        -1]+s[i];
    sort(s + 1, s + 1 + n); build();
  }
  Long sum(pii a,int d=0){return ls[d][a.y]-ls[d][a.x-1];}
  int cn(pii a, int d) {return L[d][a.y] - L[d][a.x - 1];}
  pii left(pii a, int d, int l) {return {l + cn({l, a.x-1},
      d), l-1+cn({l, a.y}, d)};}
  pii right(pii a, int d, int r) {return {a.x + cn({a.x, r},
       d), a.y + cn({a.y+1, r}, d)};}
  Int kth(int x,int y,int k,int l=1, int r=n, int d=0) {
  if (y - x + 1 < k || x > y)return unfound;
    if (l == r) {
      sl += tree[d][l]; return tree[d][l];
    } int cnt = cn({x, y}, d), m = (l + r) >> 1, nx, ny;
    if (cnt >= k) {
      tie(nx, ny) = left({x, y}, d, l);
      return kth(nx, ny, k, l, m, d + 1);
    } else {
      sl += sum({x, y}, d+1); tie(nx, ny)=right({x,y},d,r);
      return kth(nx, ny, k - cnt, m + 1, r, d + 1);
    }
  }
  int leq(int x, int y, Int k, int l=1, int r=n, int d=0){
    if (x > y)return 0;
    if (l == r) {
      if (l > y || l < x)return 0; // is it important?
      sl += tree[d][l] * (tree[d][l] <= k);
      return tree[d][l] <= k;
    } int cnt = cn({x, y}, d), m = (l + r) >> 1, nx, ny;
    if (s[m] <= k) {
      sl += sum({x,y}, d+1); tie(nx,ny) = right({x,y},d,r);
      return cnt + leq(nx, ny, k, m + 1, r, d + 1);
    } else {
      tie(nx, ny) = left({x, y}, d, l);
      return leq(nx, ny, k, l, m, d + 1);
    }
  }
  Int rmin(int x, int y, int l=1, int r=n, int d=0) {return
       kth(x, y, 1, l, r, d);}
  Int rmax(int x, int y, int l=1, int r=n, int d=0) {return
       kth(x, y, y-x+1, l, r, d);}
  Int floor(int x, int y, Int k, int l = 1, int r = n, int d
        = 0) {
    if (x > y)return INT_MIN;
    if (l == r) {
      if (l > y || l < x)return INT_MIN;
      return (tree[d][l] <= k ? tree[d][l] : INT_MIN);
    } Int ans = INT_MIN; int m = (l + r) >> 1, nx, ny;
    if (s[m] <= k) {
      tie(nx, ny) = right({x, y}, d, r);
      ans = max(ans, floor(nx, ny, k, m + 1, r, d + 1));
      if (ans == INT_MIN) {
        tie(nx, ny) = left({x, y}, d, l);
        auto an = rmax(nx, ny, l, m, d + 1);
        if (an != unfound)ans = max(ans, an);
      }
    } else {
        tie(nx, ny) = left({x, y}, d, l);
        ans = max(ans, floor(nx, ny, k, l, m, d + 1));
    } return ans;
  }
  Int ceil(int x, int y, Int k, int l=1, int r=n, int d=0){
    if (l == r) {
      if (l > y || l < x)return INT_MAX;
      return (tree[d][l] >= k ? tree[d][l] : INT_MAX);
    } Int ans = INT_MAX; int m = (l + r) >> 1, nx, ny;
    if (s[m] >= k) {
      tie(nx, ny) = left({x, y}, d, l);
      ans = min(ans, ceil(nx, ny, k, l, m, d + 1));
      if (ans == INT_MAX) {
        tie(nx, ny) = right({x, y}, d, r);
        auto an = rmin(nx, ny, m + 1, r, d + 1);
        if (an != unfound)ans = min(ans, an);
      }
    } else {
        tie(nx, ny) = right({x, y}, d, r);
        ans = min(ans, ceil(nx, ny, k, m + 1, r, d + 1));
    } return ans;
  }
} wvt;
```

1.23 Xor Basis

```
struct XorBasis{
  vector<ll> basis; ll N=0,tmp=0;
  void add(ll x){
    N++; tmp|=x;
    for(auto &i : basis) x=min(x,x^i);
    if(!x) return;
    for(auto &i : basis) if((i^x)<i) i^=x;
    basis.push_back(x); sort(basis.begin(),basis.end());}
  ll size(){ return (ll)basis.size(); }
  void clear(){ N=0;tmp=0; basis.clear(); }
  bool possible(ll x){
    for(auto &i: basis) x=min(x, x^i); return !x; }
  ll maxxor(ll x=0){
    for(auto &i: basis) x=max(x, x^i); return x; }
  ll minxor(ll x=0){
    for(auto &i: basis) x=min(x, x^i); return x; }
  ll cntxor(ll x){
    if(!possible(x)) return 0; return (1LL<<(N-size()));}
  ll sumOfAll(){
    ll ans=tmp*(1LL<<(N-1)); return ans; }
  ll kth(ll k){
    ll sz=size(); if(k > (1LL<<sz)) return -1;
    k--; ll ans=0;
    for(ll i=0; i<sz; i++) if(k>>i & 1) ans^=basis[i];
    return ans; } }xb;
////V2
```

```
int basis[LOG_K], sz;
void insertVector(int mask) {
  for (int i = LOG_K - 1; i >= 0; i--) {
    if ((mask & 1LL << i) == 0) continue;
    if (!basis[i]) { basis[i] = mask; sz++; return;}
    mask ^= basis[i]; } }
int query(int k) { int ans = 0;
  for(int i = LOG_K - 1; i >= 0; i--){
    if(k & (1LL << i)){
      if(!basis[i]) return -1;
       ans++, k ^= basis[i]; } } return ans; }
```

1.24 maximum$_bipartite_matching_blossom$

```
//O(N^3)
const int N = 2e3 + 9;
mt19937 rnd(chrono::steady_clock::now().time_since_epoch().
     count());
struct Blossom {
  int vis[N],par[N],orig[N],match[N],aux[N],t,n;
  bool ad[N];
  vector<int> g[N]; queue<int> Q;
  Blossom() {}
  Blossom(int _n){
    n = _n; t = 0;
    for(int i = 0; i <= _n; ++i) {
      g[i].clear();
      match[i] = aux[i] = par[i] = vis[i] = aux[i] = ad[i] =
           orig[i] = 0;
    }
  }
  void add_edge(int u, int v) {
    g[u].push_back(v);
    g[v].push_back(u);
  }
  void augment(int u, int v) {
    int pv = v, nv;
    do {
      pv = par[v];
      nv = match[pv];
      match[v] = pv;
      match[pv] = v;
      v = nv;
    } while (u != pv);
  }
  int lca(int v, int w) {
    ++t;
    while (true) {
      if (v) {
        if (aux[v] == t) return v;
        aux[v] = t;
        v = orig[par[match[v]]];
      }
      swap(v, w);
    }
  }
  void blossom(int v, int w, int a) {
    while (orig[v] != a) {
      par[v] = w;
      w = match[v];
      ad[v] = true;
      if (vis[w] == 1) Q.push(w), vis[w] = 0;
      orig[v] = orig[w] = a;
      v = par[w];
    }
  }
  bool bfs(int u) {
    fill(vis + 1, vis + n + 1, -1);
    iota(orig + 1, orig + n + 1, 1);
    Q = queue<int> ();
    Q.push(u);
    vis[u] = 0;
    while (!Q.empty()) {
      int v = Q.front();
      Q.pop();
      ad[v] = true;
      for (int x : g[v]) {
        if (vis[x] == -1) {
          par[x] = v;
          vis[x] = 1;
          if (!match[x]) return augment(u, x), true;
          Q.push(match[x]);
          vis[match[x]] = 0;
        } else if (vis[x] == 0 && orig[v] != orig[x]) {
          int a = lca(orig[v], orig[x]);
          blossom(x, v, a);
          blossom(v, x, a);
        }
      }
    }
    return false;
  }
  int maximum_matching() {
    int ans = 0;
    vector<int> p(n - 1);
    iota(p.begin(), p.end(), 1);
    shuffle(p.begin(),p.end(),rnd);
    for(int i = 1; i <= n; i++)
     shuffle(g[i].begin(),g[i].end(),rnd);
    for (auto u : p) {
      if (!match[u]) {
        for(auto v : g[u]) {
          if (!match[v]) {
            match[u] = v, match[v] = u;
            ++ans; break;
          }
        }
      }
    }
    for(int i = 1; i <= n; ++i)
     if(!match[i] && bfs(i)) ++ans;
    return ans;
  }
} M;
void solve() {
  M = Blossom (n);
  M.add_edge(i, j);
  ans = M.maximum_matching();
  //i --> M.match[i]
}
```

2 Dynamin Programming

2.1 Aliens Trick

```
pair<ll,ll> solveWithPenalty(ll p){
  //add p with targets
  ll ans=0,cnt=0;
  //cal ans
  //remove p from targets
  return {ans, cnt}; }
ll Alien(ll l, ll r, ll need){
  ll ans=-1;
  while(l <= r) {
    ll mid = l+(r-l)/2;
    //solveWithPenalty() retrun {ansWithPenalty, cnt}
    pair<ll,ll> res = solveWithPenalty(mid);
    if(res.second >= need)
      ans = res.first - mid * need, l = mid + 1;
    else r = mid - 1;
  }  return ans; }
```

2.2 CHT

```
struct Line {
  //mx+c
  mutable ll m, c, p;
  bool operator<(const Line& o) const { return m < o.m; }
  bool operator<(ll x) const { return p < x; } };
//for min query LineContainer(true), normally->max
struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  static const ll inf = LLONG_MAX;
  int mnQ=1;
  LineContainer(bool minQuery=false){
    if(minQuery) mnQ=-1; else mnQ=1; }
  ll div(ll a, ll b) { // floored division
    return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator x, iterator y) {
    if (y == end()) return x->p = inf, 0;
    if (x->m == y->m) x->p = x->c > y->c ? inf : -inf;
    else x->p = div(y->c - x->c, x->m - y->m);
    return x->p >= y->p; }
  void add(ll m, ll c) {
    m*=mnQ, c*=mnQ;
    auto z = insert({m, c, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if(x != begin() && isect(--x,y)) isect(x,y = erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
      isect(x, erase(y)); }
  ll query(ll x) {
    assert(!empty()); auto l = *lower_bound(x);
    return mnQ*(l.m * x + l.c); } };
score(l,r) = sum{(i-l)*ai} ; i=l+1 ... r
           = sum{i*ai} - l*sum{ai}
           = ms[r]-ms[l]-l*s[r]+l*s[l]
   ans = max{ms[i]-ms[j]-j*s[i]+j*s[j]} for j<i
       = max{ (-j)*(s[i])+(j*s[j])-ms[j] + ms[i]}
       = max{ fj(s[i]) + ms[i]}
       = max{ fj(s[i])} + ms[i]
    fj(x) = -j*(x)+(j*s[j]-ms[j])
container.add(0,0);
for(i=1; i<=n; i++){
  ans=max(ans, container.query(cum[i])+cum2[i]);
  container.add(-i, i*cum[i]-cum2[i]); }
```

2.3 DC Opt

```
ll dp[N]; ll cost(int l, int r){ return 1; } //dummy
void solve(int l, int r, int ql, int qr) {
  if(l > r) return; int mid = l + (r - l) / 2;
  ll pos =0, val = INT_MAX; //check min or max ?
  for(int k = ql; k <= min(mid, qr); k++) {
    ll ret = cost(k, mid);
    if(ret < val) val = ret, pos = k; /*min or max? */ }
  dp[mid] = val;
  // for H[j]<=H[j+1] i.e. cost(1,j)<=cost(1,j+1)
  solve(l, mid - 1, ql, pos);
  solve(mid + 1, r, pos, qr);
  /* for H[j]>=H[j+1] i.e. cost(1,j)>=cost(1,j+1)
  solve(l, mid - 1, pos, qr);
  solve(mid + 1, r, ql, pos); */ }
```

2.4 Digit DP

```
vector<int> A,B; ll DP[20][150][2][2];
//dp[curPos][sum][a is smaller than R][a is bigger than L]
ll call(int pos, int sum, int small, int big){
  if(pos==(int)B.size()) return (ll)sum;
  ll &res = DP[pos][sum][small][big];
  if(~res) return res; res = 0;
  int start=A[pos], stop=B[pos];
  if(big) start=0; if(small) stop=9;
  for(int dgt=start; dgt<=stop; dgt++)
   res+=call(pos+1,sum+dgt,small|dgt<B[pos],big|dgt>A[pos]);
  return res; }
ll solve(ll a, ll b){
  A.clear();B.clear();
  while(b>0){ B.push_back(b%10); b/=10; }
  while(a>0){ A.push_back(a%10); a/=10; }
  while(A.size()<B.size()) A.push_back(0);
  reverse(A.begin(), A.end()); reverse(B.begin(), B.end());
  memset(DP, -1, sizeof(DP)); return call(0, 0, 0, 0); }
```

2.5 LIS

```
int n,a[N],B[N],len[N];
int find_lis(){
  int ans=0;
  for(int i=1; i<=n; i++){
    len[i]=lower_bound(B+1, B+ans+1, a[i])-B;//Incresing
  //len[i]=upper_bound(B+1, B+ans+1, a[i])-B;//Non-decresing
    ans=max(ans, len[i]); B[len[i]]=a[i]; } return ans; }
void print_lis(int ans){
  vector<int> seq;
  for(int i=n; i>=1; i--){
    if(len[i]==ans){ seq.push_back(a[i]); ans--; } }
  int i=(int)seq.size();
  while(i--) cout<<seq[i]<<" \n"[!i]; }
void init(){ clean(B,-1); clean(len,0); }
```

2.6 SOS dp

```
//O(N*2^N)
void 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)];
  }
}
```

3 GeoMetry

3.1 Closest Pair

```
struct PT{
T x,y;int id; PT(){} PT(T x,T y,int id):x(x),y(y),id(id){}};
struct cmp_x{
  bool operator()(const PT &a, const PT &b) const {
    return a.x < b.x || (a.x == b.x && a.y < b.y); } };
struct cmp_y{
  bool operator()(const PT &a, const PT &b) const {
     return a.y < b.y; } };
int n; vector<PT> p,t; ll mindist; pair<int,int> best_pair;
void upd_ans(const PT &a, const PT &b){
  ll dist=(a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
  if(dist<mindist){ mindist=dist; best_pair={a.id,b.id}; }}
//[l,r)
void rec(int l, int r){
  if(r-l <= 3){
    for(int i = l; i < r; ++i){
      for(int j = i + 1; j < r; ++j) upd_ans(p[i], p[j]); }
    sort(p.begin() + l, p.begin() + r, cmp_y()); return; }
  int m=(l+r) >> 1; int midx=p[m].x; rec(l, m); rec(m, r);
  merge(p.begin()+l, p.begin()+m, p.begin()+m, p.begin()+r,
       t.begin(), cmp_y());
  copy(t.begin(), t.begin()+r-l, p.begin()+l);
  int tsz = 0;
  for(int i = l; i < r; ++i) {
    if(abs(p[i].x - midx) < mindist){
      for(int j=tsz-1; j>=0 && p[i].y-t[j].y<mindist; --j)
       upd_ans(p[i], t[j]); t[tsz++] = p[i]; } } }
void closestPair(int n){
 t.resize(n); sort(p.begin(), p.end(), cmp_x());
 mindist = (ll)18446744073709551610; rec(0, n); }
```

3.2 Pick Theorem

