1 BFS

```
#include<bits/stdc++.h>
using namespace std;
constexpr int INF= 100000000;
constexpr int MX_N = 100000;
int dist[MX_N];
vector<int> adjList[MX_N];

int main(){
    for(int i = 0; i < MX_N; i++)
        dist[i]=INF;
    queue<int> q;
    q.push(0);
    dist[0] = 0;
    while(!q.empty()){
        int u = q.front(); q.pop();
        int d = dist[u];
        for(int i : adjList[u]){
            if(dist[i]==INF){
                dist[i]=d+1;
                q.push(i);
            }
        }
    }
    return 0;
}
```

2 Convex Hull

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

struct point{
    int x,y;
    point(int _x, int _y) : x(_x), y(_y) {}
    point() {}

    point operator+ (const point &p){
        return point(x+p.x,y+p.y);
    }

    point operator- (const point &p) const{
        return point(x-p.x,y-p.y);
    }

    bool operator== (const point& p) const {
        return x==p.x && y==p.y;
    }
};

point anchor;
constexpr int MX_N=100001;
int N;
point vertices[MX_N];

double dist(const point& a, const point& b){
    return sqrt(pow(a.x-b.x,2) + pow(a.y-b.y,2));
}

int cross(const point& a, const point& b){
    return a.x*b.y - a.y*b.x;
}

bool angleCmp(const point& a, const point& b){
    point relA = a-anchor;
    point relB = b-anchor;
    if(cross(relA,relB)==0)
        return dist(a,anchor) > dist(b,anchor);
    return atan2(relA.y,relA.x) < atan2(relB.y, relB.x);
}

bool ccw(const point& a,const point& b, const point& c){
    if(a==b || b==c || a==c)
        return false;
    point u = b-c;
    point v = b-a;
    int cr = cross(u,v);
    return cr >= 0;
}

int main(){
    scanf("%d",&N);
    int x,y;
    for(int i = 0; i < N; i++)
        scanf("%d %d",&vertices[i].x,&vertices[i].y);
    int pos = 0;
    for(int i = 0; i < N; i++)
        if(vertices[i].y < vertices[pos].y || (vertices[i].y==vertices[pos].y && (vertices[i].x > vertices[pos].x)))
            pos = i;
    point _tp = vertices[0];
    vertices[0] = vertices[pos];
    vertices[pos] = _tp;
    anchor = vertices[0];
    sort(vertices+1,vertices+N, angleCmp);
    vector<point> hull;
    hull.push_back(vertices[N-1]);
    hull.push_back(vertices[0]);
    hull.push_back(vertices[1]);
    for(int i = 2; i < N;){
        int t = hull.size();
        if(ccw(hull[t-2],hull[t-1],vertices[i]))
            hull.push_back(vertices[i++]);
        else
            hull.pop_back();
    }
    return 0;
}
```

3 Dijkstras

```
dist[rs][cs] = 0;
```

```
priority_queue<path> q;
q.push(path(rs,cs,0));
while(!q.empty()){
    path p = q.top(); q.pop();
    ux = p.ux, uy=p.uy,d=p.d;
    if(dist[ux][uy] < d)
        continue;
    for(int i = 0; i < 8; ++i){
        nx = ux+moves[i][0];
        ny = uy+moves[i][1];
        nd = d+(board[ux][uy]!=i);
        if(valid(nx,ny) && dist[nx][ny] > nd){
            dist[nx][ny] = nd;
            q.push(path(nx,ny,nd));
        }
    }
}
printf("%d\n",dist[rd][cd]);
```

4 Fenwick

```
int tree[MX_N];
int N;

int lsOne(int i){
    return i&(-i);
}

void update(int k,int v){
    for(; k<MX_N; k+=lsOne(k))
        tree[k]+=v;
}

int query(int k){
    int cnt=0;
    for(; k; k-=lsOne(k)){
        cnt+=tree[k];
    }
    return cnt;
}
```

5 Inversion Count

