Totient

```
//for n

ll totient(ll n){
   ll s=n;
   for(ll i=2;i<=n/i;i++){
      if(n%i==0){
         while(n%i==0) n/=i;
         s-=s/i;
      }
   }
   if(n>1) s-=s/n;
   return s;
}
//////////////////
//for 1-n

ll ar[200010];
void totient(){
   ll n=200000;
   for(ll i=1;i<=n;i++) ar[i]=i;
   for(ll i=2;i<=n;i++){
      if(ar[i]==i){
         ar[i]=i-1;
         for(ll j=i*2;j<=n;j+=i) ar[j]=(ar[j]/i)*(i-1);
      }
   }
}
```

Divisor

```
ll sod[100010],nod[100010];
vll alldiv[100010];

int main(){
//   freopen("input.txt","r",stdin);
   ll a,b,c=0,i,j,t,k,lie,m,n,o,x,y,z;
   for(i=1;i<=100000;i++){
      for(j=i;j<=100000;j+=i){
         nod[j]++;
         sod[j]+=i;
         alldiv[j].pb(i);
      }
   }
   while(S(n)==1){
      printf("Number of divisor is -> %lld\n",nod[n]);
      printf("Sum of divisor is -> %lld\n",sod[n]);
      printf("all divisors are --> ");
      for(i=0;i<zz(alldiv[n]);i++) printf("%lld ",alldiv[n][i]);
      printf(nn);
   }

   return 0;
}
```

Sieve

```
bool ar[1000010];
vll v;
void sieve(ll n){
   ll i,j;
   v.pb(2LL);
   for(i=3;i<n;i+=2){
      if(ar[i]==0){
         v.pb(i);
         if(i>n/i) continue;
         for(j=i*i;j<=n;j+=(i+i)) ar[j]=1;
      }
   }
}
```

Big Mod

```
ll big_mod(ll b,ll p,ll m){
   if(p==0) return 1;
   if(p%2==0){
      ll s=big_mod(b,p/2,m);
      return ((s%m)*(s%m))%m;
   }
   return ((b%m)*(big_mod(b,p-1,m)%m))%m;
}

ll mod_inv(ll b,ll m){
   return big_mod(b,m-2,m);
}
```

SPF

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

ll spf[1000010];  //smallest prime factor

void spfgen(){
   ll i,j;
   for(i=1;i<1000010;i++){
      if(i%2==0) spf[i]=2LL;
      else spf[i]=i;
   }
   for(i=3;i*i<1000010;i+=2){
      if(spf[i]==i){ // if prime
         for(j=i*i;j<1000010;j+=i){
            if(spf[j]==j) spf[j]=i;
         }
      }
   }
}
ll ar[1000010];
int main(){
//   freopen("input.txt","r",stdin);
   ll a,b,i,j,t,k,lie,m,n,o,x,y,z;
```

```
    spfgen();
    while(scanf("%lld",&n)==1){
        while(n>1){
            x=spf[n];
            cout<<x<<' ';
            while(n%x==0){
                n/=x;
            }
        }
        cout<<endl;
    }

    return 0;
}
```

big number division to find reminder

```
ll mod(string num,ll a)
{
    ll res=0;
    for(ll i=0;i<zz(num);i++)
         res=(res*10+(ll)num[i]-'0')%a;
    return res;
}
```

NcR%p

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

unsigned long long power(unsigned long long x,
                                  int y, int p)
{
    unsigned long long res = 1;

    x = x % p;
    while (y > 0)
    {
        if (y & 1)
            res = (res * x) % p;
        y = y >> 1;
        x = (x * x) % p;
    }
    return res;
}

unsigned long long modInverse(unsigned long long n,
                                           int p)
{
    return power(n, p - 2, p);
}

unsigned long long nCrModPFermat(unsigned long long n,
                                 int r, int p)
{
    if (n < r)
        return 0;
    if (r == 0)
        return 1;
    unsigned long long fac[n + 1];
    fac[0] = 1;
    for (int i = 1; i <= n; i++)
        fac[i] = (fac[i - 1] * i) % p;

    return (fac[n] * modInverse(fac[r], p) % p
            * modInverse(fac[n - r], p) % p)
           % p;
}

int main()
{
    int n = 10, r = 2, p = 13;
    cout << "Value of nCr % p is "
         << nCrModPFermat(n, r, p);
    return 0;
}
```

Binary Search

```
ll Binay_Search(ll l,ll r,ll x){
    while(l<=r){
        ll mid=(l+r)/2;
        if(ar[mid]==x) return mid;
        else if(ar[mid]<x) l=mid+1;
        else r=mid-1;
    }
    return -1;
}


ll Lower_Bound(ll l,ll r,ll x){
    while(l<r){
        ll mid=(l+r)/2;
        if(ar[mid]>=x) r=mid;
        else l=mid+1;
    }
    return l;
}


ll Upper_Bound(ll l,ll r,ll x){
    while(l<r){
        ll mid=(l+r)/2;
        if(ar[mid]<=x) l=mid+1;
        else r=mid;
    }
    return l;
}
```

Extended Euclidean Algorithm

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

int gcdExtended(int a, int b, int *x, int *y)
{
    if (a == 0)
    {
        *x = 0;
        *y = 1;
        return b;
    }

    int x1, y1;
    int gcd = gcdExtended(b%a, a, &x1, &y1);
    *x = y1 - (b/a) * x1;
    *y = x1;

    return gcd;
}

int main()
{
    int x, y, a = 35, b = 15;
    int g = gcdExtended(a, b, &x, &y);
    cout << "GCD(" << a << ", " << b
         << ") = " << g << endl;
    return 0;
}
```

Segmented Sieve