```
theorem:area=In+(on_boundary/2)-1;in=strictly inside polygon
T point(PT a,PT b){ T x=abs(a.x-b.x);
  T y=abs(a.y-b.y); return __gcd(x,y)-1; }
main(){ ll on_boundary=n,area=0LL;
 for(i=0; i<n; i++){ area+=cross(p[i],p[(i+1)%n]);
   on_boundary+=point(p[i],p[(i+1)%n]); } }
```

3.3 Template

```
typedef long double ld; //typedef long long T;
typedef long double T; #define PI acos(-1.0)
#define eps 1e-9 #define inf (1e100)
int sign(T x) {return (x>eps)-(x<-eps);}
struct PT{
  T x,y; PT(){x=0,y=0;}
  PT(T x,T y) : x(x),y(y){}
  PT(const PT& p) : x(p.x), y(p.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 T a) const { return PT(x * a, y * a);
      }
  friend PT operator * (const T &a, const PT &b) { return PT
      (a * b.x, a * b.y); }
  PT operator / (const T a) const { return PT(x / a, y / a);
      }
  PT operator*(PT p) {return {x*p.x-y*p.y, x*p.y+y*p.x};}
  PT operator/(PT p) {return {((*this)*p.conj())/(p*p.conj()
      ).x};}
  bool operator == (PT a) const { return sign(a.x - x) == 0
      && sign(a.y - y) == 0; }
  bool operator != (PT a) const { return !(*this == a); }
  bool operator < (PT a) const { return sign(a.x - x) == 0 ?
      y < a.y : x < a.x; }
  bool operator > (PT a) const { return sign(a.x - x) == 0 ?
      y > a.y : x > a.x; }
  ld norm() { return sqrt(x * x + y * y); }
  T norm2() { return x * x + y * y; }//squared dis
  PT perp() { return PT(-y, x); }
  PT unit() {return (*this)/norm();}
  PT conj() {return {x,-y};}
  ld arg() { return atan2(y, x); }
  //a vector with norm r and having same direction
  PT truncate(T r) {
    ld k = norm(); if(!sign(k)) return *this;
    r /= k; return PT(x * r, y * r); }
};
typedef vector<PT> VPT;
T dot(PT a, PT b) { return a.x * b.x + a.y * b.y; }
T dist2(PT a, PT b) { return dot(a - b, a - b); }
ld dist(PT a, PT b) { return sqrt(dot(a - b, a - b)); }
T cross(PT a, PT b) { return a.x * b.y - a.y * b.x; }
T cross2(PT a, PT b, PT c) { return cross(b - a, c - a); }
int orientation(PT a, PT b, PT c) { return sign(cross(b - a,
      c - a)); }
PT perp(PT a) { return PT(-a.y, a.x); }
PT rotate_ccw90(PT a) { return PT(-a.y, a.x); }
PT rotate_cw90(PT a) { return PT(a.y, -a.x); }
PT rotate_ccw(PT a, T t) { return PT(a.x * cos(t) - a.y *
     sin(t), a.x * sin(t) + a.y * cos(t)); }
PT rotate_cw(PT a, T t) { return PT(a.x * cos(t) + a.y * sin
     (t), -a.x * sin(t) + a.y * cos(t)); }
//O er sapekke A ke radian kune CCW
PT rotate(PT O, PT A, T radian){ return O + rotate_ccw(A - O
     , radian); }
T SQ(T x) { return x * x; }
ld rad_to_deg(T r) { return (r * 180.0 / PI); }
ld deg_to_rad(T d) { return (d * PI / 180.0); }
T get_angle(PT a, PT b) {
    T costheta = dot(a, b) / a.norm() / b.norm();
    return acos(max((T)-1.0, min((T)1.0, costheta))); }
// does point p lie in angle <bac
bool is_point_in_angle(PT b, PT a, PT c, PT p) {
    assert(orientation(a, b, c) != 0);
    if (orientation(a, c, b) < 0) swap(b, c);
    return orientation(a, c, p) >= 0 && orientation(a, b, p)
         <= 0; }
bool half(PT p) {
    return p.y > 0.0 || (p.y == 0.0 && p.x < 0.0); }
// sort points in counterclockwise
void polar_sort(VPT &v) {
    sort(v.begin(), v.end(), [](PT a,PT b) {
        return make_tuple(half(a), (T)0.0, a.norm2()) <
             make_tuple(half(b), cross(a, b), b.norm2()); });
    }
struct line {
  PT a, b;// goes through points a and b
  PT v; T c;//line form: direction vec [cross](x,y)=c
  line() {}
  //direction vector v and offset c
  line(PT v, T c) : v(v), c(c) {
    auto p = get_points(); a = p.first; b = p.second; }
  // equation ax + by + c = 0
  line(T _a, T _b, T _c) : v({_b, -_a}), c(-_c) {
      auto p = get_points();
      a = p.first; b = p.second; }
  //goes through points p and q
  line(PT p, PT q) : v(q - p), c(cross(v, p)), a(p), b(q) {}
    pair<PT, PT> get_points() {
    PT p, q; T a = -v.y, b = v.x; // ax + by = c
    if(sign(a) == 0) {
      p = PT(0, c / b); q = PT(1, c / b); }
    else if(sign(b) == 0) {
      p = PT(c / a, 0); q = PT(c / a, 1); }
    else {
      p = PT(0, c / b); q = PT(1, (c - a) / b); }
    return {p, q}; }
  array<T, 3> get_abc() {
    T a = -v.y, b = v.x; return {a, b, c}; }
  //1 if on the left,-1 if on the right,0 if on the line
  int side(PT p) { return sign(cross(v, p) - c); }
  line perpendicular_through(PT p){return {p,p+perp(v)};}
  line translate(PT t) { return {v, c + cross(v, t)}; }
  bool cmp_by_projection(PT p, PT q) { return dot(v, p) <
       dot(v, q); }
  line shift_left(T d) {
    PT z = v.perp().truncate(d);
    return line(a + z, b + z); }
};
//find a point from A through B with distance d
PT point_along_line(PT a, PT b, T d) {
  return a + (((b - a) / (b - a).norm()) * d); }
PT project_from_point_to_line(PT a, PT b, PT c) {
  return a+(b-a)*dot(c-a,b-a)/(b - a).norm2(); }
PT reflection_from_point_to_line(PT a, PT b, PT c) {
  PT p = project_from_point_to_line(a,b,c);
  return point_along_line(c, p, (T)2 * dist(c, p)); }
ld dist_from_point_to_line(PT a, PT b, PT c) {
  return fabs(cross(b - a, c - a) / (b - a).norm()); }
bool is_point_on_seg(PT a, PT b, PT p) {
  if (abs(cross(p - b, a - b)) < eps) {
    if(p.x<min(a.x,b.x) || p.x>max(a.x,b.x)) return 0;
    if(p.y<min(a.y,b.y) || p.y>max(a.y,b.y)) return 0;
    return 1; } return 0; }
//minimum distance point from point c to segment ab that
     lies on segment ab
PT project_from_point_to_seg(PT a, PT b, PT c) {
  T r = dist2(a, b); if(abs(r) < eps) return a;
  r = dot(c - a, b - a) / r; if (r < 0) return a;
  if (r > 1) return b; return a + (b - a) * r; }
ld dist_from_point_to_seg(PT a, PT b, PT c) {
  return dist(c, project_from_point_to_seg(a, b, c));}
//0 if not parallel, 1 if parallel, 2 if collinear
int is_parallel(PT a, PT b, PT c, PT d) {
  T k = abs(cross(b - a, d - c));
  if (k < eps){
    if (abs(cross(a - b, a - c)) < eps && abs(cross(c - d, c
        - a)) < eps) return 2;
    else return 1;
  } else return 0; }
bool are_lines_same(PT a, PT b, PT c, PT d) {
  if(abs(cross(a - c, c - d)) < eps && abs(cross(b - c, c -
      d)) < eps) return true; return false; }
//bisector vector of <abc
PT angle_bisector(PT &a, PT &b, PT &c){
  PT p=a-b,q=c-b; return p+q*sqrt(dot(p,p)/dot(q,q)); }
//1 if point is ccw to the line, 2 if point is cw to the
    line, 3 if point is on the line
int point_line_relation(PT a, PT b, PT p) {
  int c = sign(cross(p - a, b - a)); if (c < 0) return 1;
  if (c > 0) return 2; return 3;}
bool line_line_intersection(PT a,PT b,PT c,PT d,PT &ans){
  T a1 = a.y - b.y, b1 = b.x - a.x, c1 = cross(a, b);
  T a2 = c.y - d.y, b2 = d.x - c.x, c2 = cross(c, d);
  T det=a1*b2-a2*b1; if(det == 0) return 0;
  ans=PT((b1*c2-b2*c1)/det,(c1*a2-a1*c2)/det); return 1;}
bool seg_seg_intersection(PT a,PT b,PT c,PT d,PT &ans){
  T oa = cross2(c, d, a), ob = cross2(c, d, b);
  T oc = cross2(a, b, c), od = cross2(a, b, d);
  if (oa * ob < 0 && oc * od < 0){
    ans=(a*ob-b*oa)/(ob-oa); return 1; } else return 0;}
//se.size()==0 means no intersection
//se.size()==1 means one intersection
//se.size()==2 means range intersection
set<PT> seg_seg_intersection_inside(PT a,PT b,PT c,PT d){
  PT ans; set<PT> se;
  if(seg_seg_intersection(a,b,c,d,ans)) return {ans};
  if (is_point_on_seg(c, d, a)) se.insert(a);
  if (is_point_on_seg(c, d, b)) se.insert(b);
  if (is_point_on_seg(a, b, c)) se.insert(c);
  if(is_point_on_seg(a,b,d)) se.insert(d); return se;}
//[ab],(cd). 0->not, 1->proper,2->segment intersect
int seg_line_relation(PT a, PT b, PT c, PT d) {
  T p = cross2(c, d, a); T q = cross2(c, d, b);
  if (sign(p) == 0 && sign(q) == 0) return 2;
  else if (p * q < 0) return 1; else return 0; }
bool seg_line_intersection(PT a,PT b,PT c,PT d,PT &ans){
  bool k=seg_line_relation(a, b, c, d); assert(k != 2);
  if(k) line_line_intersection(a,b,c,d,ans); return k; }
ld dist_from_seg_to_seg(PT a, PT b, PT c, PT d) {
  PT dummy;
  if(seg_seg_intersection(a,b,c,d,dummy)) return (T)0.0;
  else return min({dist_from_point_to_seg(a,b,c),
    dist_from_point_to_seg(a, b, d),
    dist_from_point_to_seg(c, d, a),
    dist_from_point_to_seg(c, d, b)});}
```

```
//c to [a,b)
ld dist_from_point_to_ray(PT a, PT b, PT c) {
  b=a+b; T r=dot(c-a,b-a); if(r<0.0) return dist(c,a);
  return dist_from_point_to_line(a, b, c);}
//starting point as and direction vector ad
bool ray_ray_intersection(PT as, PT ad, PT bs, PT bd){
  T dx = bs.x - as.x, dy = bs.y - as.y;
  T det = bd.x * ad.y - bd.y * ad.x;
  if(abs(det)<eps) return 0;
  T u = (dy * bd.x - dx * bd.y) / det;
  T v = (dy * ad.x - dx * ad.y) / det;
  if(sign(u) >= 0 && sign(v) >= 0) return 1;
  else return 0; }
ld ray_ray_distance(PT as,PT ad,PT bs,PT bd){
  if(ray_ray_intersection(as,ad,bs,bd)) return 0.0;
  ld ans = dist_from_point_to_ray(as, ad, bs);
  ans = min(ans, dist_from_point_to_ray(bs,bd,as));
  return ans;}
struct circle {
  PT p; T r; circle() {}
  circle(PT _p, T _r): p(_p), r(_r) {};
  circle(T x, T y, T _r): p(PT(x, y)), r(_r) {};
  //circumcircle of a triangle
  circle(PT a, PT b, PT c) {
    b=(a+b)*0.5; c=(a+c)*0.5;
    line_line_intersection(b, b + rotate_cw90(a - b), c, c +
        rotate_cw90(a - c), p);
    r = dist(a, p); }
  //inscribed circle of a triangle
  circle(PT a, PT b, PT c, bool t) {
    line u, v; ld m = atan2(b.y-a.y, b.x-a.x);
    ld n = atan2(c.y-a.y, c.x-a.x); u.a = a;
    u.b=u.a+(PT(cos((n+m)/2.0),sin((n+m)/2.0))); v.a=b;
    m=atan2(a.y-b.y,a.x-b.x); n=atan2(c.y-b.y,c.x-b.x);
    v.b = v.a+(PT(cos((n + m)/2.0), sin((n + m)/2.0)));
    line_line_intersection(u.a, u.b, v.a, v.b, p);
    r = dist_from_point_to_seg(a, b, p); }
  bool operator==(circle v){ return p == v.p && sign(r - v.r
      ) == 0; }
  ld area() { return PI * r * r; }
  ld circumference() { return 2.0 * PI * r; } };
//0->outside,1->on circumference, 2->inside circle
int circle_point_relation(PT p, T r, PT b) {
  ld d = dist(p,b); if(sign(d - r) < 0) return 2;
  if(sign(d - r) == 0) return 1; return 0; }
int circle_line_relation(PT p, T r, PT a, PT b){
  ld d = dist_from_point_to_line(a, b, p);
  if(sign(d - r) < 0) return 2;
  if(sign(d - r) == 0) return 1; return 0; }
VPT circle_line_intersection(PT c,T r,PT a,PT b){
```

