Hashing

```
constexpr int mod1 = 1000012253;
constexpr int mod2 = 1000000009;
template<typename T>
class MultiHashing {
public:
  int n;
  string s;
  string rev;
  vector<int> bases = {37,61,79,83,97,53,61,
                107,127,137,163,191 ,
                199,211,229,239,263 ,
                271,281,293 };

  vector<int> mods = {1000000007, 1000000009,
1000000433, 1000001329, 1000001927,
                1000003051, 1000003253, 1000003397,
1000003579, 1000003751,
                1000003793, 1000002043, 1000003967,
1000003987, 1000004123,
                1000004843, 1000006129, 1000006961,
1000012253, 1000015271};

  T base1;
  T base2;
  // T mod1;
  // T mod2;
  vector<pair<T, T>> prefix_hash;
  vector<pair<T, T>> suffix_hash;
  vector<pair<T, T>> power;
  vector<pair<T, T>> inv;


  T rng() {
    std::random_device rd;
    std::mt19937 gen(rd());
 std::uniform_int_distribution<T>
dis(0,  numeric_limits<T>::max());
    return dis(gen);
  }

 T mul(T a, T b, T mod) {
    return ((1LL * a % mod) * (b % mod)) % mod;
  }

  T add(T a, T b, T mod) {
    return (1LL * a + b) % mod;
  }

  T sub(T a, T b, T mod) {
    return ((a % mod) - (b % mod) + 2LL * mod) % mod;
  }

  T bigmod(T base, T power, T mod) {
    T res = 1;
    while (power > 0) {
      if (power & 1) {
        res = mul(res, base, mod);
      }
      base = mul(base, base, mod);
      power >>= 1;
    }
    return res;
  }

  MultiHashing(const string& str) : s(str) {
    n = s.size();
    base1 = bases[rng() % bases.size()];
    base2 = bases[rng() % bases.size()];
    rev = s;
reverse(rev.begin(), rev.end());
    // mod1 = mods[rng() % mods.size()];
    // mod2 = mods[rng() % mods.size()];
    prefix_hash.resize(n + 1, {0, 0});
    suffix_hash.resize(n + 1, {0, 0});
    power.resize(n + 1, {0, 0});
    inv.resize(n + 1, {0, 0});
    precom();
  }

void precom() {
    power[0] = {1, 1};


for (int i = 1; i <= n; i++) {
      power[i].first = mul(power[i - 1].first, base1,
mod1);
      power[i].second = mul(power[i - 1].second, base2,
mod2);
    }

    T inv_base1 = bigmod(base1, mod1 - 2, mod1);
    T inv_base2 = bigmod(base2, mod2 - 2, mod2);
    inv[0] = {1, 1};
    for (int i = 1; i <= n; i++) {
      inv[i].first = mul(inv[i - 1].first, inv_base1, mod1);
      inv[i].second = mul(inv[i - 1].second, inv_base2,
mod2);
    }

    for (int i = 1; i <= n; i++) {
      int ch = s[i - 1] - 'a' + 1;
      prefix_hash[i].first = add(prefix_hash[i - 1].first,
mul(ch, power[i - 1].first, mod1), mod1);
```

```
  prefix_hash[i].second = add(prefix_hash[i - 1].second,
mul(ch, power[i - 1].second, mod2), mod2);

        ch = rev[i - 1] - 'a' + 1;
        suffix_hash[i].first = add(suffix_hash[i - 1].first,
mul(ch, power[i - 1].first, mod1), mod1);
        suffix_hash[i].second = add(suffix_hash[i - 1].second,
mul(ch, power[i - 1].second, mod2), mod2);
      }
   }
   pair<T, T> get_hash(int l, int r) {
      T val1 = sub(prefix_hash[r].first, prefix_hash[l - 1].first,
mod1);
      val1 = mul(val1, inv[l].first, mod1);
     T val2 = sub(prefix_hash[r].second, prefix_hash[l -
1].second, mod2);

  val2 = mul(val2, inv[l].second, mod2);

      return {val1, val2};
   }

   pair<T, T> get_hash_rev(int l, int r) {
      T val1 = sub(suffix_hash[r].first, suffix_hash[l - 1].first,
mod1);
      val1 = mul(val1, inv[l].first, mod1);

      T val2= sub(suffix_hash[r].second, suffix_hash[l -
1].second, mod2);
      val2 = mul(val2, inv[l].second, mod2);

      return {val1, val2};
   }
   void change_current_hash(string &s)
    {
      this->s=s;
      this->rev=s;
      reverse(this->rev.begin(),this->rev.end());
      n=s.size();
      for(int i=1;i<=n;i++)
      {
         int ch=s[i-1]-'a'+1;
         prefix_hash[i].first=add(prefix_hash[i-
1].first,mul(ch,power[i-1].first,mod1),mod1);
         prefix_hash[i].second=add(prefix_hash[i-
1].second,mul(ch,power[i-1].second,mod2),mod2);

         ch=rev[i-1]-'a'+1;
         suffix_hash[i].first=add(suffix_hash[i-
1].first,mul(ch,power[i-1].first,mod1),mod1);
         suffix_hash[i].second=add(suffix_hash[i-
1].second,mul(ch,power[i-1].second,mod2),mod2);
      }
    }

   pair<T,T> get_new_hash(string &s)
{

         int n=s.size();
      T val1=0,val2=0;
      for(int i=1;i<=n;i++)
      {
         int ch=s[i-1]-'a'+1;
         val1=add(val1,mul(ch,power[i-
1].first,mod1),mod1);
         val2=add(val2,mul(ch,power[i-
1].second,mod2),mod2);
      }
      val1=mul(val1,inv[1].first,mod1);
 val2=mul(val2,inv[1].second,mod2);

      return {val1,val2};
   }

   // combine hash of two strings of length l1 and l2
   pair<T, T> combine_hash(pair<T, T> h1, pair<T, T> h2,
int l1) {
      T val1 = add(h1.first, mul(h2.first, power[l1].first,
mod1), mod1);
      T val2 = add(h1.second, mul(h2.second,
power[l1].second, mod2), mod2);
      return {val1, val2};
   }

   pair<T,T> get_modified_hash_changed_at_ith_index(int
i,char ch)
   {
      T firstf=0;
      T firsts=0;
      if(i>1)
      {
         pair<ll,ll> p=get_hash(1,i-1);
         firstf=p.ff;
         firsts=p.ss;
      }

      T secondf=0;

   T seconds=0;
        if(i<n)  {
 pair<ll,ll> p=get_hash(i+1,n);
 secondf=p.ff;
         seconds=p.ss;
      }
```

```
      int chh=ch-'a'+1;
      int pos=max(0,i-2);

      firstf = add(firstf, mul(chh, power[pos].first, mod1),
mod1);
      firsts = add(firsts, mul(chh, power[pos].second,
mod2), mod2);
      if(i==1)
      {
         firstf=mul(firstf, inv[i].first, mod1);
         firsts=mul(firsts, inv[i].second, mod2);
      }
      pair<T,T>
p=combine_hash({firstf,firsts},{secondf,seconds},i);
      return p;
   }
};
```

Z Algo:

```
vector<int> z_function(string str){
   Int lo = 0, hi = 0, n = str.size() ;
   vector<int> z(n) ;
   for(int i=1; i<n; i++)
   {
    if(i <= hi) z[i] = min(z[i - lo], hi - i + 1) ;
     while(i+z[i] < n && str[ z[i] ] == str[ i + z[i] ])
       z[i]++ ;
    if(i+z[i]-1 > hi) lo = i , hi = i+z[i]-1 ;
     }
     return z ;
}
```