```
vll v;
bool ar[1000010];

void sieve(){
    ll i,j,n=1000010;
    v.pb(2LL);
    for(i=3;i<=n;i+=2){
        if(ar[i]==0){
            v.pb(i);
            if(i>n/i) continue;
            for(j=i*i;j<=n;j+=(i+i)) ar[j]=1;
        }
    }
}

void segmented_sieve(ll L,ll R)
{
    ll c=0;
    bool isPrime[R-L+1];
    for(ll i=0;i<=R-L+1;i++)
        isPrime[i]=true;
    if(L==1)
        isPrime[0]=false;
    for(ll i=0;v[i]*v[i]<=R;i++){
        ll curPrime=v[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(ll i=0;i<=R-L;i++){
        if(isPrime[i]==true)
            c++;;
    }
    printf("%lld\n",c);
}

int main()
{
    sieve();
    ll l,r,t,c=0;
    S(t);
    while(t--){
        SS(l,r);
        printf("Case %lld: ",++c);
        segmented_sieve(l,r);
    }

    return 0;
}
```

Hash

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 100010
#define base 26
#define MOD 1000000007
#define ll long long

ll pH[MAX+5], po[MAX+5];
void preprocess(string &s){
    pH[0]=s[0]-'a'+1;
    po[0]=1;
    for(int i=1;i<(int)s.size();i++){
        pH[i]=((pH[i-1]*base)+(s[i]-'a'+1))%MOD;
        po[i]=po[i-1]*base%MOD;
    }
}

ll getHash(int L,int R){
    if(!L) return pH[R];
    return ((pH[R]-(pH[L-1]*po[R-L+1])%MOD)+MOD)%MOD;
}


int main(){
    string s;
    ll n,q;
    cin >> s;
    preprocess(s);
//    for(int i=0;i<s.size();i++)
//        cout<<pH[i]<<" ";
    cin>>q;
    while(q--){
        ll a,b;
        cin>>a>>b;
        cout<<getHash(a,b)<<endl;
    }

    return 0;
}
```

KMP

```
vll prefix_function(string s){
    ll n=zz(s);
    vll pf(n);
    for(ll i=1;i<n;i++){
        ll j=pf[i-1];
        while(j>0 && s[i]!=s[j]) j=pf[j-1];
        if(s[i]==s[j]) j++;
        pf[i]=j;
    }
    return pf;
}

int main(){
//    freopen("input.txt","r",stdin);
    ll a,b,c=0,i,j,t,k,lie,m,n,o,x=0,y=0,mx=0,z,ar[200010];
    string tx,pt;
    cin>>tx>>pt;
    vll lps;
    lps=prefix_function(pt);
//    for(i=0;i<zz(lps);i++) cout<<lps[i]<<ss;
    i=0;
    j=0;
    while(i<zz(tx)){
        if(pt[j]==tx[i]){
            i++;
            j++;
        }
        if(j==zz(pt)){
            printf("Found pattern at index %lld\n",i-j);
            j=lps[j-1];
        }
        else if(i<zz(tx) && pt[j]!=tx[i]){
            if(j!=0) j=lps[j-1];
            else i++;
        }
    }
    return 0;
}
```

Z

```
vll Zcal(string s){
    ll n=zz(s);
    vll z(n);
    for(ll 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;
}

int main(){
//    freopen("input.txt","r",stdin);
    ll a,b,c=0,i,j,t,k,lie,m,n,o,x,y,z,ar[200010];
    string s;
    cin>>s;
    vll Z;
    Z=Zcal(s);
    for(i=0;i<zz(Z);i++) cout<<Z[i]<<ss;

    return 0;
}

/*
to find pattern
s=pattern+"$"+text;
number of pattern_size value in z vector = number of match
*/
```

Trie

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

struct node{
    bool endmark;
    node *next[27];
    node(){
        endmark=false;
        for(int i=0;i<26;i++) next[i]=NULL;
    }
} *root;

void Insert(string s,int n){
    node *cur=root;
    for(int i=0;i<n;i++){
        int id=s[i]-'a';
        if(cur->next[id]==NULL) cur->next[id]=new node();
        cur=cur->next[id];
    }
    cur->endmark=true;
}

bool Search(string s,int n){
    node *cur=root;
    for(int i=0;i<n;i++){
        int id=s[i]-'a';
        if(cur->next[id]==NULL) return false;
        cur=cur->next[id];
    }
    return cur->endmark;
}

void Delete(node *cur){
    for(int i=0;i<26;i++){
        if(cur->next[i]) Delete(cur->next[i]);
    }
    delete(cur);
}

int main(){
    root=new node();
    int n,i,q;
    cin>>n;
    while(n--){
        string s;
        cin>>s;
        Insert(s,(int)s.size());
    }
    cin>>q;
    while(q--){
        string s;
        cin>>s;
        if(Search(s,(int)s.size())) puts("found");
        else puts("not found");
    }
    Delete(root);

    return 0;
}
```

order of two strings

```
/*
if tow strings are given,,
are the characters of the 1st string
maintain their order in the 2nd string........
*/

int main()
{
    ll a,b,c,i,j,t,k,lie,m,n,o,p,x,y,z;
    c=0;
    x=0;
    string s1,s2;
    cin>>s1>>s2;
    for(i=0;i<s1.length();i++){
        for(j=x;j<s2.length();j++){
            if(s1[i]==s2[j]){
                c++;
                x=j+1;
                break;
            }
        }
    }
    if(c==s1.length()) cout<<"Yes"<<nn;
    else cout<<"No"<<nn;

    return 0;
}
```

BFS

```
vll v[10];
ll visited[10];
ll level[10];

void bfs(ll u){
    visited[u]=1;
    level[u]=0;
    queue<ll>q;
    q.push(u);

    while(!q.empty()){
        u=q.front();
        visited[u]=1;
        printf("pop -> %lld\n",u);
        q.pop();
        visited[u]=1;

        for(ll i=0;i<zz(v[u]);i++){
            ll p=v[u][i];
            if(!visited[p]){
                printf("pushing -> %lld\n",p);
                visited[p]=1;
                level[p]=level[u]+1;
                q.push(p);
            }
        }
    }