```
  VPT ret; b = b - a; a = a - c;
  T A = dot(b, b), B = dot(a, b);
  T C=dot(a,a)-r*r, D=B*B-A*C; if(D < -eps) return ret;
  ret.push_back(c + a + b * (-B + sqrt(D + eps)) / A);
  if(D > eps) ret.push_back(c+a+b*(-B - sqrt(D)) / A);
  return ret;  }
//5 - outside and do not intersect
//4 - intersect outside in one point
//3 - intersect in 2 points
//2 - intersect inside in one point
//1 - inside and do not intersect
int circle_circle_relation(PT a, T r, PT b, T R) {
  ld d = dist(a, b); if(sign(d - r - R) > 0) return 5;
  if(sign(d - r - R) == 0) return 4; ld l = fabs(r - R);
  if(sign(d - r - R) < 0 && sign(d - l) > 0) return 3;
  if(sign(d-l) == 0) return 2; if(sign(d-l)<0) return 1;
  assert(0); return -1; }
VPT circle_circle_intersection(PT a,T r,PT b,T R){
  if(a == b && sign(r-R) == 0) return {PT(1e18, 1e18)};
  VPT ret; ld d = sqrt(dist2(a, b));
  if(d > r + R || d + min(r,R) < max(r,R)) return ret;
  ld x = (d * d - R * R + r * r) / (2 * d);
  ld y = sqrt(r * r - x * x); PT v = (b - a) / d;
  ret.push_back(a + v * x + rotate_ccw90(v) * y);
  if(y>0) ret.push_back(a + v * x - rotate_ccw90(v) * y);
  return ret; }
//circle through points a, b and of radius r, return cnt
int get_circle(PT a, PT b, T r, circle &c1, circle &c2){
  VPT v = circle_circle_intersection(a, r, b, r);
  int t = v.size(); if(!t) return 0; c1.p=v[0], c1.r=r;
  if(t == 2) c2.p = v[1], c2.r = r; return t; }
//returns two circle c1, c2 which is tangent to line u
//goes through point q and has radius r1; returns cnt
int get_circle(line u,PT q,T r1,circle &c1,circle &c2){
  T d = dist_from_point_to_line(u.a, u.b, q);
  if(sign(d - r1 * 2.0) > 0) return 0;
  if(sign(d) == 0) {
    c1.p = q + rotate_ccw90(u.v).truncate(r1);
    c2.p = q + rotate_cw90(u.v).truncate(r1);
    c1.r = c2.r = r1; return 2; }
  line u1 = line(u.a + rotate_ccw90(u.v).truncate(r1), u.b +
       rotate_ccw90(u.v).truncate(r1));
  line u2 = line(u.a + rotate_cw90(u.v).truncate(r1), u.b +
      rotate_cw90(u.v).truncate(r1));
  circle cc = circle(q, r1); PT p1, p2; VPT v;
  v = circle_line_intersection(q, r1, u1.a, u1.b);
if(!v.size()) v=circle_line_intersection(q,r1,u2.a,u2.b);
  v.push_back(v[0]); p1 = v[0], p2 = v[1];
  c1 = circle(p1, r1);
  if(p1 == p2) { c2 = c1; return 1; }
  c2 = circle(p2, r1); return 2; }
//area of intersection between two circles
ld circle_circle_area(PT a, T r1, PT b, T r2){
  ld d=(a-b).norm(); if(r1+r2 < d+eps) return 0;
  if(r1 + d < r2 + eps) return PI * r1 * r1;
  if(r2 + d < r1 + eps) return PI * r2 * r2;
  double theta_1 = acos((r1 * r1 + d * d - r2 * r2) / (2 *
       r1 * d)), theta_2 = acos((r2 * r2 + d * d - r1 * r1)
       /(2 * r2 * d));
  return r1 * r1 * (theta_1 - sin(2 * theta_1)/2.) + r2 * r2
        * (theta_2 - sin(2 * theta_2)/2.); }
//tangent lines from point q to the circle
int tangent_lines_from_point(PT p, T r, PT q, line &u, line
     &v) { int x = sign(dist2(p, q) - r * r);
  if(x<0) return 0;
  if(x==0){u=line(q,q+rotate_ccw90(q-p));v=u; return 1;}
  ld d=dist(p,q); ld l=r*r/d; ld h=sqrt(r*r - l*l);
  u = line(q, p + ((q - p).truncate(l) + (rotate_ccw90(q - p
       ).truncate(h))));
  v = line(q, p + ((q - p).truncate(l) + (rotate_cw90(q - p)
       .truncate(h)))); return 2; }
// returns outer tangents line of two circles
// if inner == 1 it returns inner tangent lines
int tangents_lines_from_circle(PT c1, T r1, PT c2, T r2,
     bool inner, line &u, line &v) {
  if(inner) r2 = -r2; PT d = c2 - c1;
  T dr = r1 - r2, d2 = d.norm(), h2 = d2 - dr * dr;
  if(d2 == 0 || h2 < 0) { return 0;}
  vector<pair<PT, PT>>out;
  for(int tmp: {- 1, 1}) {
    PT v = (d*dr+rotate_ccw90(d)*sqrt(h2)*tmp)/d2;
    out.push_back({c1 + v * r1, c2 + v * r2}); }
  u = line(out[0].first, out[0].second);
  if(out.size()==2) v=line(out[1].first,out[1].second);
  return 1 + (h2 > 0); }
//O(n^2 log n)
struct CircleUnion {
  int n,covered[2020]; T x[2020],y[2020],r[2020];
  vector<pair<T, T> > seg, cover; T arc, pol;
  int sign(T x) {return x < -eps ? -1 : x > eps;}
  int sign(T x, T y) {return sign(x - y);}
  T SQ(const T x) {return x * x;}
  ld dist(T x1, T y1, T x2, T y2) {return sqrt(SQ(x1 - x2) +
        SQ(y1 - y2));}
  ld angle(T A, T B, T C) {
    ld val = (SQ(A) + SQ(B) - SQ(C)) / (2 * A * B);
    if(val < -1) val = -1; if(val > +1) val = +1;
    return acos(val); }
  CircleUnion(){
    n = 0; seg.clear(), cover.clear(); arc = pol = 0; }
```

```
  void init(){
    n = 0; seg.clear(), cover.clear(); arc = pol = 0; }
  void add(T xx, T yy, T rr) {
    x[n]=xx, y[n]=yy, r[n]=rr, covered[n]=0, n++; }
  void getarea(int i, T lef, T rig){
    arc += 0.5*r[i]*r[i]*(rig - lef - sin(rig - lef));
    ld x1 = x[i]+r[i]*cos(lef),y1 = y[i]+r[i]*sin(lef);
    ld x2 = x[i]+r[i]*cos(rig),y2 = y[i]+r[i]*sin(rig);
    pol += x1 * y2 - x2 * y1; }
  ld solve() {
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < i; j++) {
      if(!sign(x[i] - x[j]) && !sign(y[i] - y[j]) && !sign(r[
           i] - r[j])) { r[i] = 0.0; break; }}}
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
      if(i != j && sign(r[j] - r[i]) >= 0 && sign(dist(x[i],
           y[i], x[j], y[j]) - (r[j] - r[i])) <= 0) { covered
           [i] = 1; break; }}}
  for(int i = 0; i < n; i++) {
    if(sign(r[i]) && !covered[i]) {
      seg.clear();
      for(int j = 0; j < n; j++) {
        if(i != j) {
          ld d = dist(x[i], y[i], x[j], y[j]);
          if(sign(d - (r[j] + r[i])) >= 0 || sign(d - abs(r[j
               ] - r[i])) <= 0) { continue; }
          ld alpha = atan2(y[j] - y[i], x[j] - x[i]);
          ld beta = angle(r[i], d, r[j]);
          pair<ld,ld> tmp(alpha - beta, alpha + beta);
          if(sign(tmp.first)<=0 && sign(tmp.second)<=0){ seg.
               push_back(pair<ld, ld>(2 * PI + tmp.first, 2 *
                PI + tmp.second)); }
          else if (sign(tmp.first) < 0){
            seg.push_back(pair<ld, ld>(2 * PI + tmp.first, 2
                 * PI)); seg.push_back(pair<ld, ld>(0, tmp.
                 second)); }
          else { seg.push_back(tmp); } }}
          sort(seg.begin(), seg.end()); ld rig = 0;
          for(auto it=seg.begin(); it!=seg.end();it++){
            if(sign(rig - it->first) >= 0) {
              rig = max(rig, it->second); }
            else { getarea(i, rig, it->first);
              rig = it->second; } }
          if(!sign(rig)) { arc += r[i] * r[i] * PI;}
          else { getarea(i, rig, 2 * PI); }} }
    return pol / 2.0 + arc; }  } CU;
ld area_of_triangle(PT a, PT b, PT c) {
  return fabs(cross(b - a, c - a) * 0.5); }
//-1->inside, 0->on the polygon, 1->outside
```

```
int is_point_in_triangle(PT a, PT b, PT c, PT p) {
  if(sign(cross(b - a,c - a)) < 0) swap(b, c);
  int c1 = sign(cross(b - a,p - a));
  int c2 = sign(cross(c - b,p - b));
  int c3 = sign(cross(a - c,p - c));
  if(c1<0 || c2<0 || c3 < 0) return 1;
  if(c1 + c2 + c3 != 3) return 0; return -1; }
T perimeter(VPT &p) {
  T ans=0; int n = p.size();
  for(int i=0;i<n;i++) ans+=dist(p[i],p[(i+1) % n]);
  return ans;  }
ld area(VPT &p) { ld ans=0; int n=p.size();
  for(int i=0;i<n;i++) ans+=cross(p[i],p[(i+1)%n]);
  return fabs(ans) * 0.5; }
//centroid of a (possibly non-convex) polygon, assuming that
     the coordinates are listed in a CW or CCW fashion
PT centroid(VPT &p) {
  int n = p.size(); PT c(0, 0); ld sum = 0;
  for(int i=0;i<n;i++) sum+=cross(p[i],p[(i+1)%n]);
  ld scale = 3.0 * sum;
  for(int i = 0; i < n; i++) {
    int j=(i+1)%n; c=c+(p[i]+p[j])*cross(p[i],p[j]); }
  return c / scale; }
// 0 if cw, 1 if ccw
bool get_direction(VPT &p) {
  ld ans = 0; int n = p.size();
  for(int i=0;i<n;i++) ans += cross(p[i],p[(i+1)%n]);
  if(sign(ans) > 0) return 1; return 0; }
//returns a point such that the sum of distances
//from that point to all points in p is minimum
//O(n log^2 MX)
PT geometric_median(VPT p) {
  auto tot_dist = [&](PT z) {
    T res = 0;
    for(int i=0; i<p.size(); i++) res += dist(p[i], z);
    return res; };
  auto findY = [&](T x) {
    T yl = -1e5, yr = 1e5;
    for(int i = 0; i < 60; i++) {
      T ym1 = yl + (yr - yl) / 3;
      T ym2 = yr - (yr - yl) / 3;
      T d1 = tot_dist(PT(x, ym1));
      T d2 = tot_dist(PT(x, ym2));
      if (d1 < d2) yr = ym2; else yl = ym1; }
    return pair<T, T> (yl, tot_dist(PT(x, yl)));
  };
  T xl = -1e5, xr = 1e5;
  for(int i = 0; i < 60; i++) {
    T xm1 = xl + (xr - xl) / 3;
    T xm2 = xr - (xr - xl) / 3;
    T y1, d1, y2, d2;
    auto z=findY(xm1); y1=z.first; d1=z.second;
    z=findY(xm2); y2 = z.first; d2 = z.second;
    if (d1 < d2) xr = xm2; else xl = xm1; }
  return {xl, findY(xl).first }; }
VPT convex_hull(VPT &p) {
  if(p.size() <= 1) return p; VPT v = p;
  sort(v.begin(), v.end()); VPT up, dn;
  for(auto& p : v) {
    while (up.size() > 1 && orientation(up[up.size() - 2], up
         .back(), p) >= 0) { up.pop_back(); }
    while (dn.size() > 1 && orientation(dn[dn.size() - 2], dn
         .back(), p) <= 0) { dn.pop_back(); }
    up.push_back(p); dn.push_back(p); } v = dn;
  if(v.size() > 1) v.pop_back();
  reverse(up.begin(), up.end());
  up.pop_back(); for(auto& p:up){v.push_back(p);}
  if(v.size() == 2 && v[0] == v[1]) v.pop_back();
  return v; }
bool is_convex(VPT &p) {
  bool s[3]; s[0]=s[1]=s[2]=0; int n = p.size();
  for(int i = 0; i < n; i++) {
    int j = (i+1)%n; int k = (j + 1) % n;
    s[sign(cross(p[j] - p[i], p[k] - p[i])) + 1] = 1;
    if(s[0] && s[2]) return 0; } return 1; }
//-1->inside,0->on boundary,1->outside;// O(log n)
int is_point_in_convex(VPT &p, const PT& x) {
  int n = p.size(); int a = orientation(p[0], p[1], x);
  int b=orientation(p[0],p[n-1],x);if(a<0||b>0) return 1;
  int l = 1, r = n - 1;
  while(l + 1 < r) { int mid = l + r >> 1;
    if(orientation(p[0], p[mid], x) >= 0) l = mid;
    else r = mid; }
  int k=orientation(p[l],p[r],x); if(k<=0) return -k;
  if(l == 1 && a == 0) return 0;
  if(r==n-1 && b==0) return 0; return -1; }
bool is_point_on_polygon(VPT &p, const PT& z) {
  int n = p.size();
  for(int i = 0; i < n; i++)
    if(is_point_on_seg(p[i], p[(i+1)%n], z)) return 1;
  return 0; }
int winding_number(VPT &p, const PT& z){
  if (is_point_on_polygon(p, z)) return 1e9;
  int n = p.size(), ans = 0;
  for(int i = 0; i < n; ++i) {
    int j=(i+1)%n; bool below = p[i].y < z.y;
    if(below != (p[j].y < z.y)) {
      auto orient = orientation(z, p[j], p[i]);
      if(orient == 0) return 0;
      if(below == (orient > 0)) ans += below ? 1 : -1;
```

```
    } } return ans; }
int is_point_in_polygon(VPT &p, const PT& z){
  int k = winding_number(p, z);
  return k == 1e9 ? 0 : k == 0 ? 1 : -1; }
//id of the vertex having maximum dot product with z
//top - upper right vertex
//for minimum dot prouct negate z and return -dot(z,p[id])
int extreme_vertex(VPT &p,const PT &z,int top){
  int n = p.size(); if (n == 1) return 0;
  T ans = dot(p[0], z); int id = 0;
  if(dot(p[top], z) > ans) ans=dot(p[top], z), id=top;
  int l = 1, r = top - 1;
  while(l < r){ int mid = l + r >> 1;
    if(dot(p[mid+1], z) >= dot(p[mid], z)) l = mid+1;
    else r = mid; }
  if(dot(p[l], z) > ans) ans = dot(p[l], z), id = l;
  l = top + 1, r = n - 1;
  while(l < r){ int mid = l + r >> 1;
    if(dot(p[(mid+1)%n],z)>=dot(p[mid],z)) l = mid+1;
    else r = mid; } l %= n;
  if(dot(p[l],z)>ans) ans=dot(p[l],z),id=l; return id; }
ld diameter(VPT &p){ int n = (int)p.size();
  if(n==1) return 0; if(n==2) return dist(p[0],p[1]);
  T ans = 0; int i = 0, j = 1;
  while(i < n){
    while(cross(p[(i+1)%n]-p[i],p[(j+1)%n]-p[j])>=0){
      ans=max(ans, dist2(p[i], p[j])); j = (j+1)%n; }
    ans = max(ans, dist2(p[i], p[j])); i++;
  }  return sqrt(ans); }
ld width(VPT &p){
  int n = p.size(); if(n <= 2) return 0;
  ld ans = inf; int i = 0, j = 1;
  while(i < n){
    while(cross(p[(i + 1) % n] - p[i], p[(j + 1) % n] - p[j])
         >= 0) j = (j + 1) % n;
    ans = min(ans, dist_from_point_to_line(p[i], p[(i + 1) %
         n], p[j])); i++; } return ans; }
ld minimum_enclosing_rectangle(VPT &p){
  int n = p.size(); if (n <= 2) return perimeter(p);
  int mndot = 0; T tmp = dot(p[1] - p[0], p[0]);
  for(int i = 1; i < n; i++) {
    if(dot(p[1] - p[0], p[i]) <= tmp) {
      tmp = dot(p[1] - p[0], p[i]); mndot = i; } }
  ld ans = inf; int i = 0, j = 1, mxdot = 1;
  while(i < n){ PT cur = p[(i + 1) % n] - p[i];
    while(cross(cur,p[(j+1)%n]-p[j])>=0) j=(j+1)%n;
    while (dot(p[(mxdot + 1) % n], cur) >= dot(p[mxdot], cur)
         ) mxdot = (mxdot + 1) % n;
    while (dot(p[(mndot + 1) % n], cur) <= dot(p[mndot], cur)
         ) mndot = (mndot + 1) % n;
```

