**Bài 1:**

Main Idea:  
Viết hàm đọc dữ liệu input thông qua ifstream đọc từng dòng, hàm kiểm tra bậc chẵn và hàm xử lý đồ thị euler

Code:

#include <iostream>

#include <stack>

#include <vector>

#include <fstream>

using *namespace* std;

#define FI "test.inp"

#define FO "test.out"

vector<vector<*int*>> adj;

vector<*int*> degree;

vector<*int*> euler;

*int* n;

*void* readInput(ifstream &*fi*) {

*fi* >> n;

*fi*.ignore();

    adj.assign(n + 1, vector<*int*>(n + 1, 0));

    degree.assign(n + 1, 0); //khá tương tự với resize

    for (*int* i = 1; i <= n; ++i)

        for (*int* j = 1; j <= n; ++j) {

*fi* >> adj[i][j];

            degree[i] += adj[i][j];

        }

}

*bool* isEuler() { //kiểm tra bậc chẵn

    for (*int* i = 1; i <= n; ++i)

        if (degree[i] % 2 != 0)

            return false;

    return true;

}

*void* findEulerCycle(*int* *start*) { //hàm euler

    stack<*int*> st;

    st.push(*start*);

    while (!st.empty()) {

*int* u = st.top();

*bool* found = false;

        for (*int* v = 1; v <= n; ++v) {

            if (adj[u][v]) {

                st.push(v);

                adj[u][v]--;

                adj[v][u]--;

                found = true;

                break;

            }

        }

        if (!found) {

            euler.push\_back(u);

            st.pop();

        }

    }

}

*int* main() {

    ifstream fi(FI);

    ofstream fo(FO);

    if (!fi || !fo) {

        cerr << "Không thể mở file!" << endl;

        return 1;

    }

    readInput(fi);

    if (!isEuler()) {

        fo << 0 << endl;

    } else {

        fo << 1 << endl;

        findEulerCycle(1); // bắt đầu từ đỉnh 1

        for (*int* i = euler.size() - 1; i >= 0; --i)

            fo << euler[i] << " ";

        fo << endl;

    }

    fi.close();

    fo.close();

    return 0;

}

Input:

5

0 1 1 0 0

1 0 1 1 1

1 1 0 1 1

0 1 1 0 2

0 1 1 2 0

Output:

1

1 2 3 4 2 5 4 5 3 1

**Bài 2:**

Main idea: Chỉnh sửa lại hàm nhập vào theo domino sang ma trận kề và tái sử dụng lại 2 hàm xử lý từ bài trên, chỉnh lại dữ liệu output để in ra được domino khi xếp thành vòng tròn

Code:  
#include<iostream>

#include<algorithm>

#include<stack>

#include<fstream>

#include<vector>

using *namespace* std;

#define FI "test1.inp"

#define FO "test1.out"

*int* n;

vector<vector<*int*>> adj(7, vector<*int*>(7,0));

vector<pair<*int*,*int*>>result;

*void* readInput(ifstream &*fi*)

{

*fi* >> n;

    for (*int* i=0; i<n;i++)

    {

*int* x, y;

*fi*>>x>>y;

        adj[x][y]++;

        adj[y][x]++;

    }

}

*bool* isEuler()

{

    for (*int* i=0;i<=6;i++)

    {

*int* deg=0;

        for(*int* j=0;j<=6;j++)

        {

            deg+= adj[i][j];

        }

        if (deg %2!=0)

        {

            return false;

        }

    }

    return true;

}

*void* findEuler(*int* *start*)

{

    stack <*int*> st;

    st.push(*start*);

    while (!st.empty())

    {

*int* u=st.top();

*bool* found = false;

        for (*int* v=0;v<=6;v++)

        {

            if (adj [u][v])

            {

                st.push(v);

                adj[u][v]--;

                adj[v][u]--;

                found = true;

                break;

            }

        }

            if (!found)

            {

                st.pop();

                if(!st.empty())

                {

                    result.push\_back({st.top(),u});

                }

            }

    }

}

*int* main()

{

    ifstream fi(FI);

    ofstream fo(FO);

    readInput(fi);

*int* start = -1;

    for (*int* i = 0; i <= 6; ++i) {

        for (*int* j = 0; j <= 6; ++j) {

            if (adj[i][j]) {

                start = i;

                break;

            }

        }

        if (start != -1) break;

    }

    findEuler(start);

    for (*int* i = 0; i <= 6; ++i) {

        for (*int* j = 0; j <= 6; ++j) {

            if (adj[i][j]) {

                start = i;

                break;

            }

        }

        if (start != -1) break;

    }

     if (result.size() != n) {

        fo << 0 << endl;

    } else {

        fo << 1 << endl;

        for (*auto* &p : result)

            fo << p.first << " " << p.second << endl;