}
```

DFS

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

vector<int> g[128];
bool seen[128];
int n, e;
```

```
void dfs(int u)
{
   seen[u] = true;
   printf("%d ", u);
   for(int i = 0; i < g[u].size(); i++) {
      int v = g[u][i];
      if(!seen[v]) dfs(v);
   }
}

int main()
{
   scanf("%d %d", &n, &e);
   for(int i=0; i<e; i++) {
     int u, v;
     cin >> u >> v;
     g[u].push_back(v);
   }
   for(int i=1; i<=n; i++) if(!seen[i]) dfs(i);

   return 0;
}
```

BIT

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

int ar[100010];
int tree[100010];

int query(int idx){
   int sum=0;
   while(idx>0){
      sum+=tree[idx];
      idx-=(idx&-idx);
   }
   return sum;
}

void update(int idx,int val,int n){
   while(idx<=n){
      tree[idx]+=val;
      idx+=(idx&-idx);
   }
}

void print(int *ar,int n){
   for(int i=1;i<=n;++i){
      cout<<ar[i]<<" ";
   }
   puts("");
}

int main(){
   int n,a,b;
   cin >> n;
   for(int i=1;i<=n;++i){
      cin>>ar[i];
      update(i,ar[i],n);
   }

   cout<<"input array - ";
   print(ar,n);

   cout<<"tree array - ";
   print(tree,n);
   cin>>a>>b;
   cout<<query(b)-query(a-1);

   return 0;
}

/*
5
4 3 5 6 1
input array - 4 3 5 6 1
tree array - 4 7 5 18 1
2 5
15
*/
```

Bridge

```
#include<bits/stdc++.h>
using namespace std;
vector<pair<int,int>>vp;
vector<int>v[707];
int in[707],low[707],vis[707],timer;

void dfs(int node,int par){
   vis[node]=1;
   in[node]=low[node]=timer;
   timer++;

   for(int child:v[node]){
      if(child==par) continue;
      if(vis[child]==1){
         low[node]=min(low[node],in[child]);
      }
      else{
         dfs(child,node);
         if(low[child]>in[node]){

vp.push_back(make_pair((min(node,child)),(max(node,child))
));
         }
         low[node]=min(low[node],low[child]);
      }
   }
}

int main(){
//   freopen("input.txt","r",stdin);
    int a,b,c=0,i,j,t,k,lie,m,n,o,x=0,y,z;
    scanf("%d",&t);
    while(t--){
        for(i=0;i<707;i++){
            v[i].clear();
            in[i]=0;
            low[i]=0;
            vis[i]=0;
        }
        vp.clear();
        timer=0;
        scanf("%d %d",&n,&m);
        while(m--){
            scanf("%d %d",&a,&b);
            v[a].push_back(b);
            v[b].push_back(a);
        }
        dfs(1,-1);
        sort(vp.begin(),vp.end());
        if(vp.size()==0) printf("Caso #%d\nNo Bridge\n",++c);
        else{
            printf("Caso #%d\n%d\n",++c,(int)vp.size());
            for(i=0;i<(int)vp.size();i++) printf("%d
%d\n",vp[i].first,vp[i].second);
        }
    }

    return 0;
}
```

Cycle Detection(directed graph)

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

vector<int>v[10010];
int visited[10010];

bool ans=true;
void dfs(int node)
{
   if(ans==false) return;
   visited[node]=1;
   for(int i=0;i<v[node].size();i++){
       if(visited[v[node][i]]==0) dfs(v[node][i]);
       else ans=false;
   }
    visited[node]=false;
}

int main(){
    int i,j,m,n,a,b;
    scanf("%d %d",&n,&m);
    while(m--){
        scanf("%d %d",&a,&b);
        v[b].push_back(a);
    }
    for(int i=0;i<n;i++){
//       memset(visited,0,sizeof(visited));
        if(visited[i]==0 && v[i].size()>0) dfs(i);
    }
    if(!ans) printf("cycle found\n");
    else printf("cycle not found\n");

    return 0;
}
```

Cycle Detection(undirected graph)

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

vector<int>v[10010];
int vis[10010];

bool dfs(int node,int par){
    vis[node]=1;
    for(int child:v[node]){
        if(!vis[child]){
            if(dfs(child,node)==true) return true;
        }
        else if(child!=par) return true;
    }
    return false;
}

int main(){
    int i,j,m,n,a,b;
    scanf("%d %d",&n,&m);
    while(m--){
        scanf("%d %d",&a,&b);
        v[a].push_back(b);
        v[b].push_back(a);
    }
    bool bb=dfs(1,-1);
    if(bb) printf("cycle found\n");
    else printf("cycle not found\n");

    return 0;
}
```

DSU

```
#include<bits/stdc++.h>
using namespace std;
int parent[1111],Rank[1111];
```

```
///path compression
int find_par(int a){
   if(parent[a]<0) return a;
   return parent[a]=find_par(parent[a]);
}

///union by rank
void marge(int a,int b){
   a=find_par(a);
   b=find_par(b);
   if(a==b) return;
   if(Rank[a]>=Rank[b]){
      /// a will be parent;
      parent[b]=a;
      Rank[a]+=Rank[b];
      parent[a]-=Rank[b];
   }
   else{
      /// b will be parent
      parent[a]=b;
      Rank[b]+=Rank[a];
      parent[b]-=Rank[a];
   }
}

int main(){
   int a,b,i,j,m,n;
   scanf("%d %d",&n,&m);
   for(i=1;i<=n;i++){
      parent[i]=-1;
      Rank[i]=1;
   }
   while(m--){
      scanf("%d %d",&a,&b);
      marge(a,b);
   }
   for(i=1;i<=n;i++){
      printf("for %d -> parent -> %d -- rank ->
%d\n",i,parent[i],Rank[find_par(i)]);
   }