```
#include<bits/stdc++.h>
using namespace std;
const int MX_N = 1000111;

int N;
int a[MX_N];
long long cnt=0;

void mergesort(int L, int R){
    if(L>=R)
        return;
    int mid = (L+R)/2;
    mergesort(L,mid);
    mergesort(mid+1,R);
    int n[R-L+1];
    int i = 0;
    int lp = L;
    int rp = mid+1;
    while(rp<=R || lp<=mid){
        if(rp<=R && lp <= mid){
            if(a[rp]<a[lp])
                n[i]=a[rp],rp++,cnt+=((long long) (mid-lp+1));
            else
                n[i]=a[lp],lp++;
        } else if(rp<=R){
            n[i]=a[rp++];
        } else {
            n[i]=a[lp++];
        }
        i++;
    }
    for(int j= L; j <= R; j++)
        a[j]=n[j-L];
}

int main(){
    scanf("%d",&N);
    for(int i = 0; i < N; i++)
        scanf("%d",&a[i]);
    mergesort(0,N-1);
    printf("%lld\n",cnt);
    return 0;
}
```

6 Maximum Flow

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

const long long MX_N=505,MX_M=10001,INF=1000000000;
long long N,M,S,T,f;
long long res[MX_N][MX_N];
long long graph[MX_N][MX_N];
long long dist[MX_N];
long long p[MX_N];

void aug(int u, long long minE){
    if(u==S){
        f=minE;
        return;
    }
    if(p[u]!=u){
        aug(p[u],min(minE,res[p[u]][u]));
        res[p[u]][u]-=f;
        res[u][p[u]]+=f;
```

```
    }
}

int main(){
    scanf("%lld %lld %lld %lld",&N,&M,&S,&T);
    long long u,v,c;
    for(int i = 0; i < M; i++){
        scanf("%lld %lld %lld",&u,&v,&c);
        graph[u][v] = res[u][v] =c;
    }
    long long mf=0;
    while(1){
        f=0;
        for(int i = 0; i < N; i++)
            dist[i]=INF,p[i]==i;
        dist[S]=0;
        queue<int> q;
        q.push(S);
        while(!q.empty()){
            int u = q.front(); q.pop();
            if(u==T)
                break;
            for(int i = 0; i < N; i++){
                if(res[u][i] > 0 && dist[i]==INF){
                    dist[i]=dist[u]+1;
                    p[i]=u;
                    q.push(i);
                }
            }
        }
        aug(T,INF);
        if(f==0)
            break;
        mf+=f;
    }
    vector<ii> used;
    for(int i = 0; i < N; i++)
        for(int j = 0; j < N; j++)
            if(graph[i][j] > 0 && res[i][j] < graph[i][j])
                used.push_back(make_pair(i,j));
    printf("%lld %lld %d\n",N,mf,used.size());
    for(int i = 0; i < used.size(); i++){
        int x = used[i].first; int y = used[i].second;
        printf("%d %d %lld\n",x,y,graph[x][y]-res[x][y]);
    }
    return 0;
}
```

7 MCBM

```
#include<bits/stdc++.h>
using namespace std;
typedef complex<double> cc;

const int MX=101;
const int MX_N = 300, INF=10000000;
int S,T,N,n,m,s,v;
vector<int> adjList[MX_N];
int res[MX_N][MX_N];
bool vis[MX_N];

int ff(int u, int minE){
    if(u==T)
        return minE;
    vis[u]=true;
    for(auto i = adjList[u].begin(); i != adjList[u].end(); i++){
        if(!vis[*i] && res[u][*i] > 0){
            if(int f = ff(*i, min(minE,res[u][*i]))){
                res[u][*i] -= f;
                res[*i][u] += f;
                return f;
            }
        }
    }
    return 0;
}

int main(){
    scanf("%d %d %d %d",&n,&m,&s,&v);
    memset(res,0,sizeof(res));
    S=0;
    N=n+m+2;
    T = N-1;
    int mf = 0;
    while(1){
        memset(vis,0,sizeof(vis));
        int f = ff(S,INF);
        if(f==0)
            break;
        mf+=f;
    }
    printf("%d\n",mf);
    return 0;
}
```

8 MST

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

using namespace std;
typedef pair<int,int> ii;

const int MX_N = 20002;
const int MX_M = 30003;

int p[MX_N],M,N;

int find(int i){ return p[i] = (i==p[i] ? i : find(p[i]));}
void join(int a, int b){
    int pa = find(a);
    int pb = find(b);
```