```
    ans = min(ans, 2.0 * ((dot(p[mxdot], cur) / cur.norm() -
         dot(p[mndot], cur) / cur.norm()) +
         dist_from_point_to_line(p[i], p[(i + 1) % n], p[j]))
         ); i++; } return ans; }
//expected O(n), 1st call convex_hull() for faster
circle minimum_enclosing_circle(VPT &p){
  random_shuffle(p.begin(), p.end());
  int n=p.size(); circle c(p[0], 0);
  for(int i = 1; i < n; i++) {
    if(sign(dist(c.p, p[i]) - c.r) > 0) {
      c = circle(p[i], 0);
      for(int j = 0; j < i; j++) {
        if(sign(dist(c.p, p[j]) - c.r) > 0) {
          c=circle((p[i]+p[j])/2,dist(p[i],p[j])/2);
          for(int k = 0; k < j; k++) {
            if(sign(dist(c.p, p[k]) - c.r) > 0)
              c = circle(p[i], p[j], p[k]);
          } } } } } return c; }
//returns a vector with the vertices of a polygon with
     everything to the left of the line going from a to b
     cut away
VPT cut(VPT &p, PT a, PT b) {
  VPT ans; int n = (int)p.size();
  for(int i = 0; i < n; i++) {
    T c1=cross(b-a,p[i]-a); T c2=cross(b-a,p[(i+1)%n]-a);
    if(sign(c1) >= 0) ans.push_back(p[i]);
    if(sign(c1 * c2) < 0) {
      if(!is_parallel(p[i], p[(i + 1) % n], a, b)){
        PT tmp; line_line_intersection(p[i],p[(i+1)%n],a,b,
             tmp); ans.push_back(tmp); }
    } } return ans; }
//not necessarily convex, boundary is included in the
     intersection, returns total intersected length
ld polygon_line_intersection(VPT &p, PT a, PT b){
  int n=p.size(); p.push_back(p[0]); line l=line(a, b);
  ld ans = 0.0; vector< pair<ld, int> > vec;
  for(int i = 0; i < n; i++) {
    int s1 = sign(cross(b - a, p[i] - a));
    int s2 = sign(cross(b - a, p[i+1] - a));
    if(s1 == s2) continue;
    line t = line(p[i], p[i + 1]);
    PT inter = (t.v * l.c - l.v * t.c) / cross(l.v, t.v);
    ld tmp = dot(inter, l.v); int f;
    if(s1 > s2) f = s1 && s2 ? 2 : 1;
    else f = s1 && s2 ? -2 : -1;
    vec.push_back(make_pair(tmp, f)); }
  sort(vec.begin(), vec.end());
  for(int i=0,j=0; i+1 < (int)vec.size(); i++){
    j += vec[i].second;
    if (j) ans += vec[i + 1].first - vec[i].first; }
  ans = ans / sqrt(dot(l.v, l.v)); p.pop_back();
  return ans; }
pair<PT,int> point_poly_tangent(VPT &p,PT Q,int dir,int l,
     int r){
  while(r - l > 1) {
    int mid = (l + r) >> 1;
    bool pvs = orientation(Q, p[mid], p[mid - 1]) != -dir;
    bool nxt = orientation(Q, p[mid], p[mid + 1]) != -dir;
    if(pvs && nxt) return {p[mid], mid};
    if(!(pvs || nxt)) {
      auto p1=point_poly_tangent(p, Q, dir, mid + 1, r);
      auto p2=point_poly_tangent(p, Q, dir, l, mid - 1);
return orientation(Q,p1.first,p2.first) == dir ? p1:p2;}
    if(!pvs) {
      if(orientation(Q,p[mid],p[l]) == dir) r=mid-1;
      else if(orientation(Q,p[l],p[r]) == dir) r=mid-1;
      else l = mid + 1; }
    if(!nxt) {
      if(orientation(Q,p[mid],p[l]) == dir) l=mid+1;
      else if(orientation(Q,p[l],p[r]) == dir) r=mid-1;
      else l = mid + 1; } }
  pair<PT, int> ret = {p[l], l};
  for(int i = l + 1; i <= r; i++) ret = orientation(Q, ret.
       first, p[i]) != dir ? make_pair(p[i], i) : ret; return
        ret; }
//(cw, ccw) tangents from a point that is outside this
     convex polygon returns indexes of the points
pair<int,int> tangents_from_point_to_polygon(VPT &p,PT Q){
  int cw=point_poly_tangent(p,Q,1,0,p.size()-1).second;
  int ccw=point_poly_tangent(p,Q,-1,0,p.size()-1).second;
  return make_pair(cw, ccw); }
// minimum distance from a point to a convex polygon
// it assumes point does not lie strictly inside the polygon
ld dist_from_point_to_polygon(vector<PT> &p, PT z) {
  ld ans = inf; int n = p.size();
  if (n <= 3) {
    for(int i = 0; i < n; i++) ans = min(ans,
         dist_from_point_to_seg(p[i], p[(i + 1) % n], z));
    return ans; }
  auto [r, l] = tangents_from_point_to_polygon(p, z);
  if(l > r) r += n;
  while (l < r) {
    int mid = (l + r) >> 1;
    ld left = dist2(p[mid % n], z), right= dist2(p[(mid + 1)
         % n], z);
    ans = min({ans, left, right});
    if(left < right) r = mid; else l = mid + 1; }
  ans = sqrt(ans);
  ans = min(ans, dist_from_point_to_seg(p[l % n], p[(l + 1)
       % n], z));
  ans = min(ans, dist_from_point_to_seg(p[l % n], p[(l - 1 +
       n) % n], z));
  return ans; }
ld dist_from_polygon_to_line(VPT &p,PT a,PT b,int top){ PT
     orth = (b - a).perp();
  if(orientation(a, b, p[0]) > 0) orth = (a - b).perp();
  int id = extreme_vertex(p, orth, top);
  if(dot(p[id] - a, orth) > 0) return 0.0;
  return dist_from_point_to_line(a, b, p[id]); }
ld dist_from_polygon_to_polygon(VPT &p1, VPT &p2){
  ld ans = inf; //NLogN
  for(int i = 0; i < p1.size(); i++)
    ans=min(ans, dist_from_point_to_polygon(p2, p1[i]));
  for(int i = 0; i < p2.size(); i++)
    ans=min(ans, dist_from_point_to_polygon(p1, p2[i]));
  return ans; }
ld maximum_dist_from_polygon_to_polygon(VPT &u, VPT &v){
  int n=u.size(), m=v.size(); ld ans = 0; //O(n)
  if(n < 3 || m < 3) {
    for(int i = 0; i < n; i++) {
    for(int j=0; j<m; j++) ans=max(ans,dist2(u[i],v[j]));
    } return sqrt(ans); }
  if(u[0].x > v[0].x) swap(n, m), swap(u, v);
  int i = 0, j = 0, step = n + m + 10;
  while (j + 1 < m && v[j].x < v[j + 1].x) j++ ;
  while (step--) {
  if(cross(u[(i+1)%n]-u[i],v[(j+1)%m]-v[j])>=0) j=(j+1)%m;
  else i = (i + 1) % n; ans = max(ans, dist2(u[i], v[j]));
  }  return sqrt(ans); }
//n polygons(not necessarily convex). points within each
     polygon must be given in CCW order. O(N^2), N total
     points
ld rat(PT a, PT b, PT p) {
  return !sign(a.x - b.x) ? (p.y - a.y) / (b.y - a.y) : (p.x
       - a.x) / (b.x - a.x); };
ld polygon_union(vector<VPT> &p) {
  int n = p.size(); ld ans=0;
  for(int i = 0; i < n; ++i) {
    for(int v = 0; v < (int)p[i].size(); ++v){
      PT a = p[i][v], b = p[i][(v + 1) % p[i].size()];
      vector<pair<ld, int>> segs;
      segs.emplace_back(0,0), segs.emplace_back(1,0);
      for(int j = 0; j < n; ++j) {
        if(i != j){
          for(size_t u = 0; u < p[j].size(); ++u) {
            PT c=p[j][u], d=p[j][(u+1)%p[j].size()];
            int sc = sign(cross(b - a, c - a)), sd = sign(
                 cross(b - a, d - a));
            if(!sc && !sd) {
              if(sign(dot(b - a, d - c)) > 0 && i > j)
                    segs.emplace_back(rat(a, b, c), 1), segs.
                         emplace_back(rat(a, b, d), -1); }
            else {
              ld sa=cross(d-c,a-c), sb=cross(d-c,b-c);
              if(sc >= 0 && sd < 0) segs.emplace_back(sa / (
                   sa - sb), 1);
              else if(sc < 0 && sd >= 0) segs.emplace_back(
                   sa / (sa - sb), -1);
            } } } }
      sort(segs.begin(), segs.end());
      ld pre=min(max(segs[0].first,(ld)0.0),(ld)1.0),now, sum
           = 0; int cnt = segs[0].second;
      for(int j = 1; j < segs.size(); ++j) {
          now=min(max(segs[j].first, (ld)0.0), (ld)1.0);
          if(!cnt) sum += now-pre; cnt += segs[j].second;
          pre = now; }
      ans += cross(a, b) * sum; } } return ans * 0.5; }
struct HP {
  PT a, b; HP() {}
  HP(PT a, PT b) : a(a), b(b) {}
  HP(const HP& rhs) : a(rhs.a), b(rhs.b) {}
  int operator < (const HP& rhs) const {
    PT p = b - a; PT q = rhs.b - rhs.a;
    int fp = (p.y < 0 || (p.y == 0 && p.x < 0));
    int fq = (q.y < 0 || (q.y == 0 && q.x < 0));
    if (fp != fq) return fp == 0;
    if (cross(p, q)) return cross(p, q) > 0;
    return cross(p, rhs.b - a) < 0; }
  PT line_line_intersection(PT a, PT b, PT c, PT d){
    b = b - a; d = c - d; c = c - a;
    return a + b * cross(c, d) / cross(b, d); }
  PT intersection(const HP &v) {
    return line_line_intersection(a,b,v.a,v.b); } };
int check(HP a, HP b, HP c) {
  return cross(a.b - a.a, b.intersection(c) - a.a) > -eps;
}
//consider half-plane of counter-clockwise side of each line
     . returns a convex polygon, a point can occur multiple
     times though. O(n log(n))
VPT half_plane_intersection(vector<HP> h) {
  sort(h.begin(), h.end()); vector<HP> tmp;
  for(int i = 0; i < h.size(); i++) {
    if(!i || cross(h[i].b-h[i].a, h[i-1].b-h[i-1].a))
      tmp.push_back(h[i]); }
  h=tmp; vector<HP> q(h.size()+10); int qh=0,qe=0;
  for(int i = 0; i < h.size(); i++) {
    while(qe-qh>1 && !check(h[i],q[qe-2],q[qe-1])) qe--;
    while(qe-qh > 1 && !check(h[i],q[qh],q[qh+1])) qh++;
    q[qe++] = h[i]; }
  while(qe-qh>2 && !check(q[qh],q[qe-2],q[qe-1])) qe--;
  while(qe-qh>2 && !check(q[qe-1],q[qh],q[qh+1])) qh++;
  vector<HP> res; VPT hull; int n = res.size();
  for(int i = qh; i < qe; i++) res.push_back(q[i]);
  if(n > 2) {
    for(int i = 0; i < n; i++)
      hull.push_back(res[i].intersection(res[(i+1)%n]));
  } return hull; }
//a and b -> strictly convex polygons of DISTINCT points
VPT minkowski_sum(VPT &a, VPT &b) {
  int n = a.size(), m = b.size(),i = 0, j = 0; VPT c;
  c.push_back(a[i] + b[j]);
  while(1){
    PT p1=a[i]+b[(j+1)% m], p2=a[(i+1)%n]+b[j];
    int t = orientation(c.back(), p1, p2);
    if(t>=0) j=(j+1)%m; if(t<=0) i=(i+1)%n, p1=p2;
    if(t == 0) p1 = a[i] + b[j]; if(p1 == c[0]) break;
    c.push_back(p1); } return c; }
ld triangle_circle_intersection(PT c,T r,PT a,PT b){
  ld sd1 = dist2(c, a), sd2 = dist2(c, b);
  if(sd1 > sd2) swap(a, b), swap(sd1, sd2);
  ld sd = dist2(a, b);
  ld d1 = sqrtl(sd1), d2 = sqrtl(sd2), d = sqrt(sd);
  ld x=abs(sd2-sd-sd1)/(2*d); ld h=sqrtl(sd1-x*x);
  if(r >= d2) return h * d / 2; ld area = 0;
  if(sd + sd1 < sd2) {
    if(r < d1) area=r*r*(acos(h/d2)-acos(h/d1))/2;
    else{
      area=r*r*(acos(h/d2)-acos(h/r))/2;
      ld y=sqrtl(r*r-h*h); area+=h*(y-x)/2; } }
  else {
    if(r<h) area=r*r*(acos(h/d2) + acos(h/d1)) / 2;
    else {
      area += r * r * (acos(h / d2) - acos(h / r)) / 2;
      ld y=sqrtl(r*r-h*h); area += h * y / 2;
      if(r < d1) {
        area += r*r*(acos(h / d1) - acos(h / r)) / 2;
        area+=h*y/2; }
      else area += h*x/2; } } return area; }
//intersection between a simple polygon and a circle
ld polygon_circle_intersection(VPT &v, PT p, T r) {
  int n=v.size(); ld ans=0.00; PT org = {0, 0};
  for(int i = 0; i < n; i++) {
    int x = orientation(p, v[i], v[(i + 1) % n]);
    if(x == 0) continue;
    ld area = triangle_circle_intersection(org, r, v[i] - p,
         v[(i + 1) % n] - p); if (x < 0) ans -= area;
    else ans += area; } return ans * 0.5; }
//circle of radius r that contains as many points as
     possible. O(n^2 log n);
int maximum_circle_cover(VPT p, T r, circle &c) {
  int n = p.size(),ans = 0,id = 0; ld th = 0;
  for(int i = 0; i < n; ++i) {
    vector<pair<ld, int>> events = {{-PI, +1},{PI, -1}};
    for(int j = 0; j < n; ++j) {
      if(j == i) continue;
      ld d = dist(p[i], p[j]); if(d > r * 2) continue;
      ld dir = (p[j]-p[i]).arg(); ld ang = acos(d/2/r);
      ld st=dir-ang, ed=dir+ang; if(st>PI) st -= PI*2;
      if(st <= -PI) st += PI*2; if(ed > PI) ed -= PI*2;
      if(ed<=-PI) ed += PI*2;events.push_back({st, +1});
      events.push_back({ed, -1});
      if(st > ed) {
events.push_back({-PI,1});events.push_back({PI,-1});} }
    sort(events.begin(), events.end()); int cnt = 0;
    for(auto &&e: events) {
      cnt += e.second;
      if(cnt>ans){ ans=cnt; id=i; th=e.first; } } }
  PT w=PT(p[id].x+r*cos(th), p[id].y + r * sin(th));
  c = circle(w, r); return ans; }
ld maximum_inscribed_circle(VPT p){ //returns radius
  int n=p.size(); if(n<3) return 0; ld l=0, r=20000;
  while(r - l > eps){ vector<HP> h;
    ld mid = (l + r) * 0.5; const int L = 1e9;
    h.push_back(HP(PT(-L, -L), PT(L, -L)));
    h.push_back(HP(PT(L, -L), PT(L, L)));
    h.push_back(HP(PT(L, L), PT(-L, L)));
    h.push_back(HP(PT(-L, L), PT(-L, -L)));
    for(int i = 0; i < n; i++) {
      PT z=(p[(i+1)%n]-p[i]).perp(); z=z.truncate(mid);
      PT y = p[i] + z, q = p[(i + 1) % n] + z;
      h.push_back(HP(p[i] + z, p[(i + 1) % n] + z)); }
    VPT nw = half_plane_intersection(h);
    if(!nw.empty()) l = mid; else r=mid; } return l;}
```

4 Number Theory

4.1 BigMod

```
ll BigMod(ll a,ll p, ll M=MOD){
  if(!p) return 1%M;
  ll x=Biod(a,p/2,M); x=(x*x)%M;
  if(p&1) x=(x*a)%M; return x; }

ll Big_Mod(ll a, ll p, ll M=MOD){
  ll result=1;
  while(p>0){ if(p&1){
      result*=a; if(result>M) result%=M; }
    p>>=1; a*=a; if(a>M) a%=M;
  } return result%M; }
```

4.2 BitInt