   return 0;
}

/*
6 3
1 2
2 3
4 5
for 1 -> parent -> -3 -- rank -> 3
for 2 -> parent -> 1 -- rank -> 3
for 3 -> parent -> 1 -- rank -> 3
for 4 -> parent -> -2 -- rank -> 2
for 5 -> parent -> 4 -- rank -> 2
for 6 -> parent -> -1 -- rank -> 1
number of negative parent = number of connected
component
rank = size of connected component
the negative value of parent means
it is parent of itself and it is the
size of that connected component with absolute value.
*/
```

Kruskal

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

struct edge{
   int a,b,w;
};
edge ar[100010];
int par[10010];

int Find(int a){
   if(par[a]==-1) return a;
   return par[a]=Find(par[a]);
}

void Union(int a,int b){
   par[a]=b;
}

bool cmp(edge a,edge b){
   return a.w<b.w;
}

int main(){
   int i,j,a,b,n,m,sm=0;
   scanf("%d %d",&n,&m);
   for(i=1;i<=n;i++) par[i]=-1;
   for(i=0;i<m;i++){
      scanf("%d %d %d",&ar[i].a,&ar[i].b,&ar[i].w);
   }
   sort(ar,ar+m,cmp);
   for(i=0;i<m;i++){
      a=Find(ar[i].a);
      b=Find(ar[i].b);
      if(a!=b){
         sm+=ar[i].w;
         Union(a,b);
      }
   }
   printf("%d\n",sm);

   return 0;
}
```

LCA

```
vll v[300010];
ll lca[300010][20];
ll level[300010];

void dfs(ll node,ll lvl,ll par){
    level[node]=lvl;
    lca[node][0]=par;
    for(ll child:v[node]){
        if(child!=par){
            dfs(child,lvl+1,node);
        }
    }
}

void init(ll n){
    dfs(1,0,-1);
    for(ll i=1;i<=19;i++){
        for(ll j=1;j<=n;j++){
            if(lca[j][i-1]!=-1){
                ll par=lca[j][i-1];
                lca[j][i]=lca[par][i-1];
            }
        }
    }
}
ll get_lca(ll a,ll b){
    if(level[b]<level[a]) swap(a,b);
    ll d=level[b]-level[a];
    while(d>0){
        ll i=log2(d);
        b=lca[b][i];
        d-=(1LL<<i);
    }
    if(a==b) return a;
    for(ll i=19;i>=0;i--){
        if(lca[a][i]!=-1 and (lca[a][i])!=lca[b][i]){
            a=lca[a][i];
            b=lca[b][i];
        }
    }
    return lca[a][0];
}

int main()
{
//    freopen("input.txt","r",stdin);
    ll a,b,c,s,d,i,j,t,k,n,q;
    scanf("%lld",&n);
    mms(lca,-1);
    for(i=1;i<n;i++){
        SS(a,b);
        v[a].pb(b);
        v[b].pb(a);
    }
    init(n);
    S(q);
    while(q--){
        SS(a,b);
        P(get_lca(a,b));
    }

    return 0;
}
```

Mo

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

ll ar[100010],fre[100010],cnt=0,ans[100010];

struct query{
    ll l,r,i;
}Q[200010];

bool cmp(query a,query b){
    if(a.l!=b.l) return a.l<b.l;
    return a.r<b.r;
}

void add(ll pos){
    fre[ar[pos]]++;
    if(fre[ar[pos]]==1) cnt++;
}

void remov(ll pos){
    fre[ar[pos]]--;
    if(fre[ar[pos]]==0) cnt--;
}

int main()
{
    ll n,q,i,j;
    scanf("%lld",&n);
    for(i=0;i<n;i++) scanf("%lld",&ar[i]);
    scanf("%lld",&q);
    for(i=0;i<q;i++){
        scanf("%lld",&Q[i].l);
        scanf("%lld",&Q[i].r);
        Q[i].i=i;
        Q[i].l--;
        Q[i].r--;
    }
    sort(Q,Q+q,cmp);
    ll ml=0,mr=-1;
    for(i=0;i<q;i++){
        ll L=Q[i].l;
        ll R=Q[i].r;

        while(ml>L) ml--,add(ml);
        while(mr<R) mr++,add(mr);
        while(ml<L) remov(ml),ml++;
        while(mr>R) remov(mr),mr--;
        ans[Q[i].i]=cnt;
    }
    for(i=0;i<q;i++) printf("%lld\n",ans[Q[i].i]);

    return 0;
}
```

Segment Tree

```
ll ar[10000],tree[40000];

ll combine(ll l,ll r){
        return l<r?l:r;
}

void build(ll nd,ll st,ll ed){
    if(st==ed) tree[nd]=ar[st];
    else{
        ll mid=(st+ed)/2;
        build(2*nd,st,mid);
        build(2*nd+1,mid+1,ed);
        tree[nd]=combine(tree[2*nd],tree[2*nd+1]);
    }
}

void update(ll nd,ll st,ll ed,ll id,ll val){
    if(id<st || id>ed) return;
    if(st==ed){
        ar[id]+=val;
        tree[nd]+=val;
    }
    else{
        ll mid=(st+ed)/2;
        if(st<=id and id<=mid) update(2*nd,st,ed,id,val);
        else update(2*nd+1,mid+1,ed,id,val);
        tree[nd]=combine(tree[2*nd],tree[2*nd+1]);
    }
}

ll query(ll nd,ll st,ll ed,ll l,ll r){
    if(r<st or ed<l) return OO;
    if(l<=st and ed<=r) return tree[nd];
    ll mid=(st+ed)/2;
    ll p1=query(2*nd,st,mid,l,r);
    ll p2=query(2*nd+1,mid+1,ed,l,r);
    return combine(p1,p2);
}

int main(){
//    freopen("input.txt","r",stdin);
    ll a,b,c=0,i,j,t,k,lie,m,n,o,x,y,z;
    S(n);
    for(i=1;i<=n;i++){
        S(ar[i]);
    }
    build(1,1,n);
    cout<<query(1,1,n,3,5);