MST(Kruskal)

```
class dsu
{
   vector<int> parent,size;
public:
   dsu(int n)
   {
      parent.resize(n+1);
      size.resize(n+1,1);
      iota(parent.begin(),parent.end(),0);
   }

   int findpar(int node)
   {
      if(node==parent[node])
      {
         return node;
      }
      return parent[node]=findpar(parent[node]);
   }

   void unionbysize(int u,int v)
   {
      int pu=findpar(u);
      int pv=findpar(v);

      if(pu==pv) return;

      int su=size[pu];
      int sv=size[pv];

      if(su>=sv)
      {

parent[pv]=pu;
      size[pu]+=size[pv];
      }
      else
      {
         parent[pu]=pv;
         size[pv]+=size[pu];
      }
   }

   void clear()
   {
      iota(parent.begin(),parent.end(),0);
      size.resize(parent.size(),1);
   }


};

int main()
{
   int node,edge;
   cin>>node>>edge;

   dsu ds(node);

   vector<array<int,3> > edges;

   for(int i=0;i<edge;i++)
   {
     int u,v,w;
      cin>>u>>v>>w;
      edges.push_back({w,u,v});
   }
```

```
 sort(edges.begin(),edges.end());

  int mstsum=0;
  vector<array<int,2> > mstedges;

  for(auto it : edges)
  {
     int w=it[0];
    int u=it[1];
    int v=it[2];

if(ds.findpar(u)!=ds.findpar(v))
    {
      mstsum+=w;
      mstedges.push_back({u,v});
      ds.unionbysize(u,v);
    }
  }

  cout<<mstsum<<endl;
  for(auto it : mstedges)
  {
    cout<<it[0]<<" "<<it[1]<<endl;
  }

}
```

Hashing on_seg_tree

```
const ll p=137;
const ll N=2e5+10;
const pair<ll,ll> mod={127657753,987654319};
ll powerr(ll a,ll b,ll mod){
ll r=1;
while(b){
if(b%2) r=((r%mod) *(a%mod))%mod;
a=((a%mod)*(a%mod))%mod;
b/=2;
}
return r;
}
ll add(ll a,ll b,ll mod){return
((a%mod)+(b%mod)+mod)%mod;}
ll substract(ll a,ll b,ll mod){return ((a%mod)-
(b%mod)+mod)%mod;}
ll mult(ll a,ll b,ll mod) {return
((a%mod)*(b%mod))%mod;}
ll fn(char ch){if(islower(ch)) return ch-
'a'+1;if(isupper(ch)) return ch-'A'+1;return ch-'0'+1;}

pair<ll,ll> pw[N+10],inv[N+10];


void precal(){
pw[0].F=pw[0].S=1;
for(int i=1;i<N;i++){
pw[i].F=mult(pw[i-1].F,p,mod.F);
pw[i].S=mult(pw[i-1].S,p,mod.S);
}
ll pw_inv1=powerr(p,mod.F-2,mod.F);
ll pw_inv2=powerr(p,mod.S-2,mod.S);
inv[0].F=inv[0].S=1;
for(int i=1;i<N;i++){
inv[i].F=mult(inv[i-1].F,pw_inv1,mod.F);
inv[i].S=mult(inv[i-1].S,pw_inv2,mod.S);
}
}
struct hashing {
vector<pair<ll,ll>> t;
  string s;
  hashing(){}
  hashing(string _s){
     s=_s;
     ll n=s.size();
     t.resize(n*4);
  }
  void build(int node,int l,int r){
     if(l==r){
       t[node].F=mult(pw[l].F,fn(s[l]),mod.F);
       t[node].S=mult(pw[l].S,fn(s[l]),mod.S);
       return;
     }
     ll mid=(l+r)>>1;
     build(node*2,l,mid);
     build(node*2+1,mid+1,r);
     t[node].F=add(t[node*2].F,t[node*2+1].F,mod.F);
     t[node].S=add(t[node*2].S,t[node*2+1].S,mod.S);
  }
  void upd(int node,int l,int r,int i,char value){
     if(l>i || r<i) return;
     if(l==i && r==i){
       t[node].F=mult(pw[i].F,fn(value),mod.F);
       t[node].S=mult(pw[i].S,fn(value),mod.S);
       return;
     }


    ll mid=(l+r)>>1;
     upd(node*2,l,mid,i,value);
     upd(node*2+1,mid+1,r,i,value);
     t[node].F=add(t[node*2].F,t[node*2+1].F,mod.F);
     t[node].S=add(t[node*2].S,t[node*2+1].S,mod.S);
  }
  pair<ll,ll> query(int node,int l,int r,int i,int j){
```

```
    if(l>j || r<i) return {0,0};          /// check here
    if(i<=l && r<=j) return t[node];
    ll mid=(l+r)>>1;
    pair<ll,ll> x=query(node*2,l,mid,i,j);
    pair<ll,ll> y=query(node*2+1,mid+1,r,i,j);
    return {add(x.F,y.F,mod.F),add(x.S,y.S,mod.S)};
  }
  pair<ll,ll> get_hash(int l,int r,int n){
    pair<ll,ll> ck=query(1,0,n-1,l,r);
    ck.F=mult(ck.F,inv[l].F,mod.F);
    ck.S=mult(ck.S,inv[l].S,mod.S);
    return ck;
  }
}a,b;
```

Seg_tree

```
const int N=2e5+123;
ll arr[N];
struct ST{
  ll t[N*3];
  ST(){
    memset(t,0,sizeof t);
  }
  inline void here(int node){
    t[node]=t[node*2]+t[node*2+1];  // check here
  }
  void build(int node,int l,int r){
```



```
      if(l==r){
      t[node]=arr[l];
      return;
```



```
    }
    ll mid=(l+r)>>1;
    build(node*2,l,mid);
    build(node*2+1,mid+1,r);
    here(node);
  }
  void upd(int node,int l,int r,int i,int value){
    if(l>i || r<i) return;
    if(l==i && r==i){
      t[node]+=value; // chack here
      return;
    }
    ll mid=(l+r)>>1;
    upd(node*2,l,mid,i,value);
    upd(node*2+1,mid+1,r,i,value);
    here(node);
  }
  ll query(int node,int l,int r,int i,int j){
    if(l>j || r<i) return 0;          /// check here
    if(i<=l && r<=j) return t[node];
    ll mid=(l+r)>>1;
    return
query(node*2,l,mid,i,j)+query(node*2+1,mid+1,r,i,j); //
check here
  }
}t;
```