```
const int base = 1000000000; const int base_digits = 9;
struct bigint {
  vector<int> a; int sign; bigint() : sign(1) {}
  bigint(ll v) { *this = v;}
  bigint(const string &s) { read(s);}
  void operator=(const bigint &v) { sign = v.sign; a = v.a;}
  void operator=(ll v){ sign = 1;
    if(v < 0) sign = -1, v = -v;
    for(; v > 0; v = v / base) a.push_back(v % base); }
  bigint operator+(const bigint &v) const {
    if(sign == v.sign) { bigint res = v;
      for(int i = 0, carry = 0; i < (int) max(a.size(), v.a.
           size()) || carry; ++i) {
        if(i == (int) res.a.size())
          res.a.push_back(0);
        res.a[i] += carry+(i < (int) a.size() ? a[i] : 0);
        carry = res.a[i] >= base;
        if (carry) res.a[i] -= base; } return res; }
    return *this - (-v); }
  bigint operator-(const bigint &v) const {
    if(sign == v.sign) {
      if(abs() >= v.abs()) {
        bigint res = *this;
        for(int i=0,carry=0;i<(int)v.a.size() || carry;++i){
          res.a[i] -= carry+(i<(int)v.a.size()? v.a[i] : 0);
          carry = res.a[i] < 0;
          if(carry) res.a[i] += base; }
        res.trim(); return res; }
      return -(v - *this); } return *this + (-v); }
  void operator*=(int v) {
    if(v < 0) sign = -sign, v = -v;
    for(int i=0,carry=0; i<(int) a.size() || carry; ++i) {
      if(i == (int) a.size()) a.push_back(0);
      ll cur = a[i]*(ll)v+carry; carry = (int) (cur / base);
      a[i] = (int) (cur % base); } trim(); }
  bigint operator*(int v) const {
    bigint res = *this; res *= v; return res; }
  friend pair<bigint, bigint> divmod(const bigint &a1, const
       bigint &b1) {
    int norm = base / (b1.a.back() + 1);
    bigint a = a1.abs() * norm; bigint b = b1.abs() * norm;
    bigint q, r; q.a.resize(a.a.size());
    for(int i = a.a.size() - 1; i >= 0; i--) {
      r *= base; r += a.a[i];
      int s1=r.a.size() <= b.a.size() ? 0 : r.a[b.a.size()];
  int s2=r.a.size()<=b.a.size()-1 ? 0:r.a[b.a.size()-1];
  int d = ((ll)base * s1 + s2) / b.a.back(); r -= b * d;
  while(r<0) r += b, --d;
  q.a[i] = d; }
 q.sign = a1.sign * b1.sign; r.sign = a1.sign;
 q.trim(); r.trim(); return make_pair(q, r / norm); }
bigint operator/(const bigint &v) const {
 return divmod(*this, v).first; }
bigint operator%(const bigint &v) const {
 return divmod(*this, v).second; }
void operator/=(int v) {
 if(v < 0) sign = -sign, v = -v;
 for(int i = (int)a.size()-1, rem = 0; i >= 0; --i){
  ll cur = a[i] + rem * (ll) base;
  a[i] = (int) (cur / v); rem = (int) (cur % v);
 }  trim(); }
bigint operator/(int v) const {
 bigint res = *this; res /= v; return res; }
int operator%(int v) const {
 if (v < 0) v = -v; int m = 0;
 for(int i = a.size() - 1; i >= 0; --i)
  m = (a[i] + m * (ll) base) % v;
 return m * sign; }
void operator+=(const bigint &v) { *this = *this + v; }
void operator-=(const bigint &v) { *this = *this - v; }
void operator*=(const bigint &v) { *this = *this * v; }
void operator/=(const bigint &v) { *this = *this / v; }
bool operator<(const bigint &v) const {
 if(sign != v.sign) return sign < v.sign;
 if(a.size() != v.a.size())
  return a.size() * sign < v.a.size() * v.sign;
 for(int i = a.size() - 1; i >= 0; i--)
  if(a[i] != v.a[i]) return a[i] * sign < v.a[i] * sign;
 return false; }
bool operator>(const bigint &v) const { return v < *this;}
bool operator<=(const bigint &v) const {return !(v<*this);}
bool operator>=(const bigint &v) const {return !(*this<v);}
bool operator==(const bigint &v) const {
 return !(*this < v) && !(v < *this);}
bool operator!=(const bigint &v) const {
 return *this < v || v < *this; }
void trim() {
 while (!a.empty() && !a.back()) a.pop_back();
 if(a.empty()) sign = 1; }
bool isZero() const {
 return a.empty() || (a.size() == 1 && !a[0]); }
bigint operator-() const {
 bigint res = *this; res.sign = -sign; return res;}
bigint abs() const {
 bigint res = *this; res.sign *= res.sign;
 return res; }
ll longValue() const {
 ll res = 0;
 for(int i = a.size() - 1; i >= 0; i--)
  res = res * base + a[i]; return res * sign; }
friend bigint gcd(const bigint &a, const bigint &b){
 return b.isZero() ? a : gcd(b, a % b); }
friend bigint lcm(const bigint &a, const bigint &b){
 return a / gcd(a, b) * b; }
void read(const string &s) {
 sign = 1; a.clear(); int pos = 0;
 while(pos<s.size() && (s[pos] == '-' || s[pos] == '+')){
  if(s[pos] == '-') sign = -sign; ++pos; }
 for(int i = s.size() - 1; i >= pos; i -= base_digits){
  int x = 0;
  for(int j=max(pos, i - base_digits + 1); j <= i; j++)
    x = x * 10 + s[j] - '0';
  a.push_back(x); } trim(); }
friend istream& operator>>(istream &stream, bigint &v) {
 string s; stream >> s;
 v.read(s); return stream; }
friend ostream& operator<<(ostream &stream, const bigint &
   v) {
 if (v.sign == -1) stream << '-';
 stream << (v.a.empty() ? 0 : v.a.back());
 for (int i = (int) v.a.size() - 2; i >= 0; --i)
  stream << setw(base_digits) << setfill('0') << v.a[i];
 return stream; }
static vector<int> convert_base(const vector<int> &a, int
   old_digits, int new_digits) {
 vector<ll> p(max(old_digits, new_digits) + 1);
 p[0] = 1;
 for(int i = 1; i < (int) p.size(); i++)
  p[i] = p[i - 1] * 10;
 vector<int> res; ll cur = 0; int cur_digits = 0;
 for(int i = 0; i < (int) a.size(); i++) {
  cur += a[i] * p[cur_digits];
  cur_digits += old_digits;
  while (cur_digits >= new_digits) {
   res.push_back(int(cur % p[new_digits]));
   cur /= p[new_digits]; cur_digits -= new_digits;
  }  }
 res.push_back((int) cur);
 while (!res.empty() && !res.back()) res.pop_back();
 return res; }
typedef vector<ll> vll;
static vll karatsubaMultiply(const vll &a, const vll &b){
 int n = a.size(); vll res(n + n);
 if(n <= 32) {
  for(int i = 0; i < n; i++)
     for(int j = 0; j < n; j++)
       res[i + j] += a[i] * b[j];
   return res; }
  int k = n >> 1;
  vll a1(a.begin(), a.begin() + k);
  vll a2(a.begin() + k, a.end());
  vll b1(b.begin(), b.begin() + k);
  vll b2(b.begin() + k, b.end());
  vll a1b1 = karatsubaMultiply(a1, b1);
  vll a2b2 = karatsubaMultiply(a2, b2);
  for(int i = 0; i < k; i++) a2[i] += a1[i];
  for(int i = 0; i < k; i++) b2[i] += b1[i];
  vll r = karatsubaMultiply(a2, b2);
  for(int i = 0; i < a1b1.size(); i++) r[i] -= a1b1[i];
  for(int i = 0; i < a2b2.size(); i++) r[i] -= a2b2[i];
  for(int i = 0; i<(int)r.size(); i++) res[i+k] += r[i];
  for(int i = 0; i<a1b1.size(); i++) res[i] += a1b1[i];
  for(int i=0; i<a2b2.size(); i++) res[i+n] += a2b2[i];
  return res; }
 bigint operator*(const bigint &v) const {
  vector<int> a6 = convert_base(this->a, base_digits, 6);
  vector<int> b6 = convert_base(v.a, base_digits, 6);
  vll a(a6.begin(), a6.end()); vll b(b6.begin(),b6.end());
  while(a.size() < b.size()) a.push_back(0);
  while(b.size() < a.size()) b.push_back(0);
  while(a.size() & (a.size() - 1))
   a.push_back(0), b.push_back(0);
  vll c = karatsubaMultiply(a, b);
  bigint res; res.sign = sign * v.sign;
  for(int i = 0, carry = 0; i < (int)c.size(); i++) {
   ll cur = c[i] + carry;
   res.a.push_back((int) (cur % 1000000));
   carry = (int) (cur / 1000000); }
  res.a = convert_base(res.a, 6, base_digits);
  res.trim(); return res; } };
main(){ bigint a = bigint("1");}
```

4.3 CRT

```
#define __int128 lll
/*A CRT solver which works even when moduli are not pairwise
coprime Call CRT(k) to get {x, N} pair, where x is the
unique solution modulo N. O(n*Log L) L=LCM(p1,p2,....)
Assumptions: 1. LCM of all mods will fit into long long. */
pll CRT(ll n){
 ll r1,r2,p1,p2,x,y,ans,g,mod; r1=r[0],p1=p[0];
 for(int i=1; i<n; i++){
  r2=r[i],p2=p[i]; g=__gcd(p1,p2);
  if(r1%g != r2%g) return {-1, -1}; //no solution
```

```
    EGCD(p1/g, p2/g, x, y); mod=p1/g*p2;
ans=((lll)r1*(p2/g)%mod*y%mod+(lll)r2*(p1/g)%mod*x%mod)%mod;
    r1=ans; if(r1<0) r1+=mod; p1=mod;
  }  return {r1,mod}; }
```

4.4 Catalan and Derangmenet

```
D(n) = (n-1)*( D(n-1) + D(n-2) ); D(0)=D(2)=1, D(1)=0;
C(n) = nCr(2n, n) - nCr(2n, n+1) = nCr(2n,n)/(n+1)
```

4.5 Discrete Log

```
//returns minimum integer x such that a^x = b (mod m)
// a and m are co-prime, O(sqrt(m))
int discrete_log(int a, int b, int m) {
  int n=(int)sqrt(m+.0)+1, pw=1;
  for(int i=0; i<n; ++i) pw=(1LL*pw*a)%m;
  gp_hash_table<int, int> vals;
  for(int p=1, cur=pw; p<=n; ++p){
    if(!vals[cur]) vals[cur] = p;
    cur=(1LL*cur*pw)%m; }
  int ans = inf;
  for(int q=0, cur=b; q<=n; ++q){
    if(vals.find(cur) != vals.end()){
      ll nw=1LL*vals[cur]*n-q;
      if(nw<ans) ans = nw; }
    cur = (1LL * cur * a) % m; }
  if(ans == inf) ans = -1; return ans; }
ll inverse(ll a, ll m) {
  ll x, y; ll g = e_gcd(a, m, x, y);
  if (g != 1) return -1;return (x % m + m) % m;}
// discrete log but a and m may not be co-prime
int discrete_log_noncoprime(int a, int b, int m) {
  if(m == 1 || b == 1) return 0;
  if(__gcd(a, m) == 1) return discrete_log(a, b, m);
  int g = __gcd(a, m); if(b % g != 0) return -1;
  int p = inverse(a / g, m / g);
  int nw = discrete_log_noncoprime(a, 1LL * b / g * p % (m /
        g), m / g);
  if (nw == -1) return -1; return nw + 1; }
```

4.6 Discrete and Primitive Root

```
/*Finds the primitive root modulo p. g is a primitive root
mod p if and only if for any integer a such that gcd(a,p)
=1, there exists an integer k such that: g^k = a(mod p)*/
int PrimitiveRoot (int p) {
  vector<int> fact;
  int phi = p-1; // if p is prime
  int n = phi;
  for(int i=2; i*i<=n; ++i){
    if(n%i == 0){
      fact.push_back (i); while(n % i == 0) n /= i; }}
  if(n>1) fact.push_back (n);
  for(int res=2; res<=p; ++res){
    bool ok = true;
    for(size_t i=0; i<fact.size() && ok; ++i)
      ok &= BigMod (res, phi/fact[i], p) != 1;
    if (ok) return res;
  }  return -1; }
//find all numbers x such that x^k = a (mod n)
void printDiscreteRoot(int k, int a, int n){
  if(a==0){ cout<<"1\n0\n"; return; }
  int g=PrimitiveRoot(n);
  int phi=n-1; //if n is not a prime calculate phi(n)
  int sq=(int) sqrt(n+0.0)+1;
  vector<pair<int,int> > dec (sq);
  for(int i=1; i<=sq; ++i)
    dec[i-1]={BigMod(g, 1LL*i*sq%phi*k%phi,n), i};
  sort(dec.begin(), dec.end());
  int any_ans = -1;
  for(int i=0; i<sq; ++i) {
    int my=BigMod(g, 1LL*i*k%phi, n)*1LL*a%n;
 auto it=lower_bound(dec.begin(),dec.end(),make_pair(my,0));
    if(it != dec.end() && it->first == my){
      any_ans = it->second * sq - i; break; } }
  if(any_ans==-1){ cout<<"0\n"; return ; }
  // all possible answers
  int delta = (n-1) / __gcd(k, n-1);
  vector<int> ans;
  for(int cur=any_ans%delta; cur<n-1; cur+=delta)
    ans.push_back(powmod(g,cur,n));
  sort(ans.begin(), ans.end());
  cout<<(int)ans.size()<<"\n";
  for(int answer : ans) cout<<answer<<" "; }
```

4.7 E$_g cd$

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

4.8 FFT

```
using cd = complex<double>;
const double PI = acos(-1);
void fft(vector<cd> &a,bool invert){
  int n = 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){
    double ang=2*PI/len*(invert? -1:1);
    cd wlen(cos(ang), sin(ang));
    for(int i=0; i<n; i+=len) {
      cd w(1);
      for(int j=0; j<len/2; j++){
        cd u=a[i+j], v=a[i+j+len/2]*w;
        a[i+j] = u + v;
        a[i+j+len/2] = u - v;
        w *= wlen; }  }  }
  if(invert) {
      for(cd & x : a) x /= n; } }
void for_multiplying_two_long_numbers(vector<int> &result){
  int carry=0,n=result.size();
  for(int i=0; i<n; i++){
    result[i]+=carry; carry=result[i]/10;
    result[i] %= 10; } }
vector<int> multiply(vector<int> &a, vector<int> &b){
  vector<cd> fa(a.begin(),a.end()),fb(b.begin(),b.end());
  int n = 1;
  while(n < a.size() + b.size()) n <<= 1;
  fa.resize(n); fb.resize(n);
  fft(fa, false); fft(fb, false);
  for(int i=0; i<n; i++) fa[i] *= fb[i];
  fft(fa, true);
  vector<int> result(n);
  for(int i=0; i<n; i++) result[i]=round(fa[i].real());
  //for_multiplying_two_long_numbers(result);
  return result; }
```

4.9 Factorial Mod

```
//n! % p =?,O(plogp n).
int factmod(int n, int p){
  int res = 1;
  while(n>1){
    res=(res*((n/p)%2? p-1:1))%p;
    for(int i=2; i<=n%p; ++i)
      res=(res*i)%p;
    n/=p;
  }  return res % p; }
```

4.10 Fibonacci

```
//Fast doubling {n,n+1}
pii fib(int n){
  if(n==0) return {0,1}; pii p=fib(n>>1);
  int a=p.F*(2*p.S-p.F); int b=p.F*p.F+p.S*p.S;
  if(n&1) return {b, b+a}; else return {a,b}; }
```

4.11 GCD Counting