    return 0;
}
```

Sqrt Decomposition

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

ll ar[10000],block[100],bs,input[10000]; // bs->block size

void update(ll idx,ll val){
    ll bn=idx/bs; // block number
    block[bn]+=val-ar[idx];
    ar[idx]+=val;
}

ll query(ll l,ll r){
    ll sum=0;
    while(l<r and l%bs!=0 and l!=0){
        sum+=ar[l];
        l++;
    }
    while(l+bs<=r){
        sum+=block[l/bs];
        l+=bs;
    }
    while(l<=r){
        sum+=ar[l];
        l++;
    }
    return sum;
}

void preproces(ll n){
    ll blk_idx=-1;
    bs=sqrt(n);
    for(ll i=0;i<n;i++){
        ar[i]=input[i];
        if(i%bs==0) blk_idx++;
        block[blk_idx]+=ar[i];
    }
}

int main(){
//    freopen("input.txt","r",stdin);
```

```
    ll a,b,c=0,i,j,t,k,lie,sm=0,m,n,o,x,y,q,z;
    scanf("%lld",&n);
    for(i=0;i<n;i++) scanf("%lld",&input[i]);
    preproces(n);
    for(i=0;i<sqrt(n);i++) cout<<block[i]<<' ';
    scanf("%lld",&q);
    while(q--){
        scanf("%lld %lld",&a,&b);
        printf("%lld\n",query(a,b));
    }

    return 0;
}
```

Topological Sort DFS

```
vll v[111];
bool visited[111];
stack<ll>ts;

void dfs(ll u){
    visited[u]=true;
    for(ll i=0;i<zz(v[u]);i++){
        if(!visited[v[u][i]]) dfs(v[u][i]);
    }
    ts.push(u);
}

int main(){
//    freopen("input.txt","r",stdin);
    ll a,b,c,i,j,t,vrtx,edg,k,lie,m,n,o,x,y,z,ar[200010];
    while(SS(edg,vrtx) && vrtx||edg){
        while(vrtx--){
            SS(a,b);
            v[a].pb(b);
        }
        memset(visited,0,sizeof(visited));
        for(i=1;i<=edg;i++){
            if(!visited[i]) dfs(i);
        }
        while(!ts.empty()){
            if(zz(ts)==1) printf("%lld\n",ts.top());
            else printf("%lld ",ts.top());
            ts.pop();
        }
        for(i=0;i<111;i++) v[i].clear();
    }

    return 0;
}
```

Topological_Sort BFS

```
#include<bits/stdc++.h>
using namespace std;
vector<int>v[1111];
vector<int>ans;
int in[1111];

void Kahn(int n){
    queue<int>q;
    for(int i=1;i<=n;i++){
        if(in[i]==0) q.push(i);
    }
    while(!q.empty()){
        int cur=q.front();
        ans.push_back(cur);
        q.pop();
        for(int node : v[cur]){
            in[node]--;
            if(!in[node]) q.push(node);
        }
    }
}

int main()
{
    int n,t,i,j,m,a,b;
    scanf("%d %d",&n,&m);
    while(m--){
        scanf("%d %d",&a,&b);
        v[a].push_back(b);
        in[b]++;
    }
    Kahn(n);
    for(i : ans) printf("%d ",i);

    return 0;
}
```

Bellmanford

```
#include<bits/stdc++.h>
using namespace std;
int dis[100];
vector<int>G[100];
int cost[100][100];
int flag = 0;
int n,m;
void zero()
{
    for(int i=0; i<n+2; i++)
    {
        dis[i] = 999999;
    }
}
void Bellman_Ford(int s)
{
    int i,j,k,u,uc,v,uvc;
    dis[s] = 0;
    for(i = 0 ; i< n-1; i++)
    {
        for(j=0; j<n; j++)
        {
            for(k=0; k<G[j].size(); k++)
            {
                u = j;
                uc = dis[u];
                v = G[u][k];

                uvc = cost[u][v];

                if(uc+uvc<dis[v])
                {
                    dis[v] = uc+uvc;
                }
            }
        }
    }
    flag = 0;
    for(j=0; j<n; j++)
    {
        for(k=0; k<G[j].size(); k++)
        {
            u = j;
            uc = dis[u];
            v = G[u][k];
            uvc = cost[u][v];
            if(uc+uvc<dis[v])
            {
                flag = 1;
                break;
            }
        }
        if(flag==1)
            break;
    }
}

int main()
{
    int a,b,i,j,c,x,y;
    scanf("%d %d",&n,&m);
    zero();
    for(i=1; i<=m; i++)
    {
        scanf("%d %d %d",&a,&b,&c);
        G[a].push_back(b);
        cost[a][b] = c;
    }
    flag = 0;
    Bellman_Ford(1);
    if(flag == 1)
    {
        printf("negative cycle ditected\n");
    }
    else
        printf("\n\nCost %d\n",dis[3]);

    return 0;
}
```

Dikjstra

```
vll G[20020],cost[20020];
ll inf=1000000000,dt[20020];

struct data{
    ll city,dist;
    bool operator<(const data& p) const{
        return dist>p.dist;
    }
};
//bool operator<(data a,data b) {return a.dist>b.dist;}

ll dijkstra(ll source,ll destination){
    data u,v;
    priority_queue<data>pq;
    u.city=source;
    u.dist=0;
    pq.push(u);
    dt[source]=0;

    while(!pq.empty()){
    u=pq.top();
    pq.pop();
        for(ll i=0;i<zz(G[u.city]);i++){
            v.city=G[u.city][i];
            v.dist=cost[u.city][i]+dt[u.city];
            if(dt[v.city]>v.dist){
                dt[v.city]=v.dist;
                pq.push(v);
            }
        }
    }
    return dt[destination];
}

int main(){
//    freopen("input.txt","r",stdin);
    ll c,a,b,i,j,t,k,lie,m,n,o,x,y,z=0,ar[200010];
    S(t);
    while(t--){
        for(i=0;i<20020;i++){
        dt[i]=inf;
        G[i].clear();
        cost[i].clear();
    }
        ll nd,ed,st,dt;
```

```
      SS(nd,ed);
      SS(st,dt);
      for(i=0;i<ed;i++){
         SSS(a,b,c);
         G[a].pb(b);
         G[b].pb(a);

         cost[a].pb(c);
         cost[b].pb(c);
      }
      x=dijkstra(st,dt);
      if(x==inf) printf("Case #%lld: unreachable\n",++z);
      else printf("Case #%lld: %lld\n",++z,x);
   }

   return 0;
}
```