Seg_tree(lazy)

```
const int N=2e5+123;
ll arr[N];
struct ST{
  ll t[N*3],lazy[N*3];
  ST(){
    memset(t,0,sizeof t);
   memset(lazy,0,sizeof lazy);
  }
```



```
  inline void push(int node,int l,int r){
    if(!lazy[node]) return;

t[node]+=lazy[node]*(r-l+1); // check here
    if(l!=r){
      lazy[node*2]+=lazy[node];
      lazy[node*2+1]+=lazy[node];

  }
    lazy[node]=0;
  }
  inline void here(int node){
    t[node]=t[node*2]+t[node*2+1];  // check here
  }
  void build(int node,int l,int r){
    if(l==r){
      t[node]=arr[l];
      return;
    }
    ll mid=(l+r)>>1;
    build(node*2,l,mid);
    build(node*2+1,mid+1,r);
    here(node);
  }
  void upd(int node,int l,int r,int i,int j,int value){
    push(node,l,r);
    if(l>j || r<i) return;
    if(i<=l && r<=j){
      lazy[node]+=value; // check here
      push(node,l,r);
      return;
    }
    ll mid=(l+r)>>1;
```

```
 upd(node*2,l,mid,i,j,value);
      upd(node*2+1,mid+1,r,i,j,value);
      here(node);
   }
   ll query(int node,int l,int r,int i,int j){
      push(node,l,r);
      if(l>j || r<i) return 0;          /// check here
      if(i<=l && r<=j) return t[node];
      ll mid=(l+r)>>1;
  return
query(node*2,l,mid,i,j)+query(node*2+1,mid+1,r,i,j); //
check here
   }
}t;
```

nCr&nPr

```
const ll mod=1e9+7;
ll fact[69];
ll poW(ll x, ll n){
   ll result = 1;
   while (n > 0){
      if (n & 1LL == 1){
         result = (result * x)%mod;
      }
      x = (x * x)%mod;
      n = n >> 1LL;
   }
   return result%mod;
}
ll nCr(ll n,ll r){
   return (fact[n] * poW((fact[r]*fact[n-r])%mod,mod-
2)) % mod;
}
```



```
ll nPr(ll n,ll r){
   return (fact[n] * poW(fact[n-r]%mod,mod-2)) % mod;
}
```

MO(sqrt_decmopostion)

```
#define block 555
struct query{
   int i;
   int l;
   int r;
};
query Q[200001];
```



```
int arr[200001],ans[200001];
int fre[200001];
int cnt=0;
bool cmp(query a,query b){
   if(a.l/block != b.l/block)
   return a.l/block < b.l/block;

   return a.r < b.r;
}


void add(int pos){
   fre[arr[pos]]++;
   if(fre[arr[pos]]==1)
   cnt++;
}
void remove(int pos){
   fre[arr[pos]]--;
   if(!fre[arr[pos]])
   cnt--;
}
```



```
int32_t main(){
   optimize();
   int n,q; cin>>n>>q;
   for(int i=0;i<n;i++) cin>>arr[i];
   for(int i=0;i<q;i++){
      cin>>Q[i].l>>Q[i].r;
      Q[i].i=i,Q[i].l--,Q[i].r--;
   }
   sort(Q,Q+q,cmp);
   int l=0,r=-1;
   for(int i=0;i<q;i++){
      int x=Q[i].l;
      int y=Q[i].r;
      while(l>x){
         l--, add(l);
      }
      while(r<y){
         r++,add(r);
      }
      while(l<x){
         remove(l),l++;


  while(r>y){
         remove(r),r--;
      }
      ans[Q[i].i]=cnt;
   }
   for(int i=0;i<q;i++)
   cout<<ans[i]<<endl;
}
```

Meet_in_the_middle

```
      ll n,m;
```

```
cin>>n>>m;
 vector<ll> v1,v2;
 ll bou=(n+1)/2;
 lup(0,n){
    ll x; cin>>x;
    if(n==1) v1.PB(x);
    else{
       if((ll)v1.size()<bou) v1.PB(x);
       else v2.PB(x);
    }
 }
 if(n==1) cout<<v1[0]%m<<endl;
 else{
    vector<ll> all;
    for(int musk=0;musk<(1LL<<v1.size());musk++){
       ll sum=0;
       for(int i=0;i<19;i++){
          if((musk&(1LL<<i))) sum+=v1[i];
       }
       all.PB(sum%m);
    }
    sort(all.begin(),all.end());
    all.resize(unique(all.begin(),all.end())-all.begin());
    ll ans=0;
    for(int musk=0;musk<(1LL<<v2.size());musk++){
       ll sum=0;
       for(int i=0;i<19;i++){

 if((musk&(1LL<<i))) sum+=v2[i];
       }
       ll x=(sum%m);
       ll y=((sum/m)+1)*m;
       y-=sum;
       y--;

 ll xy=upper_bound(all.begin(),all.end(),y)-all.begin();
          xy--;
          ans=max(ans,all[xy]+x);

       }
       cout<<ans<<endl;
    }
```

Ternary_search

```
  ld func(ld t,vector<prr>&vp){
  ld mx=0,mn=MAX;
  for(int i=0;i<vp.size();i++){
     mx=max(mx,(vp[i].F*t*1LL)+vp[i].S);
     mn=min(mn,(vp[i].F*t*1LL)+vp[i].S);
  }
  return (mx-mn);
}
int32_t main()
{
  optimize();
  int t = 1;
  // cin >> t;
  while (t--)
  {
     ld n,tm;
     cin>>n>>tm;
     vector<prr> vp;
     lup(0,n){
        ld speed,pos;
        cin>>speed>>pos;
        vp.PB({speed,pos});
     }
     ld l=0,r=tm;
     ld ans=1e12;
        while(r-l>=1e-12){
          ld t1=l+(r-l)/3;
          ld t2=r-(r-l)/3;
          ld x=func(t1,vp);
          ld y=func(t2,vp);
          if(x>=y){
             l=t1;
          }else {
             r=t2;
          }

 ans=min(ans,min(x,y));
       }
       fraction();
       cout<<ans<<endl;
    }
}
```

Wavelet Tree:

```
const int MAX = 1e6;
vi g[N];
int a[N];
struct wavelet_tree{
#define vi vector<int>
#define pb push_back
int lo, hi;
wavelet_tree *l, *r;
vi b;

//nos are in range [x,y]
//array indices are [from, to)
wavelet_tree(int *from, int *to, int x, int y){
```

```
lo = x, hi = y;
if(lo == hi or from >= to) return;
int mid = (lo+hi)/2;
auto f = [mid](int x){

  return x <= mid;
};
b.reserve(to-from+1);
b.pb(0);
for(auto it = from; it != to; it++)
b.pb(b.back() + f(*it));
//see how lambda function is used here
auto pivot = stable_partition(from, to, f);
l = new wavelet_tree(from, pivot, lo, mid);
r = new wavelet_tree(pivot, to, mid+1, hi);
}

//kth smallest element in [l, r]
int kth(int l, int r, int k){
if(l > r) return 0;
if(lo == hi) return lo;
int inLeft = b[r] - b[l-1];
int lb = b[l-1]; //amt of nos in first (l-1) nos that go in left

int rb = b[r]; //amt of nos in first (r) nos that go in left

if(k <= inLeft) return this->l->kth(lb+1, rb , k);
            return this->r->kth(l-lb, r-rb, k-inLeft);
}

//count of nos in [l, r] Less than or equal to k
int LTE(int l, int r, int k) {
if(l > r or k < lo) return 0;
if(hi <= k) return r - l + 1;
int lb = b[l-1], rb = b[r];
return this->l->LTE(lb+1, rb, k) + this->r->LTE(l-lb, r-rb,
k);
}

//count of nos in [l, r] equal to k
int count(int l, int r, int k) {
if(l > r or k < lo or k > hi) return 0;
if(lo == hi) return r - l + 1;
int lb = b[l-1], rb = b[r], mid = (lo+hi)/2;
if(k <= mid) return this->l->count(lb+1, rb, k);
return this->r->count(l-lb, r-rb, k);
}
~wavelet_tree(){
delete l;
delete r;
}
};
int main()
{
ios_base::sync_with_stdio(false);
cin.tie(NULL);
srand(time(NULL));
int i,n,k,j,q,l,r;

        }
           cin >> n;
fo(i, n) cin >> a[i+1];
wavelet_tree T(a+1, a+n+1, 1, MAX);
cin >> q;
while(q--){
int x;
cin >> x;
cin >> l >> r >> k;
if(x == 0){
//kth smallest
cout << "Kth smallest: ";
cout << T.kth(l, r, k) << endl;
}
if(x == 1){
//less than or equal to K
cout << "LTE: ";
cout << T.LTE(l, r, k) << endl;
}
if(x == 2){
//count occurence of K in [l, r]
cout << "Occurence of K: ";
cout << T.count(l, r, k) << endl;
}
}
return 0;
}
```