```
    if(pa!=pb)
        p[pa]=pb;
}
bool connected(int a, int b){ return find(a)==find(b);}

struct edge {
    int x,y,w;
    edge(int _x, int _y, int _w) : x(_x), y(_y), w(_w) {}
    bool operator < (edge e) const {
        return w < e.w;
    }
};

int main(){
    while(scanf("%d %d",&N,&M),N||M){
        for(int i = 0; i < N; i++)
            p[i]=i;
        vector<edge> eList;
        vector<ii> treeList;
        int u,v,w;
        for(int i = 0; i < M; i++){
            scanf("%d %d %d",&u,&v,&w);
            eList.push_back(edge(u,v,w));
        }
        sort(eList.begin(),eList.end());
        int cost = 0;
        int sz=N;
        for(auto i = eList.begin(); i != eList.end(); i++){
            v=(*i).x; u=(*i).y; w=(*i).w;
            if(!connected(u,v)){
                join(u,v);
                treeList.push_back(make_pair(min(u,v),max(u,v)));
                sz--;
                cost+=w;
            }
        }
        if(sz!=1)
            puts("Impossible");
        else {
            printf("%d\n",cost);
            sort(treeList.begin(), treeList.end());
            for(int i = 0; i < treeList.size(); i++){
                printf("%d %d\n",treeList[i].first,treeList[i].second);
            }
        }
    }
    return 0;
}
```

9 Segment Tree

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

const int MX_N=200,INF=1000*1000*1000;
int tree[MX_N*4];
int a[MX_N];
int N;

void construct(int p, int L, int R){
    if(L==R){
        tree[p] = a[L];
        return;
    }
    if(R<L)
        return;
    int md = (L+R)/2;
    construct(2*p,L,md);
    construct(2*p+1,md+1,R);
    tree[p] = min(tree[2*p],tree[2*p+1]);
}

int minR(int p, int L, int R, int l, int r){
    if(r < L || l > R)
        return INF;
    if(l>=L && r<=R)
        return tree[p];
    int md = (l+r)/2;
    return min(minR(2*p,L,R,l,md),minR(2*p+1,L,R,md+1,r));
}

int main(){
    scanf("%d",&N);
    for(int i = 0; i < N;++i)
        scanf("%d",&a[i]);
    construct(1,0,N-1);
    int Q,l,r;
    scanf("%d",&Q);
    for(int i = 0; i < Q; ++i){
        scanf("%d %d",&l,&r);
        printf("%d\n",minR(1,l,r,0,N-1));
    }
    return 0;
}
```

10 RectInHist

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

const int MX_RC=1000;
int R,C;
char board[MX_RC][MX_RC];
int h[MX_RC][MX_RC];

int perim(int l, int w){
    if(l==0 || w==0)
        return 0;
    return 2*l + 2*w;
}
```

```
int main(){
    scanf("%d %d",&R,&C);
    for(int i = 0; i < R; i++)
        for(int j = 0; j < C; j++)
            scanf(" %c",&board[i][j]);
    for(int i = 0; i < R; i++){
        int run=0;
        for(int j = 0; j < C; j++){
            run = (board[i][j] == '.' ? run+1:0);
            h[i][j] = run;
        }
    }
    int mx = 0;
    for(int j = 0; j < C; j++){
        stack<int> s;
        for(int i = 0; i < R; i++){
            if(s.empty() || h[i][j] > h[s.top()][j])
                s.push(i);
            else if(h[i][j] < h[s.top()][j]){
                while(!s.empty() && h[i][j] < h[s.top()][j]){
                    int l = h[s.top()][j]; s.pop();
                    int pm = perim(l, (s.empty() ? i : i-s.top()-1));
                    mx = max(mx,pm);
                }
                s.push(i);
            } else if(h[i][j] == h[s.top()][j]) {
                s.pop();
                s.push(i);
            }
        }
        while(!s.empty()){
            int l = h[s.top()][j]; s.pop();
            int pm = perim(l, s.empty() ? R : R - s.top()-1);
            mx = max(mx,pm);
        }
    }
    printf("%d\n",mx-1);
    return 0;
}
```

11 SCC Tarjans

```
typedef pair<int, int> ii;

const int MX_N = 20002;
int N,M;
vector<int> adjList[MX_N];
int dfs_num[MX_N],dfs_low[MX_N];
bool vis[MX_N];
stack<int> scc;
int dfsCounter=1;
int sccIdx=1;

map<int, int> sccMap;

void tarjans(int u){
    scc.push(u);
    vis[u]=true;

    dfs_low[u]=dfs_num[u]=dfsCounter++;

    for(int i = 0; i < adjList[u].size(); i++){
        int v = adjList[u][i];
        if(dfs_num[v]==0){
            tarjans(v);
            dfs_low[u]=min(dfs_low[u],dfs_low[v]);
        } else if(vis[v]){
            dfs_low[u]=min(dfs_low[u],dfs_num[v]);
        }
    }
    if(dfs_low[u]==dfs_num[u]){
        while(1){
            int v = scc.top(); scc.pop();
            sccMap[v]=sccIdx;
            vis[v]=false;
            if(v==u)
                break;
        }
        sccIdx++;
    }
}
```

12 Sparse Table