Flood Fill

```
ll fx[]={ 1, -1,  0,  0};   // 4 direction
ll fy[]={ 0,  0,  1, -1};  // 4 direction
ll row,column,cnt;
char ch[22][22];

void flood_fill(ll i,ll j){
   if(i<0 || j<0 || i>row-1 || j>column-1) return;
   if(ch[i][j]=='.'){
      ch[i][j]='*';
      cnt++;
      for(ll k=0;k<4;k++){
         ll x=i+fx[k];
         ll y=j+fy[k];
         flood_fill(x,y);
      }
   }
}

int main()
{
   ll i,j,b,c=0,t,a,p;
   S(t);
   while(t--){
      cnt=0;
      p=0;
      SS(a,b);
      row=b;
      column=a;
      getchar();
      for(i=0;i<row;i++){
         for(j=0;j<column;j++) scanf(" %c",&ch[i][j]);
      }
      for(i=0;i<row;i++){
         for(j=0;j<column;j++){
            if(ch[i][j]=='@'){ // start from "@" sign, find all "."
sign;
               ch[i][j]='.';
               flood_fill(i,j);
               p=1;
               break;
            }
         }
         if(p==1) break;
      }
      printf("Case %lld: %lld\n",++c,cnt);
   }

   return 0;
}
```

All Possible Sub-Array

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

int main(){
   ll a,b,m,n,i,j,k,x,y,t,c=0,ar[100];

   scanf("%lld",&n);
   vector<ll>v[1<<n+5];
   for(i=0;i<n;i++) scanf("%lld",&ar[i]);
   for(i=0;i<(1<<n);i++){
      for(j=0;j<n;j++){
         if(i&(1<<j)) v[i].push_back(ar[j]);
      }
   }
   for(i=0;i<(1<<n);i++){
      for(j=0;j<(ll)v[i].size();j++){
         printf("%lld ",v[i][j]);
      }
      printf("\n");
   }

   return 0;
}
```

All subTree size

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

vector<int>v[10010];
int vis[10010],sub_tree_size[10010];

int dfs(int node){
   vis[node]=1;
   int cur_size=1;
   for(int child:v[node]){
      if(!vis[child]) cur_size+=dfs(child);
```

```
    }
    sub_tree_size[node]=cur_size;
    return cur_size;
}

int main(){
    int i,j,m,n,a,b;
    scanf("%d %d",&n,&m);
    while(m--){
        scanf("%d %d",&a,&b);
        v[a].push_back(b);
        v[b].push_back(a);
    }
    dfs(1);
    for(i=1;i<=n;i++)
        printf("subtree for node %d is ->
%d\n",i,sub_tree_size[i]);

    return 0;
}
```

BFS in 2D grid

```
ll fx[]={ 1, -1,  0,  0};   // 4 direction
ll fy[]={ 0,  0,  1, -1};  // 4 direction

ll matrx[1001][1001];
ll visited[1001][1001];
ll level[1001][1001],R,C;
vll v[10];

void bfs(ll x,ll y){
    visited[x][y]=1;
    level[x][y]=0;
    queue<ll>q;
    q.push(x);
    q.push(y);

    while(!q.empty()){
        ll x1=q.front();
        q.pop();
        ll y1=q.front();
        q.pop();
        for(ll i=0;i<4;i++){
            ll x2=x1+fx[i];
            ll y2=y1+fy[i];

            if((x2>=0&&x2<=R) && (y2>=0&&y2<=C) &&
(matrx[x2][y2]==0)){
                if(!visited[x2][y2]){
                    visited[x2][y2]=1;
                    q.push(x2);
                    q.push(y2);
                    level[x2][y2]=level[x1][y1]+1;
                }
            }
        }
    }
}

int main()
{
    ll a,b,c,i,j,n,s,t,k,ro,num,r,x1,y1,x2,y2;
    while(SS(R,C)&&R||C){
        mms(matrx,0);
        mms(visited,0);
        mms(level,0);
        S(n);
        while(n--){
            SS(ro,num);
            while(num--){
                S(a);
                matrx[ro][a]=1;
            }
        }
        SS(x1,y1);
        SS(x2,y2);
        bfs(x1,y1);
        printf("%lld\n",level[x2][y2]);
    }

    return 0;
}
```

direction array

```
ll fx[]={ 1, -1,  0,  0};   // 4 direction
ll fy[]={ 0,  0,  1, -1};  // 4 direction

ll fx[]={ 0,  0,  1, -1, -1,  1, -1,  1};  // King's Move / 8 direction
ll fy[]={-1,  1,  0,  0,  1,  1, -1, -1}; // King's Move / 8 direction

ll fx[]={-2, -2, -1, -1,  1,  1,  2,  2};  // Knight's Move
ll fy[]={-1,  1, -2,  2, -2,  2, -1,  1}; // Knight's Move
```

find the path—BFS

```
vll v[10];
ll visited[10],pt[10];
ll level[10];

void bfs(ll u){
    visited[u]=1;
    level[u]=0;
    queue<ll>q;
    q.push(u);
    pt[u]=u;

    while(!q.empty()){
        u=q.front();
        visited[u]=1;
        q.pop();
        visited[u]=1;

        for(ll i=0;i<zz(v[u]);i++){
            ll p=v[u][i];
            if(!visited[p]){
                visited[p]=1;
                level[p]=level[u]+1;
                pt[p]=u;
                q.push(p);
            }
        }
    }

}

int main()
{
    ll a,b,c,i,j,s,t,k,node,edge;
    SS(node,edge);
    for(i=1;i<=edge;i++){
        SS(a,b);
        v[a].pb(b);
        v[b].pb(a);
    }
    S(s); // source
    bfs(s);
    S(t); // destination
    if(visited[t]==0) printf("no path\n");
    else{
        vll path;
        path.pb(t);
        ll now=t;
        while(now!=s){
            now=pt[now];
            path.pb(now);
        }
        reverse(all(path));
        for(i=0;i<zz(path);i++) cout<<path[i]<<ss;
    }

    return 0;
}
```