Dijkstra

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

vector<int> path;

void dijkstra(int &source,int
&destination,vector<array<long long,2> > graph[])
{
   priority_queue<array<long long,2>,vector<array<long
long,2> >,greater<array<long long,2> > > pq;

   int n=destination+1;

   vector<long long> dist(n,LONG_LONG_MAX);
   vector<int> parent(n);
```

```
 iota(parent.begin(),parent.end(),0);

   pq.push({0,source});
   dist[source]=0;

   while(!pq.empty())
   {
      int node=pq.top()[1];
      long long wt=pq.top()[0];

      pq.pop();

      if(wt>dist[node]) continue;

      for(auto it : graph[node])
      {
int newnode=it[0];
         long long newwt=it[1];

         if(dist[node]+newwt<dist[newnode])
         {
            dist[newnode]=dist[node]+newwt;
            pq.push({dist[newnode],newnode});
            parent[newnode]=node;
         }
      }
   }

   if(dist[destination]==LONG_LONG_MAX)
   {
      cout<<"NOT POSSIBLE"<<endl;
      return;
   }

 cout<<"Cost of shortest path is :
"<<dist[destination]<<endl;

   cout<<"Shortest path is : ";
   int node=destination;
   while(node!=source)
   {
      path.push_back(node);
      node=parent[node];
   }
   path.push_back(source);

   reverse(path.begin(),path.end());

   for(auto it : path)
   {
      cout<<it<<" ";
   }
   cout<<endl;
}
int main()
{
   ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);

   freopen("input.txt","r",stdin);
   freopen("output.txt","w",stdout);


   int node,edge;
   cin>>node>>edge;

   vector<array<long long,2> > graph[node+1];

   for(int i=0;i<edge;i++)
   {
      int u,v,w;
      cin>>u>>v>>w;
      graph[u].push_back({v,w});
      graph[v].push_back({u,w});

   }

   int source;
    cin>>source;

   int destination;
   cin>>destination;

   dijkstra(source,destination,graph);

}
```

SPFA

```
void spfa(vector<array<int,2> > graph[],int node)
{
   int inf=INT_MAX;
   vector<int> dis(node+1,inf);

   queue<int> q;
   vector<int> count(node+1,0);


vector<bool> inqueue(node+1,false);

   dis[1]=0;
   q.push(1);

  while(!q.empty())
   {
```

```
int node=q.front();
    q.pop();
    inqueue[node]=false;

    for(auto it : graph[node])
    {
      int newnode=it[0];
      int wt=it[1];

      if(dis[newnode]>dis[node]+wt)
      {
        dis[newnode]=dis[node]+wt;
        if(!inqueue[newnode])
        {
          q.push(newnode);
          inqueue[newnode]=true;
          count[newnode]++;
          if(count[newnode]>node)
          {
            cout<<"Negative Cycle Found"<<endl;
            return;
          }
        }
      }


    }
  }


  cout<<"No Negative Cycle Found"<<endl;

}
```

Floyed Warshall

```
bool floyd_warshall(vector<vector<int> >& graph,int
nodes)
{
  for(int k=1;k<=nodes;k++)
  {
    for(int i=1;i<=nodes;i++)
    {
      for(int j=1;j<=nodes;j++)
      {

graph[i][j]=min(graph[i][j],graph[i][k]+graph[k][j]);
      }
    }
  }

  for(int i=1;i<=nodes;i++)
  {
    if(graph[i][i]<0)  // if any node has negative weight
cycle then it will be less than 0
    {
      return true;
    }
  }

  return false;
}
```

Topo_Sort

```
 int node,edge;
cin>>node>>edge;
 vector<int> graph[node+1];
 for(int i=0;i<edge;i++)
 {
   int x,y;
  cin>>x>>y;
   graph[x].push_back(y);
 }

 vector<int> indegree(node+1,0);

 for(int i=1;i<=node;i++)
 {
   for(auto it: graph[i])
   {
     indegree[it]++;
   }
 }
 queue<int> q;
 vector<int> order;

 for(int i=1;i<=node;i++)
 {
   if(indegree[i]==0) q.push(i);
 }

 while(!q.empty())
 {
   int node=q.front();
   q.pop();

   order.push_back(node);

   for(auto it: graph[node])
```

```
{
        indegree[it]--;

     if(indegree[it]==0) q.push(it);
     }
  }

  if(order.size()!=node) cout<<"Cycle is present"<<endl;
  else
  {
     for(auto it: order) cout<<it<<" ";
     cout<<endl;
}
```

Cycle(undirected)

```
bool cycle_dfs(int node,int parent)
{
   dis.push_back(node);
   vis[node]=1;
   for(auto x: graph[node])
   {

      if(vis[x]==0)
      {
         if(cycle_dfs(x,node)==true) // if true then returns
true doesnt call for further dfs
         {
            return true;
         }
      }
      else if(vis[x]==1 and x!=parent)
      {
         return true;
}
   }
   return false;

}
```

Cycle(directed)

```
vector<int> graph[mx];

vector<bool> vis(mx,0);
vector<int> path(mx,0);
int start=-1;
bool dfs(int node)
{
   vis[node]=1;
   path[node]=1;
   for(auto child : graph[node])

{
      if(vis[child]==0)
      {
         if(dfs(child)==true)
         {
            return true;
         }
      }
      else if(path[child]==1)
      {
         start=child;
         return true;
      }
}
   path[node]=0;
   return false;
}

int main()
{
   file();
   int node,vertice;
   cin>>node>>vertice;
   for(int i=1;i<=node;i++)
   {
      graph[i].clear();
      vis[i]=0;
      path[i]=0;
   }
   map<int,int> mp;
   for(int i=1;i<=vertice;i++)
   {
      int x,y;
      cin>>x>>y;
      graph[x].push_back(y);
      mp[x]=y;
   }
   for(int i=1;i<=node;i++)
   {
      if(vis[i]==0)
   {
         if(dfs(i)==true)
         {
            cout<<"Cycle found"<<endl;
            break;
         }
      }
   }
```

```
if(start==-1)
  {
    cout<<"No cycle found"<<endl;
  }
  else
  {
    vector<int> ans;
    ans.push_back(start);
    int x=mp[start];
    while(x!=start)
    {
      ans.push_back(x);
      x=mp[x];
    }
    for(auto x: ans)
    {
      cout<<x<<" ";
    }
    cout<<endl;
  }
}
```

Seg_Seive

