Longest Increasing Sequence (LIS)

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

bool cmp(pair<int, int> a, pair<int, int> b){
    return a.fi < b.fi;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
        freopen("TASK.INP", "r", stdin);
        freopen("TASK.OUT", "w", stdout);
    }

    int n;
    cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++){
        cin >> v[i];
    }

    vector<pair<int, int>> f;
    vector<int> m(n, -1);

    for (int i = n - 1; i >= 0; --i) {
        int ve = -v[i];
        auto it = lower_bound(f.begin(), f.end(), make_pair(ve, 0LL), cmp);
        int j = distance(f.begin(), it);
        if (j == f.size()) {
            f.emplace_back(ve, i);
        } else {
            f[j] = {ve, i};
        }
        if (j > 0) m[i] = f[j - 1].se;
    }

    vector<int> ans;
    int cur = f.back().se;
    while (cur != -1) {
        ans.push_back(v[cur]);
        cur = m[cur];
    }
    cout << ans.size() << '\n';
```

```
    for (auto it : ans){
        cout << it << " ";
    }
    cout << '\n';
}
```

Knapsack

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;



signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n;
    int M;
    cin >> n >> M;
    vector<int> w(n), v(n);
    for (int i = 0; i < n; i++){
        cin >> w[i] >> v[i];
    }
    vector<vector<int>> f(n + 1, vector<int>(M + 1, 0));

    for (int i = 1; i <= n; i++){
        for (int j = 0; j <= M; j++){
            f[i][j] = f[i - 1][j];
            if (j >= w[i - 1]){
                f[i][j] = max(f[i][j], f[i - 1][j - w[i - 1]] + v[i - 1]);
            }
        }
    }
    cout << f[n][M];

}
```

Sum of Four Values

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

```
#define fi first
#define se second

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")) {
        freopen("TASK.INP", "r", stdin);
        freopen("TASK.OUT", "w", stdout);
    }

    int n, X;
    cin >> n >> X;
    vector<int> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i];
    }

    map<int, pair<int, int>> m;

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            int need = X - v[i] - v[j];
            if (m.count(need)) {
                auto [x, y] = m[need];
                if (x != i && x != j && y != i && y != j) {
                    cout << x + 1 << " " << y + 1 << " " << i + 1 << " " << j + 1
                    return 0;
                }
            }
        }
        for (int j = 0; j < i; j++) {
            m[v[i] + v[j]] = {j, i};
        }
    }

    cout << "IMPOSSIBLE\n";
}
```

Meet in the midle

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> v(N), a, b;
int n, X;
void mitm1(int i, int sum){
    if (sum > X) return;
    if (i > n/2){
```

```
        a.push_back(sum);
        return;
    }
    mitm1(i+1, sum);
    mitm1(i+1, sum + v[i]);
}

void mitm2(int i, int sum){
    if (sum > X) return;
    if (i > n){
        b.push_back(sum);
        return;
    }
    mitm2(i+1, sum);
    mitm2(i+1, sum + v[i]);
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    cin >> n >> X;
    for (int i = 1; i <= n; i++){
        cin >> v[i];
    }

    mitm1(1, 0);
    mitm2(n/2+1, 0);
    sort(b.begin(), b.end());
    int ans = 0;
    for (int i = 0; i < a.size(); i++){
        ans += upper_bound(b.begin(), b.end(), X - a[i]) - lower_bound(b.begin(),
    }
    cout << ans;
}
```

khangtd.ConnectedComponents

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> g[N];
vector<int> visited(N, 0);
vector<int> ans;
void dfs(int u){
    visited[u] = 1;
```

```
    ans.push_back(u);
    for (int v : g[u]){
        if (!visited[v]){
            dfs(v);
        }
    }
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m;
    cin >> n >> m;
    for (int i=0; i<m; i++){
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    int q;
    cin >> q;
    dfs(q);
    cout << ans.size() << "\n";
    sort(ans.begin(), ans.end());
    for (int i=0; i<ans.size(); i++){
        cout << ans[i] << " ";
    }
}
```

Đồ thị vô hướng - Đếm số đỉnh cô lập

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> g[N];
vector<int> visited(N, 0);
int c = 0;
void dfs(int u){
    visited[u] = 1;
    c++;
    for (int v : g[u]){
        if (!visited[v]){
            dfs(v);
        }
    }
```

```
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m;
    cin >> n >> m;
    for (int i=0; i<m; i++){
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    int ans = 0;
    for (int i=0; i<n; i++){
        if (!visited[i]){
            dfs(i);
            if (c == 1){
                ans++;
            }
        }
        c = 0;
    }
    cout << ans;
}
```

Đồ thị vô hướng - Đếm số thành phần liên thông

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> g[N];
vector<int> visited(N, 0);
int c = 0;
void dfs(int u){
    visited[u] = 1;
    c++;
    for (int v : g[u]){
        if (!visited[v]){
            dfs(v);
        }
    }
}
```

```
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m;
    cin >> n >> m;
    for (int i=0; i<m; i++){
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    int ans = 0;
    for (int i=0; i<n; i++){
        if (!visited[i]){
            dfs(i);
            ans++;
        }
    }
    cout << ans;

}
```

Đồ thị vô hướng - Liệt kê các đỉnh có thể tới từ đỉnh S.

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> g[N];
vector<int> visited(N, 0);
int c = 0;
void dfs(int u){
    visited[u] = 1;
    c++;
    for (int v : g[u]){
        if (!visited[v]){
            dfs(v);
        }
    }
}


signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
```

```
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m;
    cin >> n >> m;
    for (int i=0; i<m; i++){
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    dfs(0);
    vector<int> ans;
    for (int i=0; i<n; i++){
        if (visited[i] && i != 0){
            ans.push_back(i);
        }
    }
    sort(ans.begin(), ans.end());
    if (ans.size() == 0){
        cout << "KHONG";
        return 0;
    }
    for (auto it : ans){
        cout << it << " ";
    }

}
```

Đồ thị vô hướng - Kiểm tra có đường đi từ đỉnh S tới đỉnh E

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> g[N];
vector<int> visited(N, 0);
int c = 0;
void dfs(int u){
    visited[u] = 1;
    c++;
    for (int v : g[u]){
        if (!visited[v]){
            dfs(v);
        }
    }
}
```

```
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m;
    cin >> n >> m;
    for (int i=0; i<m; i++){
        int x, y;
        cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    dfs(0);
    vector<int> ans;
    for (int i=1; i<n; i++){
        if (visited[i]){
            cout << "CO";
        }
        else{
            cout << "KHONG";
        }
        cout << "\n";
    }

}
```