in-out time

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

vector<int>v[10010];
int vis[10010],in[10010],out[10010],timer=1;

void dfs(int node){
    vis[node]=1;
    in[node]=timer++;
    for(int child:v[node]){
        if(!vis[child]) dfs(child);
    }
    out[node]=timer++;
}

int main(){
    int i,j,m,n,a,b;
    scanf("%d %d",&n,&m);
    while(m--){
        scanf("%d %d",&a,&b);
        v[a].push_back(b);
        v[b].push_back(a);
    }
    dfs(1);
    for(i=1;i<=n;i++)
        printf("for %d -- in -> %d ,, out -> %d\n",i,in[i],out[i]);

    return 0;
}
```

PBDS

```
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
using namespace std;
using namespace __gnu_pbds;
typedef    long long      ll;
typedef    vector<ll>     vll;
typedef
tree<ll,null_type,less<ll>,rb_tree_tag,tree_order_statistics_n
ode_update> ordered_set;

// *os.find_by_order(k) ; kth element
//  os.order_of_key(k) ; number of element less than k
```

Maximum Subarray Sum

```
int maxSubArray(vector<int>& v) {
    int i,j,n=v.size();
    int ans=v[0];
    for(i=1;i<n;i++){
        if(v[i-1]+v[i]>v[i]){
            v[i]=v[i-1]+v[i];
        }
        ans=max(ans,v[i]);
    }
    return ans;
}
```

Polygon Area

(X[i], Y[i]) are coordinates of i'th point.

```
double polygonArea(double X[], double Y[], int n)
{
    // Initialize area
    double area = 0.0;
```

```
  // Calculate value of shoelace formula
  int j = n - 1;
  for (int i = 0; i < n; i++)
  {
    area += (X[j] + X[i]) * (Y[j] - Y[i]);
    j = i;  // j is previous vertex to i
  }

  // Return absolute value
  return abs(area / 2.0);
}
```

Circumcentre of Triangle

```
#define pdd pair<double, double>
void lineFromPoints(pdd P, pdd Q, double &a,double &b,
double &c){
        a = Q.second - P.second;
        b = P.first - Q.first;
        c = a*(P.first)+ b*(P.second);
}
// Function which converts the input line to its
// perpendicular bisector. It also inputs the points
// whose mid-point lies on the bisector
void perpendicularBisectorFromLine(pdd P, pdd Q, double
&a, double &b, double &c){
        pdd mid_point = make_pair((P.first +
Q.first)/2,(P.second + Q.second)/2);
        // c = -bx + ay
        c = -b*(mid_point.first) + a*(mid_point.second);
        double temp = a;
        a = -b;
        b = temp;
}
// Returns the intersection point of two lines
pdd lineLineIntersection(double a1, double b1, double c1,
double a2, double b2, double c2){
        double determinant = a1*b2 - a2*b1;
        if (determinant == 0){
                // The lines are parallel. This is simplified
                return make_pair(FLT_MAX, FLT_MAX);
        }
        else{
                double x = (b2*c1 - b1*c2)/determinant;
                double y = (a1*c2 - a2*c1)/determinant;
                return make_pair(x, y);
        }
}
pdd findCircumCenter(pdd P, pdd Q, pdd R){
        // Line PQ is represented as ax + by = c
        double a, b, c;
        lineFromPoints(P, Q, a, b, c);
        // Line QR is represented as ex + fy = g
        double e, f, g;
        lineFromPoints(Q, R, e, f, g);
        // Converting lines PQ and QR to perpendicular
        // vbisectors. After this, L = ax + by = c
        // M = ex + fy = g
        perpendicularBisectorFromLine(P, Q, a, b, c);
        perpendicularBisectorFromLine(Q, R, e, f, g);
        // The point of intersection of L and M gives
        // the circumcenter
        pdd circumcenter = lineLineIntersection(a, b, c, e, f,
g);
        if (circumcenter.first == FLT_MAX &&
circumcenter.second == FLT_MAX){
                //not a triangle
        }
        else
                return circumcenter;
}
```

Incenter of a triangle

```
pair<int,int> findInCenter(int x1,int x2,int x3,int y1,int y2,int
y3,double a,double b,double c)

  double x = (a*x1 + b*x2 + c*x3)/(a + b + c);
  double y = (a*y1 + b*y2 + c*y3)/(a + b + c);

  return make_pair(x,y);
}
```

Distance between 2 points

```
float distance(int x1, int y1, int x2, int y2)
{
  return sqrt(pow(x2 - x1, 2)+pow(y2 - y1, 2)*1.0);
}
```

Distance between 2 points 3D

```
double distance(float x1,float y1,float z1, float x2,float y2,
float z2)
{
  float d = sqrt(pow(x2 - x1, 2)+pow(y2 - y1, 2)+pow(z2 - z1,
2)*1.0);
  return d;
}
```

LIS with path printing

```
#define MAX_N 20
#define EMPTY_VALUE -1

int mem[MAX_N];
int next_index[MAX_N];

int f(int i, vector<int> &A) {
  if (mem[i] != EMPTY_VALUE) {
    return mem[i];
  }

  int ans = 0;
  for (int j = i + 1;j < A.size();j++) {
    if (A[j] > A[i]) {
        int subResult = f(j, A);
        if (subResult > ans) {
          ans = subResult;
          next_index[i] = j;
        }
      }
    }

    mem[i] = ans + 1;
    return mem[i];
}

vector<int> findLIS(vector<int> A){
  int ans = 0;

  for(int i = 0;i<A.size();i++) {
    mem[i] = EMPTY_VALUE;
    next_index[i] = EMPTY_VALUE;
  }

  int start_index = -1;

  for(int i = 0;i<A.size();i++) {
    int result = f(i, A);
    if (result > ans) {
      ans = result;
      start_index = i;
    }
  }
  vector<int> lis;
  while(start_index != -1) {
    lis.push_back(A[start_index]);
    start_index = next_index[start_index];
  }
  return lis;
}
```

LCS iterative