```
vector<ll> primes,primes1;
void seive(ll n)
{
vector<ll> isprime(n+1,true);
for(ll i=2;i<=n;i++)
{
if(isprime[i])
          {
for(ll j=i*i;j<=n;j+=i) isprime[j]=false;
}
}

for(int i=2;i<=n;i++)
{
if(isprime[i]) primes.pb(i);
}
}

void segseive(ll low,ll high)
{
seive(sqrtl(high));
vector<bool> prime(high-low+100,true);
int sz=high-low+5;

for(auto it : primes)
{

ll sm=(low/it)*1LL*it;
if(sm<low) sm+=it;

for(ll i=sm;i<=high;i+=it)
{
prime[i-low]=false;
}
}
for(ll i=low;i<=high;i++)
{

          if(prime[i-low]) primes1.pb(i);
}
}
```

Sparse_Table

```
template <typename T>
class SparseTable
{
  public:
  vector<vector<T> > st;

  T op(T a,T b)
  {
    return __gcd(a,b);
  }

  SparseTable(int n,vector<T> &vec)
  {
    st.resize(n+2,vector<T> (__lg(n)+2));

    for(int i=1;i<=n;i++)
    {
      st[i][0]=vec[i];
    }

    int k=__lg(n)+1;

    for(int j=1;j<=k;j++)
    {
      for(int i=1;i+(1<<j)<=n+1;i++)
      {
        st[i][j]=op(st[i][j-1],st[i+(1<<(j-1))][j-1]);
      }
    }

  }

  T query(int l,int r)
  {
    int j=__lg(r-l+1);
```

```
    return op(st[l][j],st[r-(1<<j)+1][j]);
  }


 T query1(int l,int r) // query in logn for non idempotent
function
  {
    int ans=0;
    for(int j=__lg(r-l+1);j>=0;j--)
    {
      if((1<<j)<=(r-l+1))
      {
        ans=op(ans,st[l][j]);
        l+=(1<<j);
      }
    }
    return ans;
  }


};
```

N_Queen:

```
  Int n;
  cin>>n
   l done=(1LL<<n)-1;
   int cnt=0;
   vector<int> queen;
    function<void(int,int,int)> rec=[&](int rowmask,int
ld,int        rd)
  {
    if(rowmask==done)
    {
      cnt++;
      for(int i=0;i<n;i++)
      {
        for(int j=0;j<n;j++)
        {
          if(queen[i]==j) cout<<"Q";
          else cout<<".";


        }
          cout<<el;
      }
      cout<<el;
      return;
    }
    ld&=done;
    rd&=done;
    int safe=done & (~(rowmask | ld | rd));
    while(safe)
    {
      int p=safe & (-safe);
      safe=safe-p;
      int col=__builtin_ctz(p);
      queen.push_back(col);
      rec(rowmask|p,(ld|p)<<1,(rd|p)>>1);
      queen.pop_back();

    }
  };
  rec(0,0,0);
  cout<<cnt<<el;
```

Diophantine Euation

```
int diophantine(int &a,int &b,int &c,int &x,int &y){
int g = egcd(abs(a),abs(b),x,y) ;
if(c % g != 0) return -1 ;
x *= c/g ;
y *= c/g ;
if(a < 0) x *= -1 ;
if(b < 0) y *= -1 ;
return g ;
}
```

LIS

```
/// use upper_bound for non decreasing order
int lis(){
vector<int> dp ;
for(int i=0; i<lim; i++){
auto it=lower_bound(dp.begin(),dp.end(),arr[i]) ;
if(it == dp.end()) dp.push_back(arr[i]) ;
else dp[it-dp.begin()] = arr[i] ;
}
return dp.size() ;
}

void lis_print(){
vector<int> dp ;
for(int i=0; i<lim; i++){
f[i] = lower_bound(dp.begin(), dp.end(), arr[i]) -
dp.begin() ;
if(f[i] == dp.size()) dp.push_back( arr[i] ) ;
else dp[ f[i] ] = arr[i] ;
f[i]++ ; /// LIS from 0 - i for ith element include
}
int x = dp.size() ;
vector<int> lis ;
for(int i=lim-1; i>=0; i--){
```

```
if(f[i] == x){
lis.push_back( arr[i] ) ;
x-- ;
}
}
reverse(lis.begin(), lis.end()) ;
}
```

Mobious

```
int mob[lim] ;
/*
mob[i] = 0 , if i has squared prime factor

mob[i] = 1 , if it is square free and it has odd
number of prime factor
mob[i] = -1, if it is square free and it has even
number of prime factor
*/
void mobius(){
mob[1] = 1 ;
for(int i = 1; i < lim; i++){
for(int j = i+i; j < lim; j += i)
mob[j] -= mob[i] ;
}
}
```

Legendre

```
///it calculates the maximum power or total power of p in
N! (factorial)
///if n = 5 then n! = 120 , if p = 2 then total power of 2 is =
3;
///120 = (2 ^ 3) * (3 ^ 1) * (5 ^ 1)
int legendre(int n,int p )
{
  int ans =0;
  while(n)
  {
    ans+= n / p;
    n/=p;
  }
  return ans;
}
```

DSU(rollbacks)

```
class DSU {
  vector<int> parent, size;
  int comp;

 stack<pair<int, int>> parentChanges;
  stack<pair<int, int>> sizeChanges;
public:
  DSU(int n) {
    parent.resize(n + 1, 0);
    size.resize(n + 1, 1);
    for (int i = 0; i <= n; i++) {
      parent[i] = i;
    }
    comp = n;
  }

  int findUpar(int node) {
    if (node == parent[node]) {
      return node;
    }
    return parent[node] = findUpar(parent[node]);
  }

  void unionBysize(int u, int v) {
    Int ulpar_u = findUpar(u);
    int ulpar_v = findUpar(v);

    if (ulpar_u == ulpar_v) {
      return;
    }

    if (size[ulpar_u] >= size[ulpar_v]) {



parentChanges.push({ulpar_v, parent[ulpar_v]});
      sizeChanges.push({ulpar_u, size[ulpar_u]});

      parent[ulpar_v] = ulpar_u;
      size[ulpar_u] += size[ulpar_v];
    } else {
      parentChanges.push({ulpar_u, parent[ulpar_u]});
      sizeChanges.push({ulpar_v, size[ulpar_v]});

      parent[ulpar_u] = ulpar_v;
      size[ulpar_v] += size[ulpar_u];
    }

    comp--;
  }

  void rollback() {
    if (!parentChanges.empty() && !sizeChanges.empty())
{
      // Revert the last parent and size changes
      auto [node, oldParent] = parentChanges.top();
      parentChanges.pop();
      parent[node] = oldParent;
```

```
    auto [nodeSize, oldSize] = sizeChanges.top();
        sizeChanges.pop();
        size[nodeSize] = oldSize;

        comp++;  // Increase the number of components
back
      }
   }

   int getComp() {
 return comp;
   }
};
```

XOR_TRIE

```
class TrieNode
{
public:
   TrieNode *left;
   TrieNode *right;
   int cnt=0;
   TrieNode()
   {
      left=NULL;
      right=NULL;
      cnt=0;
   }
};
class Trie
{
   TrieNode *root;
public:
Trie()
  {
     root=new TrieNode();
  }
  void insert(int n)
  {
     TrieNode *curr=root;
     for(int i=31;i>=0;i--)
     {
        int bit=(1&(n>>i));

      if(bit==0)
       {
         if(curr->left==NULL)
          {
             curr->left=new TrieNode();
          }
          curr=curr->left;
          curr->cnt++;
       }
       else
       {
          if(curr->right==NULL)
          {
             curr->right=new TrieNode();
          }
          curr=curr->right;
          curr->cnt++;
       }
     }
  }
  void remove(int n)
  {
     TrieNode* curr = root;
     for (int i = 31; i >= 0; i--)
     {
        if(curr==NULL)
        {
           break;
        }

        int bit = (n >> i) & 1;

  if (bit == 0)
        {
           curr = curr->left;
            curr->cnt--;
        }
        else
        {
           curr = curr->right;
           curr->cnt--;