```
ll cnt[MAX];
ll nC4(ll n){ return n*(n-1)*(n-2)*(n-3)/24LL; }
solve(){
  memset(cnt, 0, sizeof cnt);
  for(i=0; i<n; i++) { cin>>x; cnt[x]++; }
  for(i=1; i<MAX; i++){
    for(j=i+i; j<MAX; j+=i)
      cnt[i]+=cnt[j];
    //cnt[i]=total number such that gcd=i*k,k=1,2,3...
  }
  for(i=1; i<MAX; i++) {
      cnt[i]=nC4(cnt[i]);
    //cnt[i]=total number of group of 4 element s.t. gcd=i*k
  }
  for(i=MAX-1; i>0; i--){
    for(j=i+i; j<MAX; j+=i)
      cnt[i]-=cnt[j];//multiple bad dilam
    //cnt[i]=total number of group of 4 element s.t. gcd=i
  } }
//For Continuous sub-array
map<int, ll> cnt;//cnt[i] = number sub-array s.t. gcd=i
void CountGcd(vector<int>& v) {
  map<int, int> divisors, nextDivisors;
  int n=(int)v.size();
  for(int i=0; i<n; i++){
    nextDivisors.clear();
    for(auto &p : divisors)
      nextDivisors[__gcd(p.first, v[i])] += p.second;
    nextDivisors[v[i]]++;
    swap(nextDivisors, divisors);
    for (auto &p : divisors) cnt[p.first] += p.second; } }
```

4.12 Linear Diophantine Eqn

```
void shift_solution(int &x,int &y,int a,int b,int cnt){
    x+=cnt*b; y-=cnt*a;
}
void find_positive_solution(int a,int b,int &x,int &y,int g)
     {
    a/=g; b/=g;
    int mov;
    if(x<0){
        mov=(0-x+b-1)/b;
        shift_solution(x,y,a,b,mov);
    }
    if(y<0) {
        mov=(-y)/a;
        if((-y)%a) mov++;
        shift_solution(x,y,a,b,-mov);
    }
}
bool find_any_solution(int a,int b,int c,int &x0,int &y0,int
      &g){
    g=egcd(abs(a),abs(b),x0,y0);
    if(c%g) return 0;
    x0*=c/g; y0*=c/g;
    if(a<0) x0*=-1;
    if(b<0) y0*=-1;
    //find_positive_solution(a,b,x0,y0,g);
    return 1;
}
int find_all_solutions(int a, int b, int c, int minX, int
     maxX, int minY, int maxY){
    if(!a && !b){
        if(c) return 0LL;
        return (maxX-minX+1LL)*(maxY-minY+1LL);
    }
    if(!a){
        if(c%b) return 0LL;
        int y=c/b;
        if(minY<=y && y<=maxY) return (maxX-minX+1LL);
        return 0LL;
    }
    if(!b){
        if(c%a) return 0LL;
        int x=c/a;
        if(minX<=x && x<=maxX) return (maxY-minY+1LL);
        return 0LL;
    }
    int x, y, g;
    if(!find_any_solution(a,b,c,x,y,g))
        return 0;
    a/=g; b/=g;
    int sign_a = a > 0 ? +1 : -1;
    int sign_b = b > 0 ? +1 : -1;
    shift_solution(x, y, a, b,(minX-x)/b);
    if(x<minX)
        shift_solution(x, y, a, b, sign_b);
    if(x>maxX) return 0;
    int lx1 = x;
    shift_solution(x, y, a, b, (maxX-x)/b);
    if(x>maxX)
        shift_solution(x, y, a, b, -sign_b);
    int rx1 = x;
    shift_solution(x, y, a, b, -(minY-y)/a);
    if(y<minY)
        shift_solution(x, y, a, b, -sign_a);
    if(y>maxY) return 0;
    int lx2 = x;
    shift_solution(x, y, a, b, -(maxY-y)/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;
}

/** LD Less or Equal:
    ax+by <= c **/
ll count(ll a, ll b, ll c){
    if(b>a) swap(a,b);
    ll m = c/a;
    if(a==b) return m*(m-1)/2LL;
    ll k = (a-1)/b, h = (c - a*m)/b;
    return m*(m-1)/2*k+m*h+count(b,a-b*k, c - b*(k*m+h));
}
```

4.13 NOD SOD POD

```
{p^a} in prime factorization
nod=nod*(a+1)%MOD;
sod=sod*(BigMod(p,a+1)-1+MOD)%MOD*inverse(p-1,MOD)%MOD;
pod=BigMod(pod,a+1)*BigMod(BigMod(p,(a*(a+1)/2)),tnod)%MOD;
tnod=tnod*(a+1)%(MOD-1);

//O(N^1/3),need prime till n^1/3
int CB(int x) {return x*x*x;}
int NOD(int n){
  int cnt=1;
  for(int i=0; CB(prime[i])<=n; i++){
    int c=1;
    while(n&prime[i]==0) { c++; n/=prime[i]; }
    cnt*=c; }
  if(isPrime[n]) cnt*=2; else if(isPrimeSq[n]) cnt*=3;
  else if(n!=1) cnt*=4; return cnt; }
//O(sqrt(n)), NOD(1)+..+NOD(n)
ll CNOD(ll n) {
  ll i,cnt=0,sq=sqrt(n);
  for(i=1; i<=sq; i++) cnt+=(n/i)-i;
  cnt*=2; cnt+=sq; return cnt; }
//O(sqrt(n))
ll CSOD(ll n){
  ll i,j,ans=0LL;
  for(i=1; i*i<=n; i++){
    j=n/i; ans+=(i+j)*(j-i+1)/2LL;
    ans+=i*(j-i); } return ans; }
//O(NLogN)
void SOD(){
  for(i=1; i<MAX; i++)
    for(j=i; j<MAX; j+=i) sod[j]+=i; }
```

4.14 Pollard RHO and Miller Rabin

```
using ll = uint64_t; using u128 = __uint128_t;
bool isComposite(ll n,ll a,ll d,int s){
  ll x=BigMod(a, d, n);
  if(x==1LL || x==n-1) return 0;
  for(int r=1; r<s; r++){
    x = ( (u128)x*x)%n;
    if(x==n-1) return 0; } return 1; }
bool MillerRabin(ll n){
  int s=0; ll d=n-1;
  while((d&1) == 0){ d >>= 1; s++; }
  for(ll a : {2,3,5,7,11,13,17,19,23,29,31,37}){
    if(n==a) return 1;
    if(isComposite(n, a, d, s)) return 0;
  } return 1; }
bool isPrime(ll n){
  if(n==2LL || n==3LL) return 1;
  if(n<5LL || n%2==0 || n%3==0) return 0;
  return MillerRabin(n); }
int pollard_rho(int n){
  if(n%2==0) return 2;
  int x = rand()%n+1; int c = rand()%n+1;
  int y=x, g=1;
  while(g==1){
    x=((x*x)%n+c)%n; y=((y*y)%n+c)%n;
    y=((y*y)%n+c)%n; g= __gcd(abs(x-y),n);
  } return g; }
```

```
vector<int>fact;
//O(N^1/4)
void factorize(int n){ if(n==1) return;
  if(isPrime(n)) { fact.push_back(n); return;}
  int divisor=pollard_rho(n);
  factorize(divisor);factorize(n/divisor); }
```

4.15 Random Staff

```
number theory stuff and tricks:
every prime number modulo 4 = 1 can be written as sum of two
squares. beizuts identity, if gcd(x1,x2,x3,....xn) = g, then
(a1x1+12x2+13x3+....anxn) = k*g (for any k>=1) where a1,a2,
a3 are integers(might be negative or zero)
*** going from left to right taking prefix gcd, after we add
a new number(different from any number seen sofar) gcd will
becrease by at least a prime factor... so we can have at
most 20/30 element before we hit gcd = 1;

A number is Fibonacci iff one or both of (5*n*n+4) or
(5*n*n-4) is a perfect square. (n - 1)! = (n-1)=
 1 (mod n) exactly when n is a prime number.

Pythagorean Triple(a,b,c):
 a = m*m - n*n
 b = 2*m*n
 c = m*m + n*n
 for m>n>0

gcd(a1+d, a2+d, a3+d,...)=gcd(a1+d, a2-a1, a3-a1, ...)
gcd(pow(n,a)-1, pow(n,b)-1)=pow(n, gcd(a,b))-1
gcd(a,lcm(b,c))=lcm(gcd(a,b),gcd(a,c))
phi(lcm(a,b))*phi(gcd(a,b))=phi(a)*phi(b)

x1+x2+...+xk=n, xi>=0 : NOS = nCr(n+k-1, k-1)
x1+x2+...+xk=n, xi>=ai : NOS = nCr(n+k-1-(sum of ai), k-1)
```

4.16 Sieve

```
//Mem Opt
int marked[MAX/64 + 2];
#define isOn(x) (marked[x/64] & (1<<((x%64)/2)))
#define setOn(x) marked[x/64] |= (1<<((x%64)/2))

//segmented
start=max(prime[i]*prime[i], ((low+prime[i]-1)/prime[i])*
     prime[i]);
for(j=start; j<=high; j+=prime[i])
   mark[j-low]=1;
```

4.17 Stirling Numbers

```
Stirling numbers 1st kind:
S(n,k)= kS ( n1 ,k)+S( n1 , k1 )
Base case:
S(0,0)=1,S(n,0)=S(0,n)=
S(n,2)=2^( n1 ) 1

Stirling numbers 2nd kind: (put r objects in n shelves)
Base cases:
S(r, 0) = 0
S(r, 1) = (r   1)!
S(r, r) = 1.
S(r, r -1) = rC2
Transition:
S(r,n) = S( r1 , n1 )+( r1 ) * S( r1 ,n)
```

4.18 mat$_{e}xpo$

```
struct MAT{ll n,m;vector<vector<ll>> v;MAT(){ }
 MAT(ll _n,ll _m){n=_n;m=_m;
 v.assign(n,vector<ll>(m,0));}
 MAT(vector<vector<ll>> a){n=a.size();
 m=n?a[0].size():0;v=a;}
 inline void make_unit(){assert(n==m);
  for(ll i=0;i<n;i++)for(ll j=0;j<n;j++)
  v[i][j]=(i==j);}
 inline MAT operator + (const MAT &b){
  assert(n==b.n and m==b.m);MAT ans(n,m);
  for(ll i=0;i<n;i++)for(ll j=0;j<m;j++)
  ans.v[i][j]=(v[i][j]+b.v[i][j])%mod;return ans;}
 inline MAT operator - (const MAT &b){
  assert(n==b.n and m==b.m);MAT ans(n,m);
  for(ll i=0;i<n;i++)for(ll j=0;j<m;j++)
   ans.v[i][j]=(v[i][j]-b.v[i][j]+mod)%mod;
  return ans;}
 inline MAT operator * (const MAT &b){
  assert(m==b.n);MAT ans(n,b.m);
  for(ll i=0;i<n;i++)for(ll j=0;j<b.m;j++)
   for(ll k=0;k<m;k++)ans.v[i][j]=
   (ans.v[i][j]+(v[i][k]*b.v[k][j])%mod)%mod;
  return ans;}
 inline MAT expo (ll k){
  assert(n==m);MAT tmp=v,ans(n,n);ans.make_unit();
  while(k){if(k&1) ans=ans*tmp;tmp=tmp*tmp;k>>=1;}
  return ans;}
 inline bool operator == (const MAT &b){
  return v==b.v;}
 inline bool operator != (const MAT &b){
  return v!=b.v;}
};
```

4.19 mobius

```
// returns mu[x] O(logn)
ll mu(ll x){ll cnt=0;while(x>1){ll cur=0,d=spf[x];
    while(x%d==0){x/=d;cur++;if(cur>1) return 0;}
    cnt++;}if(cnt&1) return -1;else return 1;}
// from 1 to n
void mobius(){mu[1] = 1;for(ll i=2;i<N;i++){
    mu[i]--;for(ll j=2*i;j<N;j+=i)mu[j]-=mu[i];}}
```

4.20 nCr MOD p

```
//nPr: (n,0)=1, (n,r)=n*(n-1, r-1);
//nCr: (n,0)=1,(n,1)=n,(n,r)=(n-1,r)+(n-1,r-1)
//Circular permutation: nPr/r

ll fact[MAX],inv[MAX],invFact[MAX];
ll nCr(ll n, ll r){ if(r>n) return 0;
  return fact[n]*invFact[r]%MOD*invFact[n-r]%MOD; }
void PreCalculaton(){ fact[0]=1;
  for(i=1; i<MAX; ++i) fact[i]=i*fact[i-1]%MOD;
  inv[1]=1;
  for(i=2; i<MAX; ++i) ///must be MAX<MOD
      inv[i]=((MOD-MOD/i)*inv[MOD%i])%MOD;
  invFact[0]=1;
  for(i=1; i<MAX; ++i)
    invFact[i]=(invFact[i-1]*inv[i])%MOD; }
//nCr_mod_p_by_lucas_for_small_p._O(LogN)
ll Lucas(ll n, ll m){
  if(n==0 && m==0) return 1ll;
  ll ni = n % MOD; ll mi = m % MOD;
  if(mi>ni) return 0;
  return (Lucas(n/MOD, m/MOD)*nCr(ni, mi))%MOD; }
ll nCr_by_lucas(ll n, ll r){ return Lucas(n, r); }

//nCr_mod_M, M_is_not_prime
const int N = 142858; //need primes <=M (nCr%M)
int spf[N]; vector<int> primes;
void sieve(){
  for(int i = 2; i < N; i++){
    if(spf[i] == 0) spf[i] = i, primes.push_back(i);
    int sz = primes.size();
    for(int j = 0; j < sz && i * primes[j] < N && primes[j]
         <= spf[i]; j++)
      spf[i * primes[j]] = primes[j]; } }

// returns n! % mod, mod = multiple of p
// O(mod * log(n))
int factmod(ll n, int p, const int mod){
  vector<int> f(mod + 1); f[0] = 1 % mod;
  for(int i = 1; i <= mod; i++){
    if (i % p) f[i] = 1LL * f[i - 1] * i % mod;
    else f[i] = f[i - 1]; }
  int ans = 1 % mod;
  while (n > 1) {
    ans = 1LL * ans * f[n % mod] % mod;
    ans = 1LL*ans*BigMod(f[mod], n/mod, mod)%mod;
    n /= p;
  } return ans; }
ll multiplicity(ll n, int p){
  ll ans = 0;
  while (n) { n /= p; ans += n;} return ans;}
// C(n, r) modulo p^k
// O(p^k log n)
int ncr(ll n, ll r, int p, int k){
  if(n < r || r < 0) return 0;
  int mod = 1;
  for (int i = 0; i < k; i++) { mod *= p; }
  ll t = multiplicity(n, p) - multiplicity(r, p) -
      multiplicity(n - r, p);
  if(t >= k) return 0;
  int ans = 1LL * factmod(n, p, mod) * inverse(factmod(r, p,
       mod), mod) % mod * inverse(factmod(n - r, p, mod),
      mod) % mod;
  ans = 1LL * ans * BigMod(p, t, mod) % mod;
  return ans;
}
pair<ll, ll> CRT(ll a1, ll m1, ll a2, ll m2) {
  ll p, q;
  ll g = e_gcd(m1, m2, p, q);
  if(a1 % g != a2 % g) return make_pair(0, -1);
  ll m = m1 / g * m2;
  p = (p % m + m) % m; q = (q % m + m) % m;
  return make_pair((p * a2 % m * (m1 / g) % m + q * a1 % m *
       (m2 / g) % m) % m, m); }
// O(m log(n) log(m))
int ncr(ll n, ll r, int m) {
  if(n < r || r < 0) return 0;
  pair<ll, ll> ans({0, 1});
  while (m > 1) {
    int p = spf[m], k = 0, cur = 1;
    while (m % p == 0) {
      m /= p; cur *= p; ++k; }
    ans=CRT(ans.first, ans.second, ncr(n,r,p,k),cur);
  } return ans.first; }
```

4.21 phi and sieve(linear)