```
int lcsIterative(string S, string W) {
    int n = S.size();
    int m = W.size();
    for (int i = 0;i < n;i++) mem[i][m] = 0;
    for (int j = 0;j < m;j++) mem[n][j] = 0;
    for (int i = n - 1; i >= 0; i--) {
      for (int j = m - 1; j >= 0; j--) {
        if (S[i] == W[j]) {
          mem[i][j] = mem[i + 1][j + 1] + 1;
        } else {
          mem[i][j] = max(mem[i + 1][j], mem[i][j + 1]);
        }
      }
    }
    return mem[0][0];
}
```

Coin change DP

```
#define MAX_N 20
#define MAX_W 10000

#define INF 99999999
#define EMPTY_VALUE -1

int C[MAX_N];
int mem[MAX_N][MAX_W];
int n;

int f(int i, int W) {
    if (W < 0) return INF;
    if (i == n) {
      if (W == 0) return 0;
      return INF;
    }
    if (mem[i][W] != EMPTY_VALUE) {
      return mem[i][W];
    }

    int res_1 = 1 + f(i + 1, W - C[i]);
   // int res_1 = 1 + f(i, W - C[i]); //if take the same coin again
    int res_2 = f(i + 1, W);

    mem[i][W] = min(res_1, res_2);

    return mem[i][W];
}
```

Coin change (multiple choice) Optimized

```
#define MAX_N 20
#define MAX_W 10000
#define INF 99999999
#define EMPTY_VALUE -1
int C[MAX_N];
int mem[MAX_W];
int n;
int f_optimized(int W) {
    if (W < 0) return INF;
    if (W == 0) return 0;
    if (mem[W] != EMPTY_VALUE) {
      return mem[W];
    }
    int ans = INF;
    for (int i = 0;i < n;i++) {
      ans = min(ans, 1 + f_optimized(W - C[i]));
    }
    mem[W] = ans;
    return mem[W];
}
```

Important Formula

cylinder:
v=pi*r*r*h
sa=2*pi*r*(h+r)
curve surface area=2*pi*r*h

Trapezium:
a=(1/2)*(a+b)*h
p=a+b+c+d

sphere:
v=(4/3)*pi*r*r*r
p=4*pi*r*r
sa=4*pi*r*r

cone:
v=(1/3) pi * r *r*h
SA=pi*r*l+pi*r*r

cylinder:
v=pi*r*r*h
sa=2*pi*r*(h+r)
curve surface area=2*pi*r*h

cube:
v=s*s*s
p=12*s

square:
a=s*s
p=4s

Klite:
area=(0.5)*(a*b) [a =diagonal]
p=2*x+2*y [x,y side]

prism:
v=area of cross-section * length
or
v=(1/3)*(base area)*h

sin rule:
sinA/a=sinB/b
a/sinA=b/sinB
cos rule:
a^2=b^2+c^2-2*b*c*cos(A)
cosA=(b^2+c^2-a^2)/(2*b*c)

Sjr template

```
#include<bits/stdc++.h>
using namespace std;
typedef     long long     ll;
typedef     vector<ll>    vll;
#define     ss          ' '
#define     nn          "\n"
#define     fi          first
#define     se          second
#define     PB          pop_back
#define     pb          push_back
#define     pi          acos(-1.0)
#define     gcd(a,b)    __gcd(a,b)
#define     OO          1000000007
#define     NN          printf("\n")
#define     zz(v)       (ll)v.size()
#define     lcm(a,b)    (a*b)/gcd(a,b)
#define     no          printf("NO\n")
#define     mo          printf("-1\n")
#define     yes         printf("YES\n")
#define     S(a)        scanf("%lld",&a)
#define     all(p)      p.begin(),p.end()
#define     P(a)        printf("%lld\n",a)
#define     db          printf("be steady\n")
#define     mms(ar,a)   memset(ar,a,sizeof(ar))
#define     SS(a,b)     scanf("%lld %lld",&a,&b)
#define     PP(a,b)     printf("%lld %lld\n",a,b)
#define     prv(v)      for(auto it:v) cout<<it<<ss;NN;
#define     SSS(a,b,c)  scanf("%lld %lld %lld",&a,&b,&c)
ll Set(ll N,ll pos){return N=N | (1LL<<pos);}
bool check(ll N,ll pos){return (bool)(N & (1LL<<pos));}

ll ar[200010];
void solve(){
   ll a=1,b=0,c=0,n,i,j,k,m,o,x,y,z;
   ll p=0,sm=0,cnt=0,mx=-OO,mn=OO;

}

int main(){
//   freopen("input.txt", "r", stdin);
   ll t;
   S(t);
   while(t--)
      solve();

   return 0;
}
```

Jahin template

```
#include<algorithm>
#include<iostream>
#include<stdlib.h>
#include<stdio.h>
#include<utility>
#include<math.h>
#include<vector>
#include<string>
#include<queue>
#include<set>
#include<map>
using namespace std;

#define li       long long int
#define uli      unsigned long long int
#define test()   int t;cin>>t;while(t--)
#define fast()
ios_base::sync_with_stdio(false);cin.tie(NULL);
```

```
#define pcs        cout<<"Case "<<tk<<": "
#define mx          100005
#define mx2         200005
#define md          1000000007
#define spc         ' '
#define nn          "\n"



typedef struct{
    li x;
    li y;
    li z;
}mymy;

int cmp(mymy a, mymy b){
    if(a.x!=b.x)
        return a.x<b.x;
    return a.y<b.y;
}

void inline inout()
{
    #ifndef ONLINE_JUDGE
    freopen("D:/C programming/Online-judge-
solve/input.txt","r",stdin);
    freopen("D:/C programming/Online-judge-
solve/output.txt","w",stdout);
    #endif
}


void logic(){

}

int main()
{
    fast();
    inout();
    test()
        logic();

    return 0;
}
```