        }
     }

}

int max_xor_pair(int n)
{
   TrieNode *curr=root;
   int ans=0;
   for(int i=31;i>=0;i--)
   {
      if(curr==NULL)
      {
         break;
      }
      int bit=(1&(n>>i));
```

```
        if(bit==0)
        {
          if(curr->right!=NULL and curr->right->cnt>0)
          {
            ans+=(1<<i);
            curr=curr->right;
          }
           else
          {
            curr=curr->left;
          }

    }
        else
        {
          if(curr->left!=NULL and curr->left->cnt>0)
          {
            ans+=(1<<i);
            curr=curr->left;
          }
          else
          {
            curr=curr->right;
          }
        }
    }
    return ans;
  }

  int find_x_such_that_x_xor_y_less_than_or_equal_k(int
y,int k)
  {
    TrieNode *curr=root;
int ans=0;
    for(int i=31;i>=0;i--)
    {
      int bity=(1&(y>>i));
      int bitk=(1&(k>>i));
      if(bity)
      {
        if(bitk)
        {
        if(curr->right!=NULL)
          {
            ans+=curr->right->cnt;
           }
          curr=curr->left;
        }
        else
        {
          curr=curr->right;
        }
      }
      else
      {
        if(bitk)
        {
          if(curr->left!=NULL)
          {
            ans+=curr->left->cnt;
          }
          curr=curr->right;
        }
        else curr=curr->left;
      }
    }
    ans+=curr->cnt;
return ans;
 }
 int subarray_less_than_k(int n,int k)
 {
    TrieNode *curr=root;
    int ans=0;
    for(int i=31;i>=0;i--)
     {
      if(curr==NULL)
      {
        break;
        }
      int bitk=(1&(k>>i));
      int bitn=(1&(n>>i));

      if(bitn==bitk)
      {
        if(bitk)
        {
          if(curr->right!=NULL)
          {
            ans+=curr->right->cnt;
          }
        }
        curr=curr->left;
      }
      else
      {
        if(bitk)
        {
          if(curr->left!=NULL)
          {
            ans+=curr->left->cnt;
          }
        }
```

```
        curr=curr->right;
       }
     }
     return ans;
   }

};
```

Formula

$1^2 + 2^2 + 3^2 + ... + n^2 = (1/6)\{n(n + 1)(2n + 1\}$ for all $n \in N$.

$(1 \times 2) + (3 \times 4) + (5 \times 6) + ... + (2n - 1) \times 2n = n(n+1)(4n-1)/3$

$1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + ... + n(n + 1) = (1/3)\{n(n + 1)(n + 2)\}$.

$1 \cdot 3 + 3 \cdot 5 + 5 \cdot 7 + ...+ (2n - 1)(2n + 1) = (1/3)\{n(4n^2 + 6n - 1)$.

$1/(1 \cdot 2) + 1/(2 \cdot 3) + 1/(3 \cdot 4) + ... + 1/\{n(n + 1)\} = n/(n + 1)$

1. 3 ways to do modulo inverse. 1/a % m which is equivalent to ax = 1 mod m
a. m is prime - a^(m-2) = 1/a mod m

b. m is not prime - ax+my = 1 find x by egcd
c. m is not prime - a^phi(m)-1 = 1/a mod m
2. NOD(n) = (a1+1) * (a2+1) * (a3+1) * … * (ak+1)
3. SOD(n) = (pk^(ak+1) - 1) / (pk - 1)
4. Egcd solve the equation ax + by = gcd(a, b)
a. X += k * b/gcd(a,b) ; Y -= k * a/gcd(a,b)
5. Diophantine equation : ax + by = c solution exist iff c | gcd(a, b)
6. gcd(a, b) = gcd(-a, y) = gcd(x, -y) = gcd(-x, -y)
7. gcd(a, b) = gcd(a, b+a) = gcd(a, b-a), here b > a
8. gcd(a[0], a[1], a[2],..,a[n]) = gcd(a[0], a[1]-a[0], a[2]-a[1],..,a[n]-a[n-1])
a. k is added to all value then just add it to a[0]
9. phi(n) = n * (1-1/p1) * (1-1/p2) .. (1-1/pk)
10.Number of x <= n and gcd(x, n) = 1 is equal to phi(n/d)
11.Sum of above x is equal to phi(n) * n / 2
12.Sum of phi(d) = n for all d.
13.Arithmetic progression : a + (d+a) + (2d+a) ..
a. Nth term : a + (n-1)d
b. Sum of n term : n/2[2a + (n-1)*d]

c. Sum of AP : n/2[1st term + nth term]

14.Geometric progression: a + ar + ar^2 + ar^(n-1)
a. Nth term : ar(n-1)
b. Sum of n term : a[(r^n - 1)/(r-1)]
c. Sum of n term : (nth*r - 1st)/(r-1)
15.Harmonic sum 1+1/2+1/3+..+1/n = logn +1

Bitwise

16.- Check kth bit of a number by : x & (1 << k) if greater then 0 then kth bit is 1 otherwise 0.
17.- Set kth bit of a number by : x = x | (1 << k) .
18.- Reset kth bit of a number by : x = x & ~(1<<k)
19.- Toggle kth bit of a number by : x = x ^ (1<<k).
20.- Check x is a power of 2 by : x & (x-1) == 0 then it is power of 2 otherwise not.
21.- All bit of x are on if x & (x + 1) == 0 .
22.- XOR of 1 to N in constant time : (n&3==0)→n,(n&3==1)→1, (n&3==2)→n+1,otherwise 0.
23.- Find k bit number whose all bit are on : number = (1 << k) - 1

24.- 2's complement of a number means if we traverse a binary number from LSB to MSB and if we find a set bit, after that first set bit if we flip remaining bits it will for 2's complement . We can get 2's complement of x by : (-x).
25.- 2's biggest power which is divisor of x

: 1 << (__builtin_ctz(x)).
26.- 2's smallest power which is not smaller then x : 1 << (32 - __builtin_clz(x - 1)). But it has Undefined Behavior for x ≤ 1 so we need an extra check for this.
27.a+b = (a^b) + 2*(a&b)
28.a|b = (a^b) + (a&b)

Graph
29.Mincut = Max flow
30.Maximum independent set = V - matching
31.Minimum vertex cover = matching
32.Minimum edge cover (general) = V - Matching

Geometry and Trigonometry

33.sin(th)^2 + cos(th)^2 = 1
34.sec(th)^2 - tan(th)^2 = 1
35.cosec(th)^2 - cot(th)^2 = 1
36.a/sinA = b/sinB = c/sinC = 2R
37.a^2 = b^2 + c^2 - 2bccosA
38.A = bcosC + ccosB
39.T.area = 1/2 bcsinA = sqrt(s*(s-a)*(s-b)*(s-c))
40.In circle radius,r = (2*T.area)/(a+b+c)
41.Circumcicle radius,R = abc/4T.area
42.Rr = abc/4s
43.T.area = 1/2bh
44.T.area = 1/2(x1*(y1-y3)+x2*(y3-y1)+x3*(y1-y2))
45.C.area = pi * r^2
46.Circumference = 2*pi*r
47.Length of arc = pi*r*th/180
48.Sector area = (th/360) * pi * r^2
49.y-y1 = m (x-x1)
50.y = mx + c
51.Line - Ax + By = C or Ax+By+C = 0
52.Parametric - p(t) = p0 + t(p1-p0)

Template