```
inline int rmq(int u, int v){
    if(u > v)
        return -2000000000;
    int k = (int) floor(log2((double) (v-u+1)));
    if(r[mtable[u][k]] > r[mtable[v-(1<<k) + 1][k]])
        return r[mtable[u][k]];
    return r[mtable[v-(1<<k) + 1][k]];
}

for(int i = 0; i < N; i++)
    mtable[i][0] = i;
for(int j = 1; (1 << j) <= N; j++)
    for(int i = 0; i + (1<<j)-1 < N; ++i)
        if(r[mtable[i][j-1]] > r[mtable[i + (1 << (j-1))][j-1]])
            mtable[i][j] = mtable[i][j-1];
        else
            mtable[i][j] = mtable[i+(1<<(j-1))][j-1];
```

13 Suffix Array

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

```
const int MX_N = 200020;
char * buff;
int RA[MX_N],SA[MX_N],tempRA[MX_N],tempSA[MX_N],N,c[MX_N];

void countingSort(int k){
    int i,sum,maxi=max(300,N);
    memset(c,0,sizeof(c));
    for(i = 0; i < N; i++)
        c[i+k < N ? RA[i+k] : 0]++;
    for(i=sum=0; i < maxi; i++){
        int t = c[i];
        c[i]=sum;
        sum+=t;
    }
    for(i = 0; i < N; i++)
        tempSA[c[SA[i]+k < N ? RA[SA[i]+k]: 0]++] = SA[i];
    for(i=0; i < N;i++)
        SA[i]=tempSA[i];
}

int main(){
    buff = new char[100011];
    char * doubled = new char[MX_N];
    cin.getline(buff,MX_N);
    N=strlen(buff);
    for(int i = 0; i < N; i++)
        SA[i]=i,RA[i]=buff[i];
    int r;
    for(int k = 1; k < N; k <<= 1){
        countingSort(k);
        countingSort(0);
        tempRA[SA[0]]=r=0;
        for(int i = 1; i < N; i++){
            tempRA[SA[i]] = (RA[SA[i]]==RA[SA[i-1]] && RA[SA[i]+k]==RA[SA[i-1]+k] ? r:++r);
        }
        for(int i = 0; i < N; i++)
            RA[i]=tempRA[i];
    }
    delete buff;
    return 0;
}
```

14 Trie

```
#include<bits/stdc++.h>
using namespace std;
struct node {
    node * children[26];
    int count;
    node(){
        memset(children,0,sizeof(children));
        count=0;
    }
};

void insert(node* nd, char *s){
    if(*s){
        if(!nd->children[*s-'a'])
            nd->children[*s-'a']=new node();
        insert(nd->children[*s-'a'],s+1);
    }
    nd->count++;
}

int count(node* nd, char *s){
    if(*s){
        if(!nd->children[*s-'a'])
            return 0;
        return count(nd->children[*s-'a'],s+1);
    } else {
        return nd->count;
    }
}

int main(){
    node * trie = new node();
    int N; scanf("%d",&N);
    char * buff = new char[40];
    for(int i = 0; i < N; i++){
        scanf("%s",buff);
        printf("%d\n",count(trie,buff));
        insert(trie,buff);
    }
    return 0;
}
```

15 UFDS

```
int find(int u){ return p[u] = (p[u] == u ? u : find(p[u])); }

inline void join(int a, int b){
    pa = find(a);
    pb = find(b);
    if(pa!=pb){
        if(rank[pa] < rank[pb]){
            ni = pb;
            pb = pa;
            pa = ni;
        }
        p[pb] = pa;
        if(rank[pa]==rank[pb])
            rank[pa]++;
    }
}
```

16 vimrc

```
set nocompatible
set backspace=indent,eol,start
```

```
set backup
set undofile

set history=50
set ruler
set showcmd
set incsearch
set laststatus=2
set number
set relativenumber
set cursorline
set grepprg=grep\ -nH\ $*
let g:tex_flavor='latex'
imap jj <ESC>
nnoremap <CR> :noh<CR><CR>
colo slate
filetype indent on
set wildmenu

" Tabs"
set tabstop=8
set softtabstop=0
set expandtab
set shiftwidth=4
set smarttab

set autoindent " always set autoindenting on
```