```
int lp[MAX+1],phi[MAX+1];
vector<int> prm;
void pre() {
  phi[1]=1;
  for(int i=2; i<=MAX; ++i){
    if(lp[i] == 0){
      lp[i]=i,phi[i]=i-1; prm.push_back(i);
    } else {
      if(lp[i]==lp[i/lp[i]]) phi[i]=phi[i/lp[i]]*lp[i];
      else phi[i]=phi[i/lp[i]]* (lp[i]-1); }
    for(int j=0; j<(int)prm.size()&&prm[j]<=lp[i]&& i*prm[j
        ]<=MAX; ++j)
      lp[i*prm[j]] = prm[j]; } }

//Characteristics of phi:
phi=n*MULof(1-1/p)
gcd(i,N)=g --> phi(N/g);
gcd(x,y)=g, 1<=x,y<=N --> sumPhi(N/g);(cumSum->sumPhi)
phi(lcm(a,b))*phi(gcd(a,b))=phi(a)*phi(b)
sum of co-prime   phi(n)*n/2;
sum of gcd(i,N), for i= 1 to N --> sum of (d*phi(n/d)) for
     all d|N;
sum of LCM(i,N) for i=1 to N --> (N/2) *( x +1) where x=sum
     of (d* phi(d) ) for all d|N;
sum of gcd(k-1,N), for k=1 to N and gcd(k,N)=1 --> phi(N)*
     NOD(N)

//Preity
generalization of Eulers theorem:
x^n = x^( phi(m)+n mod phi(m)) mod m
LCM sum: 2(SUM-n)=n*(sum of d*phi(d),s.t d|n and d is npt 1)

int phi(int n) { int ret = n;
  for(int i = 2;i*i<=n;i++){
    if(n%i == 0){
      ret-=ret/i; while(n%i == 0) n/=i;
    } } if(n>1) ret-=ret/n; return ret; }
ll Euler_Phi(ll n) {
  ll i,result=n;
  for(i=0;i<prime.size()&&prime[i]*prime[i]<=n;i++){
    if(n%prime[i]==0){
      while(n%prime[i]==0) n/=prime[i];
      result=(result/prime[i])*(prime[i]-1); } }
  if(n>1) result=(result/n)*(n-1); return result; }
void Euler() {
  for(int i=0; i<MAX; i++) phi[i]=i;
  for(i=2; i<MAX; i++){
    if(phi[i]==i){ phi[i]=i-1;
      for(j=i+i; j<MAX; j+=i)
        phi[j]=phi[j]/i*(i-1); } } }
```

5 Random

5.1 Bit Basic

```
a+b=a^b+2*(a&b); a^b=(a|b)&(~a|~b)
#define setbit(n) (n&-n)
#define bitOn(n,i) ((1LL<<i)|n)
#define bitOn2(n,i,j) ((((1LL<<(j-i+1))-1)<<i)|n)
#define bitOff(n,i) (~(1LL<<i)&n)
#define bitOff2(n,i,j) (~(((1LL<<(j-i+1))-1)<<i)&n)
#define bitToggle(n,i) ((1LL<<i)^n)
#define bitToggle2(n,i,j) ((((1LL<<(j-i+1))-1)<<i)^n)
#define bitOnAll(n) (n=-1)
#define bitOffAll(n) (n=0)
#define getbit(n,i) ((n>>i)&1)
#define last1cng(n) (n)&((n)-1)
#define last0cng(n) ~(last1cng(~(n)))
//all the subsets of the mask
void subSet(ll mask){
  for(ll i = mask; i>0; i = (i-1)&mask)
    cout<<bitset<8>(i)<<endl; }
```

5.2 GP Hash Table

```
//gp_hash_table<key, int, chash> table;
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
mt19937 rng(chrono::steady_clock::now().time_since_epoch().
     count());
struct chash {
  const uint64_t C = 2713661325311;
  const uint32_t RANDOM = chrono::steady_clock::now().
       time_since_epoch().count();
  size_t operator()(uint64_t x) const {
    return __builtin_bswap64((x ^ RANDOM) * C); }
  size_t operator()(pair<uint64_t, uint64_t> x) const {
    return x.first* 31 + x.second;
    //return __builtin_bswap64(((something) ^ RANDOM) * C);
  } };
```

5.3 Greater In Left

```
void print(ll i){
  if(st.empty()) cout<<"No element in left";
  else cout<<st.top();
  cout<<" > "<<arr[i]<<endl; }
twoMax(ll n){
  st.push(arr[0]);
  for(ll i=1; i<n; i++){
    while(!st.empty() && arr[i]>=st.top()){
      //max_1st=arr[i],max_2nd=top()
      st.pop(); }
    print(i);
    //max_1st=top(),max_2nd=arr[i]
    st.push(arr[i]); } }
```

5.4 HandWritten Note

```
//Start:
















//end
```

5.5 Nim Game and Grundy Number

```
Nim Game: Lose=who can't make a move; if(xorSum != 0) 1 win;
Misera Nim: Lose = who make the last move
```

```
if all 1: if N is even, 1 win; else: if xorSum != 0, 1 win
Composite: Sob pile er jonno Grundy ber kore Nim er moto xor
Stones: N=> sub 2/3/5; if n%7 > 1, 1 win ;;
/*Grundy/Nimbers: smallest number jei state a jawa jabe nah
Problem: Divide pile in two unequal piles */
void Grandy(int n){
  int G[sz]; set<int>s;
  for(i=1; ; i++){
      j=n-i; if(j<=i) break;
      //i,j te jabo. insert i,j othoba i^j
      s.insert(G[i]^G[j]); }
  i=0;
  while(s.find(i)!=s.end()) i++;
  G[n]=i; }
```

5.6 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>
/*find_by_order(),order_of_key(),greater<int>,
less_equal<int>,less_equal<int> -->multiset,
order_of_key will return 1st pos, multiset er jonno
upper_bound --> lower_bound hisebe kaj kore.
To avoid: use pair {a[i], uniqueID}
tree<pii, null_type,less<pii>,...*/
```

5.7 Search

```
bool B_S(int key){
  int id=0; //-1
  for(int jump=n; jump>0; jump/=2)
    while(id+jump<n && arr[id+jump]<=key) id+=jump;
  return arr[id]==key; }
T_S(ll l, ll r) {
  while(r - l > 3) {
    m1 = l + (r - l) / 3; m2 = r - (r - l) / 3;
    f1 = f(m1); f2 = f(m2);
    if(f1 < f2) l = m1;
    else  r = m2;
  }
  for(i=l; i<=r; i++) mx = max(mx, f(i)); }
```

5.8 Sublime Build

```
{
  "shell_cmd": "g++ -std=c++20 \"${file}\" -o \"${file_path
      }/${file_base_name}\" && timeout 10s \"${file_path}/${
      file_base_name}\"<~/Documents/input.txt>~/Documents/
      output.txt && rm \"${file_path}/${file_base_name}\"",
  "file_regex": "^(..[^:]*):([0-9]+):?([0-9]+)?:? (.*)$",
  "working_dir": "${file_path}",
  "selector": "source.c++",
  "variants":
  [
    {
      "name": "Run",
      "shell_cmd": "\"${file_path}/${file_base_name}\"<~/
          Documents/input.txt>~/Documents/output.txt"
    },
    {
      "name": "Format Astyle",
      "shell_cmd": "astyle -A2 -T -p \"${file}\" && rm \"${
          file_path}/${file_name}\".orig"
    },
    {
      "name": "Format Astyle + build + run",
      "shell_cmd": "astyle -A2 -T -p \"${file}\" && rm \"${
          file_path}/${file_name}\".orig && g++ -std=c++20
          \"${file}\" -o \"${file_path}/${file_base_name}\"
          && timeout 10s \"${file_path}/${file_base_name
          }\"<~/Documents/input.txt>~/Documents/output.txt
          && rm \"${file_path}/${file_base_name}\""
    },
    {
      "name": "debug + input output",
      "shell_cmd": "g++ -DLOCAL -std=c++17 -Wshadow -Wall \"
          $file_name\" -o \"$file_base_name\" -fsanitize=
          address -fsanitize=undefined -D_GLIBCXX_DEBUG -g
          && timeout 10s \"${file_path}/${file_base_name
          }\"<~/Documents/input.txt>~/Documents/output.txt
          && rm \"${file_path}/${file_base_name}\""
    },
    {
      "name": "c++14",
      "shell_cmd": "g++ -std=c++14 \"${file}\" -o \"${
          file_path}/${file_base_name}\" && timeout 10s \"${
          file_path}/${file_base_name}\"<~/Documents/input.
          txt>~/Documents/output.txt && rm \"${file_path}/${
          file_base_name}\""
    },
    {
      "name": "opencv",
      "shell_cmd": "echo 'cmake_minimum_required(VERSION
          3.25.2)\nproject( \"$file_base_name\" )\
          nfind_package( OpenCV REQUIRED )\
          ninclude_directories( \\${OpenCV_INCLUDE_DIRS} )\
          nadd_executable( main \"$file_name\" )\
          ntarget_link_libraries( main \\${OpenCV_LIBS} )\n'
           > CMakeLists.txt && cmake . && make && gnome-
          terminal -- sh -c '\"${file_path}/main\";echo;echo
          ; echo Press ENTER to continue; read line'"

    },
  ]
}
```

5.9 builtin$_{function}$

```
double remainder(double x, double y)//float remainder
double fabs(double x ); //return double abs
double fmax(double x, double y); // return double max
double fracpart = modf (double x, double* intpart);
double trunc(double x)//integer part of a double
double fmod(double x, double y)//return float remainder
double exp(double x); double atan(double x);
double atan2(double y, double x); // tan-1(y/x)
binary_search(v.begin(),v.end(),x)
int i = stoi(str); int i = atoi(str) // char str[]
```

5.10 int128

```
#define ull __int128
#define getchar _getchar_nolock
#define putchar _putchar_nolock
ull read() {
    ull x = 0, f = 1;
    char ch = getchar();
    while(ch<'0' || ch >'9'){
        if(ch == '-') f=-1; ch = getchar(); }
    while(ch>='0' && ch<='9'){
        x=x*10+(ch-'0'); ch = getchar(); }
    return x * f; }
void print(ull x) {
    if(x < 0) { putchar('-'); x = -x; }
    if(x>9) print(x/10); putchar(x%10+'0'); }
bool cmp(ull x, ull y){ return x > y; }
```

5.11 preCode

```
#include<bits/stdc++.h>
using namespace std;
#define ALL(v) v.begin(),v.end()
#define clean(x,y) memset(x,y,sizeof(x));
#define endl "\n"
#define MOD 1000000007
#define MAX 200005
typedef long long ll;
int32_t main() {
  ios_base::sync_with_stdio(false); cin.tie(NULL);
  int TC = 1;
  //cin>>TC;
  for(int cs = 1; cs <= TC; cs++) {
    //cout << "Case " << cs << ": ";
    solve();
  } return 0; }

int x4[]={+1,-1,+0,+0}; int y4[]={+0,+0,+1,-1};
int x8[]={+0,+0,+1,-1,-1,+1,-1,+1};
int y8[]={-1,+1,+0,+0,+1,+1,-1,-1};
int xk[]={1,1,2,2,-1,-1,-2,-2};
int yk[]={2,-2,1,-1,2,-2,1,-1};
//freopen("a.in","r", stdin);
//freopen("a.out","w", stdout);
#pragma GCC target ("avx2")
#pragma GCC optimization ("O3")
#pragma GCC optimization ("unroll-loops")
```

6 String

6.1 Aho Corasick

```
#define t_sz 26
struct node{
  int ending,link,par,next[t_sz]; vector<int> idx;
  node(){
    ending=0; par=-1; link=-1; clean(next,-1); } };
struct aho_corasick{
  vector< node > aho;
  aho_corasick(){ aho.emplace_back(); }
  int ID(char ch){ if('a'<=ch && ch<='z') return ch-'a';
    return ch-'A'; }
  void ADD(string &s,int id=0){
    int now,u=0;
    for(auto ch:s){
      now=ID(ch);
      if(aho[u].next[now]==-1) {
        aho[u].next[now]=aho.size(); aho.emplace_back();}
      u=aho[u].next[now]; }
    aho[u].ending++; aho[u].idx.push_back(id); }
  int transition(int u, int i) {
    if(u==-1) return 0;
    if(~aho[u].next[i]) return aho[u].next[i];
    return aho[u].next[i]=transition(aho[u].link, i); }
  void push_links(){
    queue<int> q; q.push(0);
    while(!q.empty()){
      int u=q.front();q.pop();
      for(int i=0; i<t_sz; i++){
        int v=aho[u].next[i]; if(v==-1) continue;
        aho[v].par=u; aho[v].link=transition(aho[u].link,i);
        aho[v].ending+=aho[aho[v].link].ending;
        for(auto &id : aho[aho[v].link].idx)
            aho[v].idx.push_back(id);
        q.push(v); } } }
  int CNT(string &s, vector<string> &v){
    //text,list of pattern,int n=v.size();vector<int>pos[n];
    int u=0,sum=0;
    for(int i=0; i<s.size(); i++){
      int x=ID(s[i]); u=transition(u, x);
      sum+=aho[u].ending;
      //for(auto id : aho[u].idx) pos[id].push_back(i);
    }
    /*for(int i=0; i<n; i++){
      bool sp=0;
      for(auto p : pos[i]){
        if(sp) cout<<" "; cout<<p-v[i].size()+1; sp=1;
      } cout<<endl; } */
    return sum; }
  void clear(){
    aho.clear();aho.emplace_back(); } };
//min delete from s to avoid the set of string
//pos in s, state in aho-corasick call:solve(0,0)
int solve(int pos, int state) {
  if(ac.aho[state].ending) return INF;
  if(pos==s.size()) return 0;
  if(~dp[pos][state]) return dp[pos][state];
  int x=solve(pos+1, ac.transition(state, ac.ID(s[pos])));
  int y = 1+solve(pos+1, state);
  return dp[pos][state]=min( x , y); }
```

6.2 BitSet

```
bitset<MAX> mask[26];
void computeMask(string &text) {
  for(int i=1; i<(int)text.size(); ++i) {
    int c=text[i]-'a'; mask[c].set(i); } }
int StringMatchingInRange(string &pattern, string &text, ll
     l, ll r) {
  if(((int)pattern.size() - 1 )> (r-l+1)) return 0;
  /*computeMask(text);*/
  bitset<MAX> startMask; startMask.set();
  for(int i=1; i<(int)pattern.size(); ++i) {
    int c=pattern[i]-'a'; startMask &= (mask[c] >> i); }
  for(int i=l-1; i<r; i++) if(startMask[i]) cout<<i+1<<" ";
  return (startMask>>(l-1)).count()-(startMask >> (r - (int)
       pattern.size() + 2)).count(); }
//set text[idx]=ch, 1-based idx
void update(int idx, char ch, string &text) {
  char old=text[idx]; mask[old-'a'][idx]=0;
  text[idx]=ch; mask[ch-'a'][idx]=1; }
main(){ s="#"+s; t="#"+t; computeMask(t); }
```