```
// pragmas
// #pragma GCC optimize("Ofast")
// #pragma GCC target("avx,avx2,fma")
// #pragma GCC optimization ("unroll-loops")
// #pragma GCC optimization ("strict-overflow")

#include<bits/stdc++.h>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/assoc_container.hpp>

using namespace std;
using namespace __gnu_pbds;

                    ////    TEMPLATE    ////

//------------------------------------------------------------------
----------------------------------------------------------|


# define   el '\n'
# define   sp " "
# define   ff first
# define   ss second
# define   ll long long
# define   pb push_back
# define   mp make_pair
# define   yess1 cout<<1<<el
# define   noo cout<<"NO"<<el
# define   yess cout<<"YES"<<el
# define   siz(x) (int)x.size()
# define   ull unsigned long long
# define   all(v) v.begin(),v.end()
# define   allr(v) v.rbegin(),v.rend()
# define   torad(x) ((x) * ((2*acos(0))/180.0))
# define   todeg(x) ((x) * (180.0/(2*acos(0))))

constexpr ll mod=1000000000+7;
constexpr ll INF=LLONG_MAX;
constexpr double PI= acos(-1);
constexpr double eps=1e-9;

# define mem(a,b) memset(a,b,sizeof(a))
# define sqr(a) ((a)*(a))
```

```
# define lcm(a,b) (a*b)/__gcd(a,b)

# define optimise   { ios_base::sync_with_stdio(false);
cin.tie(NULL); cout.tie(NULL); }
# define fraction(a) cout.unsetf(ios::floatfield);
cout.precision(a); cout.setf(ios::fixed,ios::floatfield);
# define ordered_set tree<ll, null_type,less_equal<ll>,
rb_tree_tag,tree_order_statistics_node_update>

// find_by_order() - Returns an iterator to the k-th largest
element (counting from zero)
// order_of_key()  - The number of items in a set that are
strictly smaller than our item
// greater instead of less for descending order
// less_equal works as ordered multiset
// we can use pair<int,int> instead of int for pair of orderd
set
// for multiset st.lower_bound(x) works as upper bound
and st.upper_bound(x) works as lower bound

inline void file() {
#ifndef ONLINE_JUDGE
freopen("input.txt", "r", stdin);
                                freopen("output.txt",
"w", stdout);
#endif // ONLINE_JUDGE
}
//------------------------------------------------------------------
----------------------------------------------------------|


                                // DEBUGGER //
```

```
//------------------------------------------------------------------
----------------------------------------------------------|

template < typename F, typename S > ostream& operator
<< ( ostream& os, const pair< F, S > & p ) { return os << "("
<< p.first << ", " << p.second << ")"; }
template < typename T > ostream &operator << ( ostream
& os, const vector< T > &v ) { os << "{"; for(auto it =
v.begin(); it != v.end(); ++it) { if( it != v.begin() ) os << ", ";
os << *it; }  return os << "}";  }
template < typename T >  ostream &operator << ( ostream
& os, const set< T > &v ) { os << "["; for(auto it = v.begin();
it != v.end(); ++it) { if( it != v.begin() ) os << ", "; os << *it;  }
return os << "]"; }
template < typename T > ostream &operator << ( ostream
& os, const multiset< T > &v ) { os << "["; for(auto it =
v.begin(); it != v.end(); ++it) { if( it != v.begin() ) os << ", ";
os << *it; } return os << "]"; }



template < typename F, typename S > ostream &operator
<< ( ostream & os, const map< F, S > &v ) { os << "[";
for(auto it = v.begin(); it != v.end(); ++it) { if( it !=
v.begin() ) os << ", "; os << it -> first << " = " << it ->
second ; } return os << "]";  }
#define dbg(args...) do {cerr << #args << " : ";
iamarman(args); } while(0)
void iamarman () { cerr << endl; }
template <typename T> void iamarman( T a[], int n )
{   for(int i = 0; i < n; ++i) cerr << a[i] << ' '; cerr << endl;  }
template <typename T, typename ... hello>  void
iamarman( T arg, const hello &... rest) {   cerr << arg << ' ';
iamarman(rest...);  }
//------------------------------------------------------------------
----------------------------------------------------------|
```

```
                ////   FUNCTIONS   ////


ll bigmod(ll base,ll power){ ll res=1; ll p=base%mod;
while(power>0) { if(power%2==1)
{  res=((res%mod)*(p%mod))%mod; }  power/=2;
p=((p%mod)*(p%mod))%mod; } return res; }

ll inversemod(ll base) { return bigmod(base,mod-2); }


ll poww(ull a,ull b) { ull ans=1; if(!b) return 1; while(b>1)
{  if(b&1) { ans=ans*a%mod; } a=a*a%mod; b/=2; }return
a*ans%mod; }
                        int gcd(ll a,ll b) { ll rem; while(b%a!=0)
{  rem=b%a;  b=a;  a=rem; } return a; }

ll sqrtt(ll a){ long long x = sqrt(a) + 2; while (x * x > a) x--;
return x;}

ll sqrt(ll n) {ll low=0,high=1e10; while(high-low>1){ ll
mid=low+(high-low)/2; if(mid*mid<=n) low=mid; else
high=mid; }return low; }

long double sqrtd(long double n){ long double
low=0,high=n,mid; for(int i=0;i<100;i++)
{ mid=(low+high)/2; if(mid*mid<=n) low=mid; else
high=mid;} return low;}

mt19937
rng(chrono::high_resolution_clock::now().time_since_epoc
h().count());
```

```
inline ll getrandom(ll a,ll b) { return
uniform_int_distribution<ll>(a,b)(rng); }


int dx[]={-1, 1 , 0 , 0 , -1 ,-1, 1, 1};
int dy[]={ 0, 0 ,-1 , 1 , -1 , 1,-1, 1};

// up = { -1,0 } , down = { 1,0 } , right = { 0,1 } , left = { 0,-1 }
// up-right = { -1,1 } , up-left = { -1,-1 } , down-right =
{ 1,1 } , down-left = { 1,-1 }

 ///  _____CODE STARTS FROM HERE_____    ///

void solve()
{

}
int main()
{
   optimise;
   file();

   clock_t start= clock();
   int t;
   cin>>t;
   for(int i=1;i<=t;i++)
   {
      solve();

   }
   cerr << "Run Time : " <<((double)(clock() - start) /
    CLOCKS_PER_SEC)<<el;
}
```

Template