    }

    fi.close();

    fo.close();

    return 0;

}

Input:   
10

1 2

1 3

1 4

1 5

2 3

2 4

2 5

3 4

3 5

4 5

Output:

1

5 1

4 5

3 4

5 3

2 5

4 2

1 4

3 1

2 3

1 2

**Bài 3:**

Main Idea: Tái sử dụng lại hàm đọc dữ liệu và hàm euler, nhưng tại hàm main, thì sẽ có vòng for đếm bậc lẻ, nếu chỉ có 0 hoặc 2 đỉnh bậc lẻ, thì sẽ chạy hàm euler và in kết quả, còn nếu không có thì trả về 0

Code:

#include <iostream>

#include <stack>

#include <vector>

#include <fstream>

using *namespace* std;

#define FI "test2.inp"

#define FO "test2.out"

vector<vector<*int*>> adj;

vector<*int*> degree;

vector<*int*> euler;

*int* n;

*void* readInput(ifstream &*fi*) {

*fi* >> n;

    adj.assign(n + 1, vector<*int*>(n + 1, 0));

    degree.assign(n + 1, 0);

    for (*int* i = 1; i <= n; ++i)

        for (*int* j = 1; j <= n; ++j) {

*fi* >> adj[i][j];

            degree[i] += adj[i][j];

            degree[j] += adj[i][j];

        }

}

//bool isEuler() {

   // for (int i = 1; i <= n; ++i)

   //     if (degree[i] % 2 != 0)

   //         return false;

   // return true;

//}

*void* findEulerCycle(*int* *start*) {

    stack<*int*> st;

    st.push(*start*);

    while (!st.empty()) {

*int* u = st.top();

*bool* found = false;

        for (*int* v = 1; v <= n; ++v) {

            if (adj[u][v]) {

                st.push(v);

                adj[u][v]--;

                adj[v][u]--;

                found = true;

                break;

            }

        }

        if (!found) {

            euler.push\_back(u);

            st.pop();

        }

    }

}

*int* main() {

    ifstream fi(FI);

    ofstream fo(FO);

    if (!fi || !fo) {

        cerr << "Không thể mở file!" << endl;

        return 1;

    }

    readInput(fi);

*int* countOdd = 0;

*int* startBD = 1;

for (*int* i = 1; i <= n; ++i) {

    if (degree[i] % 2 != 0) {

        countOdd++;

        startBD = i;

    }

}

if (countOdd == 0 || countOdd == 2) {

    fo << 1 << endl;

    findEulerCycle(startBD);

    for (*int* i = euler.size() - 1; i >= 0; --i)

        fo << euler[i] << " ";

    fo << endl;

} else {

    fo << 0 << endl;

}

    //if (!isEuler()) {

    //    fo << 0 << endl;

    //} else {

    //    fo << 1 << endl;

    //    findEulerCycle(1); // bắt đầu từ đỉnh 1

    //    for (int i = euler.size() - 1; i >= 0; --i)

    //        fo << euler[i] << " ";

    //    fo << endl;

    //}

    fi.close();

    fo.close();

    return 0;

}

Input: (Đồ thị có 2 đỉnh bậc lẻ)

5

0 1 1 1 0

1 0 1 1 0

1 1 0 1 1

1 1 1 0 1

0 0 1 1 0

Output:   
1

1 2 3 1 4 3 5 4 2

**Bài 4:**

Main Idea: Sử dụng thuật toán backtrack để tìm được đồ thị Hamilton

Code:

#include <iostream>

#include <stack>

#include <vector>

#include <fstream>

using *namespace* std;

#define FI "test3.inp"

#define FO "test3.out"

vector<vector<*int*>>adj;

vector<*bool*> visited;

vector<*int*>hamilton;

*int* n;

*bool* backtrack(*int* *pos*) {

    if (hamilton.size() == n) {

        if (adj[hamilton.back()][hamilton[0]] == 1) {

            hamilton.push\_back(hamilton[0]);

            return true;

        }

        return false;

    }

    for (*int* v = 0; v < n; ++v) {

        if (adj[*pos*][v] == 1 && !visited[v]) {

            visited[v] = true;

            hamilton.push\_back(v);

            if (backtrack(v)) return true;

            visited[v] = false;

            hamilton.pop\_back();

        }

    }

    return false;

}

*int* main()

{

    ifstream fi(FI);

    ofstream fo(FO);

    fi >> n;

    adj.assign(n, vector<*int*>(n));

    visited.assign(n, false);

    for (*int* i = 0; i < n; ++i)

        for (*int* j = 0; j < n; ++j)

            fi >> adj[i][j];

    visited[0] = true;

    hamilton.push\_back(0);

    if (backtrack(0)) {

        fo << 1 << "\n";

        for (*int* v : hamilton)

            fo << v + 1 << " ";

        fo << "\n";