6.3 Hash

```
#define MXN 10010
#define M1 1000050169
#define M2 1000004119
#define B1 1021
#define B2 2111
ll pow1[MXN] , pow2[MXN];
void pre(){
  pow1[0] = pow2[0] = 1;
  for(ll i=1; i<MXN; i++){
    pow1[i]=(pow1[i-1]*B1)%M1; pow2[i]=(pow2[i-1]*B2)%M2;}}
struct DoubleHash{
  ll H1[MXN], H2[MXN]; string keystring;
  DoubleHash(string &s){ keystring=s; calculate(s); }
  int id(char ch){ if('a'<=ch && ch<='z') return ch-'a'+1;
    return ch-'A'+27; }
  void calculate(string &str){ll n=str.size(),H1[0]=H2[0]=0;
    for(ll i=1; i<=n; i++){
      H1[i]=(B1*H1[i-1]+id(str[i-1]))%M1;
      H2[i]=(B2*H2[i-1]+id(str[i-1]))%M2; } }
  pair<ll,ll> tempHash(string &str){
    ll n=str.size(), h1=0, h2=0;
    for(ll i=1; i<=n; i++){
      h1=(B1*h1+id(str[i-1]))%M1;h2=(B2*h2+id(str[i-1]))%M2;
    }  return {h1,h2}; }
  ll getHash1(ll lft,ll rgt){ /*1 based*/ ll len=rgt-lft+1;
    return (H1[rgt]-H1[lft-1]*pow1[len]%M1+M1)%M1; }
  ll getHash2(ll lft,ll rgt){ ll len=rgt-lft+1;
    return (H2[rgt]-H2[lft-1]*pow2[len]%M2+M2)%M2; }
  pair<ll,ll> getHash(ll lft, ll rgt){
    //pre(): must be calculated
    return {getHash1(lft,rgt), getHash2(lft,rgt)}; }
  ll cmpSub(ll l1, ll r1, ll l2, ll r2){
    ll len1=r1-l1+1,len2=r2-l2+1,eql=0;
    ll low=0, high=min(len1,len2),mid;
    while(low<=high) {
      mid=(low+high)/2;
      if(getHash(l1,l1+mid-1)==getHash(l2,l2+mid-1)){
        eql=mid; low=mid+1;
      } else { high=mid-1; } }
    if(eql == min(len1, len2)) {
      if(len1==len2) return 0; if(len1<len2) return -1;
      return 1;/*large [l1...r1]*/ }
    if(keystring[l1+eql-1]<keystring[l2+eql-1])
      return -1; return 1; } };
```

6.4 KMP

```
/*  [i]=the longest length of a substring ending in position
i, that coincides with the prefix.
search: n=s.size(); s=s+"#"+t;if(pi[i]==n) pos=i-2n.
compress: k=n-pi[n-1],if(n%k==0) ans=k, else ans = n
pali pref: s=s+'#'+rev, pi[s.size()-1];
pali suf: s=rev+'#'+s, pi[s.size()-1]; */
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; }
/*Given a string s of length n. Count the number of
 appearances of each prefix s[0i] in string T. */
vi count_prefix(string s, string t){
  int n=s.size(),i,m=t.size(); string S=s; S+="#"; S+=t;
  vi ans(S.size()+2); vi pi = prefix_function(S);
  for(int i=n+1; i<S.size(); i++) ans[pi[i]]++;
  for(i=S.size()-1; i>n+1; i--) ans[pi[i-1]]+=ans[i];
  for(int i=n+1; i<=S.size(); i++) ans[i]++;
  return ans; }
```

6.5 Manachar

```
/*Count number of pali in s[L...R],If the size of palindrome
centered at i is x, then d1[i] stores (x+1)/2. If the size
of palindrome centered at i is x, then d0[i] stores x/2;*/
int d[2][MAX];
void manachar(string &s) {
  int n=(int)s.size();
  for(int t=0; t<2; t++) {
    for(int i=0,l=0,r=-1; i<n; i++){
      int k=(i>r)? 0:min(d[t][l+r-i+(t^1)], r-i+1);
      while(0<=i-k-(t^1) && i+k<n && s[i-k-(t^1)]==s[i+k])
        k++;
      d[t][i]=k--;
      if(i+k>r) { l=i-k-(t^1); r=i+k; } } } }
bool isPalindrom(int l, int r) {//0-based
  if(l>r) return 0; int len=r-l+1; int center=(l+r+1)>>1;
  return (d[(len&1)][center]>=((len+1)>>1)); }
int dp[MAX][MAX]; //cnt pal in s[l--r] = dp[l][r];
void preCalculate(string &s) {
  manachar(s); int n=(int)s.size();
  for(int i=1; i<=n; i++) {
    for(int j=i; j<=n; j++) {
      if(isPalindrom(i-1, j-1)){
        dp[1][j]++; dp[1][n+1]--;
        dp[i+1][j]--;dp[i+1][n+1]++; } } }
  for(int i=1; i<=n; i++)
    for(int j=1; j<=n; j++)
      dp[i][j]+=dp[i-1][j]+dp[i][j-1]-dp[i-1][j-1]; }
```

6.6 Palindromic Tree

```
const int N = 10004; const int t_sz= 26;
struct Palindromic_Tree{
  int tree[N][t_sz], idx,len[N], link[N], t,n;
  int endHere[N],occ[N],total=0; string s="#";
  int stop[N];//for printing
  Palindromic_Tree(){
    memset(occ, 0, sizeof(occ));
    len[1] = -1, link[1] = 1; len[2] = 0, link[2] = 1;
    endHere[1]=endHere[2]=0; idx = t = 2; }
  void add(int p) {
    while(s[p-len[t]-1] != s[p]) t=link[t];
    int x=link[t], c=s[p]-'a';
    while(s[p-len[x]-1] != s[p]) x=link[x];
    if(!tree[t][c]) {
      tree[t][c] = ++idx; len[idx] = len[t] + 2;
      link[idx] = len[idx] == 1 ? 2 : tree[x][c];
      endHere[idx]=1+endHere[link[idx]]; }
    t = tree[t][c]; occ[t]++; stop[t]=p; }
  void init(string &ss){
    s+=ss; n=(int)s.size();
    for(int i=1; i<n; i++){ add(i); total+=endHere[t];
    /*palindrom end in pos i=endHere[t]*/ }
    for(int i=idx; i>2; i--){
      occ[link[i]]+=occ[i];
      /*print palindrom
      int r=stop[i],l=stop[i]-len[i]+1;
      cout<<s.substr(l,len[i])<<endl;
      cout<<"len="<<len[i]<<",cnt="<<occ[i]<<"\n";*/ } }
  void clear(){
    for(int i=0; i<=idx; i++){
      occ[i]=endHere[i]=len[i]=link[i]=0;
      for(int j=0; j<t_sz; j++) tree[i][j]=0; }
    len[1] = -1, link[1] = 1; len[2] = 0, link[2] = 1;
    endHere[1]=endHere[2]=0; idx = t = 2; total=0; s="#";}
  int cntDistinct(){ return idx-2; }
  int cntTotal(){ return total; } };
//counting
vi suf;//suf[i]= # pali end at i
vi pref;//pref[i]= #pali start from i
p.init(s, suf); p.clear();
reverse(s);p.init(s,pref);reverse(pref);
void init(string &ss, vi &v) {
  //...//palindrom end in pos i=endHere[t]
  v.push_back(endHere[t]); /*....*/ }
void add(int p, int sId) {
  /*...*/ occ[t][sId]++; }
ll init(string &s1, string &s2) {
  ll commonPal=0LL; s+=s1; n=s.size();
  add(i, 0) for i = [1,n-1]
  s+=char('a'+26); s+=s2; n2=s.size();
  add(i, 1); for i = [n, n2-1]
  for(int i = idx; i > 2; i--){
    occ[link[i]][0] += occ[i][0];
    occ[link[i]][1] += occ[i][1];
    commonPal+=1LL*occ[i][0]*occ[i][1]; }
  return commonPal; }
```

6.7 Suffix Array

```
vector<int>p,c,lcp;
void build_suffix_array(string& s){
  int alphabet=256, n=(int)s.size(),i,k,cls;
  vector<int> cnt(max(alphabet, n), 0);
  vector<int> ptmp(n), ctmp(n);
  p.resize(n); c.resize(n); pair<int, int> cur, prev;
  for(i=0; i<n; ++i) cnt[s[i]]++;
  for(i=1; i<alphabet; i++) cnt[i]+=cnt[i-1];
  for(i=0; i<n; i++) p[--cnt[s[i]]]=i;
  c[p[0]]=0; cls=1;
  for(i=1; i<n; i++) {
    if(s[p[i]] != s[p[i-1]]) cls++;
    c[p[i]]=cls-1; }
  for(k=0; (1<<k)<n; ++k) {
    for(i=0; i<n; ++i) {
      ptmp[i]=p[i]-(1<<k);
      if(ptmp[i]<0) ptmp[i]+=n; }
    fill(cnt.begin(),cnt.begin()+cls,0);
    for(i=0; i<n; i++) cnt[c[ptmp[i]]]++;
    for(i=1; i<cls; i++) cnt[i]+=cnt[i-1];
    for(i=n-1; ~i; i--) p[--cnt[c[ptmp[i]]]]=ptmp[i];
    ctmp[p[0]]=0; cls=1;
    for(i=1; i<n; ++i) {
      cur={c[p[i]], c[(p[i]+(1<<k))%n]};
      prev={c[p[i-1]], c[(p[i-1]+(1<<k))%n]};
      if(cur!=prev) cls++; ctmp[p[i]]=cls-1; }
    c.swap(ctmp); } }
void build_lcp_array(string& s){
  int i,k=0,n=(int)s.size(); lcp.resize(n);
  //lcp[i]=lcp of suffix in pos i and pos i-1
  for(i=0; i<n-1; i++) {
    int pi=c[i]; int j=p[pi-1];
    while(s[i+k]==s[j+k]) k++;
    lcp[pi]=k; k=max(k-1, 0); } }
int LCP(int i, int j) {
  //use sparse table/segment tree
  int pi=c[i], pj=c[j]; if(pi>pj) swap(pi,pj);
  int mn=INT_MAX;
  for(i=pi+1; i<=pj; i++) mn=min(mn,lcp[i]);
  return mn; }
void init(string s) { s+="$";
  build_suffix_array(s); build_lcp_array(s); }
bool diffClass(int x, int y, int n){
  if(x>y) swap(x,y); return x<n&&y>n; }
void LongestCommonSubString(string s, string t){
  string ss=s+"@"+t+"$";
  build_suffix_array(ss); build_lcp_array(ss);
  int n=(int)s.size(); int N=(int)ss.size();
  int mx=0,start=0;
  for(int i=1; i<N; i++){
    if(diffClass(p[i], p[i-1], n)){
        if(lcp[i]>mx){ mx=lcp[i]; start=p[i]; } } }
  cout<<ss.substr(start, mx)<<endl; }
//Unique SubString: cnt+=(n-p[i]-1-lcp[i])
//cntSubStr:
int getPrev(int i, int len){
  int ans=i, lo=0,hi=i-1,mid;
  while(lo<=hi){
    mid=(lo+hi)/2;
    if(getLCP(mid, i)>=len){ ans=mid; hi=mid-1; }
    else { lo=mid+1; } } return ans; }
int getNext(int i, int len){
  int ans=i; int lo=i+1,hi=(int)lcp.size()-1,mid;
  while(lo<=hi){
    mid=(lo+hi)/2;
    if(getLCP(mid, i)>=len){ ans=mid;lo=mid+1; }
    else { hi=mid-1; } } return ans; }
{call-->
int len=r-l+1; int SaID=c[l];//index in SA
int L=getPrev(SaID, len); //lower,this is rank of S[l..r]
int R=getNext(SaID, len); //upper
cout<<(R-L+1)<<endl;}
//cnt pair of (i,j):k len same from i,j
struct MyStack{
  stack< pll >st; ll sum;
  MyStack() { sum=0LL; st.push({0LL,0LL}); }
  void pop() { auto x=st.top(); st.pop();
    sum-=(x.first)*(x.second-st.top().second); }
  void push(ll x, ll idx){
    while(st.top().first && st.top().first>x) pop();
    sum+=x*(idx-st.top().second); st.push({x,idx}); } };
{call-->
MyStack ms;
for(i=2; i<=n; i++){ ms.push(lcp[i],i-1); total+=ms.sum;} }
//k'th pos of t in s
int isEqual(int x){
  int m=(int)t.size();
  for(int i=0; i<m; i++) {
    if(s[i+x]==t[i]) continue; if(s[i+x]<t[i]) return -1;
    else return 1; } return 0; }
int Lower(vector<int>& p){
  int low=0,x,high=(int)p.size()-1,mid;
  while(low<=high) {
    mid=(low+high)/2; x=isEqual(p[mid]);
    if(x<0) low=mid+1; else high=mid-1; } return low; }
int Uppwer(vector<int>& p){
  int low=0,x, high=(int)p.size()-1, mid;
  while(low<=high) {
    mid=(low+high)/2; x=isEqual(p[mid]);
    if(x<=0) low=mid+1; else high=mid-1; } return low; }
main(){ s+="$"; vector<int> v=build_suffix_array(s);
  for(i=1; i<=n; i++) A[i] = v[i-1];
  wavelet_tree wt(A+1, A+n+1);
  cin>>t>>k; x = Lower(v); y = Uppwer(v);
  if((y-x)<k){pos=-1;} else pos = 1 + wt.kth(x+1, y, k) }
```

6.8 Trie

```
ll counter=1; //don't forget to ADD(0)
struct node{
  ll cnt[2],next[2];
  node() { cnt[0]=cnt[1]=0; next[0]=next[1]=0; }
}tree[MAX];
void ADD(ll n){
  ll i,now,cur=0;
  for(i=31; i>=0; i--){
    now=getbit(n,i); tree[cur].cnt[now]++;
    if(tree[cur].next[now]==0)
        tree[cur].next[now]=counter++;
    cur=tree[cur].next[now]; } }
ll xor_less_than_k(ll n, ll k){
  //n-->cum xor till now
  ll cur=0,i,now_n,now_k,ans=0;
  for(i=31; i>=0; i--){
    now_k=getbit(k,i); now_n=getbit(n,i);
    if(now_k){ ans+=tree[cur].cnt[now_n];
      if(tree[cur].next[now_n^1]==0) break;
      cur=tree[cur].next[now_n^1];
    } else { if(tree[cur].next[now_n]==0) break;
        cur=tree[cur].next[now_n]; } } return ans; }
int find_min(int n){
  int i,now,ans=0,cur=0;
  for(i=31; i>=0; i--){
    now=getbit(n,i); ans<<=1;
    if(data[cur].next[now]!=0) cur=data[cur].next[now];
    else{ cur=data[cur].next[now^1]; ans++; }} return ans;}
int find_max(int n){
  int cur=0,now,ans=0;
  for(int i=31; i>=0; i--){ now=getbit(n,i); ans<<=1;
    if(data[cur].next[now^1]!=0){
        cur=data[cur].next[now^1]; ans++;
    } else cur=data[cur].next[now]; } return ans; }
void deleteFromTrie(int n){
  int i,now,cur=0;
  for(i=31; ~i; i--){ now=getbit(n,i); data[cur].cnt[now]--;
    if(data[cur].cnt[now]==0){data[cur].next[now]=0; break;}
    cur=data[cur].next[now]; } }
void delete_trie(){ counter=1; memset(tree,0,sizeof tree); }
```

6.9 Z

```
/* z[i]=longest common prefix between s[1...n], s[i...n]
z[i]+i==n suffix matches with prefix
search: for(int i=0; i<t.size(); i++)
if(z[i+p.size()+1]==p.size()) found i; */
vector<int> z_function(string s){
  int n = (int) s.length(); vector<int> z(n); /*z[0]=n;*/
  for(int i=1, l=0, r=0; 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;  }
```