```
#include <bits/stdc++.h>
using namespace std;
#define optimize()              \
   ios_base ::sync_with_stdio(0); \
   cin.tie(0);                 \
   cout.tie(0);
#define fraction()
cout.unsetf(ios::floatfield);cout.precision(10);cout.setf(ios:
:fixed,ios::floatfield);
void file(){
#ifndef ONLINE_JUDGE
   freopen("input.txt", "r", stdin);
   freopen("output.txt", "w", stdout);
#endif
}
#define endl '\n'
#define ll long long int
#define ha cout << "YES" << endl
#define na cout << "NO" << endl
#define na1 cout << "-1" << endl;
#define lup(z, n) for (int i = z; i < n; ++i)
#define lup_1(z, n) for (int i = z; i <= n; ++i)
#define MAX (ll)(1LL << 31) - 1
#define F first
#define S second
#define PB push_back
#define MP make_pair
#define IS to_string
#define SI stoi
#define popcnt __builtin_popcountll
#define clz __builtin_clz
#define um unordered_map
#define mem(arr,x) memset(arr,x,sizeof arr)
#define prr pair<ll,ll>
#define rt(v,l) rotate(v.begin(),v.begin()+l,v.end()); // arr
= 1,2,3,4,5,6 dile amra index 3 input dile ans arr=
4,5,6,1,2,3 dibe
ll lcm(ll a, ll b)
{
   ll g = __gcd(a, b);
   return (a * b) / g;
}
//----------policy base data structure ---------
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
typedef tree< ll, null_type, less_equal<ll>, rb_tree_tag,
tree_order_statistics_node_update > pbds;
/// change ll to any data type
/// less_equal for multiset increasing order
/// less for set increasing order
/// greater_equal for multiset decreasing order
/// greater for set decreasing order

/// cout<<*X.find_by_order(1)<<endl; // eta index input
dile oi index er value print korbe
/// cout<<X.order_of_key(-5)<<endl;  // input value theke
koto gula small value ase setar count print korbe
// for pbds set---
// 1.cout<<*lower_bound(x)<<endl;
2.cout<<*upper_bound(x)<<endl;

//-----------PBDS tamplate end------------------
//------------------ seive ----------------------------
// to show prime numbers
const ll mx=1e8;
```

```
bitset<mx> prime;
vector<ll> prime_show;
void seive(ll n){prime[1]=1;for(int i=3;i*i<=n;i+=2){
if(!prime[i]){for(int j=i*i;j<=n;j+=(i*2)){prime[j]=1;}}}
prime_show.PB(2);
for(int i=3;i<=n;i+=2){if(!prime[i]) prime_show.PB(i);}
}
//----------------segment seive-------------------------
vector<ll>segsiv;
void segmentedSieve(ll L, ll R)
{
bool isPrime[R-L+1];
for(int i=0 ; i<=R-L+1 ; i++) isPrime[i]=true;

if(L==1) isPrime[0]=false;

for(int i=0 ; prime_show[i]*prime_show[i]<=R ; i++){
  ll curPrime=prime_show[i];
  ll base=curPrime*curPrime;
  if(base<L){base=((L+curPrime-1)/curPrime)*curPrime;}
  for(ll j=base ; j<=R ; j+=curPrime) isPrime[j-L]=false;
}
for(int i=0 ; i<=R-L ; i++){if(isPrime[i]==true)
segsiv.PB(L+i);}
}
//---------------- seive end ----------------------------
vector<ll> prime_factorization(ll n){ // n number take kon
kon prime numer dara vag jai
vector<ll> fector;
for(auto u : prime_show){if(1LL*u*u > n)
break;if(!(n%u)){while(!(n%u)){fector.PB(u);n/=u;}}}if(n
>1) fector.PB(n);
return fector;
}
```

```
ll NOD(ll n){ // number of divisiors of a number ?Time :
O(n * (sqrt n)/ln n)
ll nod=1;
for(auto u:prime_show){if(1LL*u*u>n) break;if(!(n%u)){ll
cnt=0;while(!(n%u)){n/=u;cnt++;}cnt++;nod*=cnt;}}
if(n>1){nod*=2;}return nod;
}
ll DSF(ll n){ // sum of number of divisors of a given
number (1 to n) // DFS: divisor summetory
function.. ?time: O(sqrt n)
  ll x=sqrt(n),sum=0;lup_1(1,x) sum+=(n/i)-i;sum*=2;
sum+=x;
  return sum;
}
ll SOD(ll n) {// Sum of devisors of an number
  ll sod=1;
  for(auto u:prime_show){if(1LL*u*u>n) break;
    if(!(n%u)){ll
sum=1,a=1;while(!(n%u)){a*=u;sum+=a;n/=u;}sod*=sum;
}}
  if(n>1){sod*=(n+1);}
 return sod;
}
ll eulerphi(ll n){ // n er asthe 1 theke n obdi emon koto
gulo songkha ase jar sathe n er GCD 1 gcd(n,x)=1 or
calculate the numbe of co prime
  ll phi = n; for(auto u:prime_show){if(1LL*u*u>n) break;
// time complexity O(sqrt(n)/ln(sqrt n))
    if(!(n%u)){while(!(n%u)){n/=u;}phi/=u;phi*=(u-
1);}}if(n>1){phi/=n;phi*=(n-1);} return phi;
}
const  ll p=5e6+123;
unsigned long long phi[p],ans[p];
void eulerphi_using_harmonic(ll x){
  for(int i=1;i<=x;i++) phi[i]=i;
  for(auto u:prime_show){
 for(int i=u;i<=x;i+=u){phi[i]/=u;phi[i]*=(u-1);}}
}
// summetion of coprimes of a singel given number is
(phi[n]*n)/2
void lcm_sum(ll x){
  for(int i=1;i<=x;i++){
    for(int j=i;j<=x;j+=i){
      ans[j]+=(phi[i]*i);
    }
  }
}
ll pow(ll x, ll n) // optimize power function log(n) //
power of x
{
  ll result = 1;
  while (n > 0) {
    if (n & 1LL == 1) // y is odd
    {
      result = (result * x);
    }
    x = (x * x);
    n = n >> 1LL; // y=y/2;
  }
  return result;
}
// comparator for sort
bool cmp(const pair<int,int> &p1 ,const pair<int ,int>
&p2)
{
  if(p1.first>p2.first) return true;
  else if(p1.first==p2.first) return (p1.second<p2.second);
  return false;
}
```

```
//::::::::::::::::::::::::::::::::::::::::GlitcH::::::::::::::::::::::::::::::::
int32_t main()
{
   file();
   optimize();
   ll x=1e7;
   seive(x);
   // for(int  i=0;i<segsiv.size();i++) cout<<segsiv[i]<<' ';
cout<<endl;
   // while(1){
   //    ll n;
   //    cin>>n;
   //    if(n==0) break;
   //    if(n==1) cout<<0<<endl;
   //    else cout<<eulerphi(n)<<endl;
   // }
   /* lcm sum code
   ll n;
      cin>>n;
      ll sum=ans[n]+1;
      sum*=n;
      sum/=2;
      cout<<sum<<endl;
   */
   // ll q=1e6;
   // seive(q);
   // vector<ll> fec=prime_factorization(12);
   // for(auto u:fec) cout<<u<<' ';

/*
   vector <pair<int,int>> v;
   sort(v.begin(),v.end(),cmp);
   */
   // int t;
 // cin >> t;
   // while (t--)

   // {

   // }
}
// lembda function

//       function<ll(ll,ll)>dfs = [&] (ll i,ll p){
//       }; dfs(1,0);

// gp_hash_table<int,int> mp; // faster than unordered
map
```