    } else {

        fo << 0 << "\n";

    }

    fi.close();

    fo.close();

    return 0;

}

Input:

5

0 1 1 1 0

1 0 1 1 1

1 1 0 1 1

1 1 1 0 1

0 0 1 1 0

Output:

1

1 2 3 5 4 1

**Bài 5:**

Main idea: Tái sử dụng lại bài 1 nhưng sử dụng hàm kiểm tra đồ thị euler không đệ quy

Code:

#include <iostream>

#include <stack>

#include <vector>

#include <fstream>

using *namespace* std;

#define FI "test4.inp"

#define FO "test4.out"

vector<vector<*int*>> adj;

vector<*int*> degree;

vector<*int*> euler;

*int* n;

*void* readInput(ifstream &*fi*) {

*fi* >> n;

    adj.assign(n + 1, vector<*int*>(n + 1, 0));

    degree.assign(n + 1, 0);

    for (*int* i = 1; i <= n; ++i)

        for (*int* j = 1; j <= n; ++j) {

*fi* >> adj[i][j];

            degree[i] += adj[i][j];

        }

}

*bool* isEuler() {

    for (*int* i = 1; i <= n; ++i)

        if (degree[i] % 2 != 0)

            return false;

    return true;

}

*void* findEulerCycle(*int* *start*) {

    stack<*int*> st;

    st.push(*start*);

    while (!st.empty()) {

*int* u = st.top();

*bool* found = false;

        for (*int* v = 1; v <= n; ++v) {

            if (adj[u][v]) {

                st.push(v);

                adj[u][v]--;

                adj[v][u]--;

                found = true;

                break;

            }

        }

        if (!found) {

            euler.push\_back(u);

            st.pop();

        }

    }

}

*int* main() {

    ifstream fi(FI);

    ofstream fo(FO);

    if (!fi || !fo) {

        cerr << "Không thể mở file!" << endl;

        return 1;

    }

    readInput(fi);

    if (!isEuler()) {

        fo << 0 << endl;

    } else {

        fo << 1 << endl;

*int* start = 1;

        for (*int* i = 1; i <= n; ++i)

        if (degree[i] % 2 != 0) {

          start = i;

         break;

    }

        findEulerCycle(start); // bắt đầu từ đỉnh 1

        for (*int* i = euler.size() - 1; i >= 0; --i)

            fo << euler[i] << " ";

        fo << endl;

    }

    fi.close();

    fo.close();

    return 0;

}

Input:

5

0 1 1 0 0

1 0 1 1 1

1 1 0 1 1

0 1 1 0 2

0 1 1 2 0

Output:

1

1 2 3 4 2 5 4 5 3 1

**Bài 6:**

Main Idea: Tương tự như bài trên nhưng phải biến tấu lại kiểu dữ liệu nhập là danh sách kề và thay đổi cách xoá cạnh của những đỉnh đã được thêm vào mảng lưu kết quả ở của hàm xử lý euler của bài trên

Code:

#include <iostream>

#include <stack>

#include <vector>

#include <fstream>

#include <algorithm>

#include<sstream>

using *namespace* std;

#define FI "test5.inp"

#define FO "test5.out"

vector<vector<*int*>> adj;

vector<*int*> degree;

vector<*int*> euler;

*int* n;

*void* readInput(ifstream &*fi*) {

*fi* >> n;

    adj.assign(n + 1, vector<*int*>());

    degree.assign(n + 1, 0);

    string line;

    getline(*fi*, line); // bỏ dòng số lượng đỉnh

    for (*int* i = 1; i <= n; ++i) {

        getline(*fi*, line);

        istringstream iss(line);

*int* u;

        iss >> u;

*int* v;

        while (iss >> v) {

    adj[u].push\_back(v);

    adj[v].push\_back(u);

    degree[u]++;

    degree[v]++;

        }

    }

}

// void readInput(ifstream &fi) {

//     fi >> n;

//     adj.assign(n + 1, vector<int>());

//     degree.assign(n + 1, 0);

//     for (int i = 1; i <= n; ++i)

//         for (int j = 1; j <= n; ++j) {

//             int x;

//             fi >> x;

//             for (int k = 0; k < x; ++k) {

//                 adj[i].push\_back(j); //biến đổi sang danh sách kề

//                 degree[i]++;

//             }

//         }

// }

*bool* isEuler() {

*int* oddCount = 0;

    for (*int* i = 1; i <= n; ++i)

    if (degree[i] % 2 != 0)

        oddCount++;

        return (oddCount == 0 || oddCount == 2);

}

*void* removeEdge(*int* *u*, *int* *v*) {

*auto* it\_u = find(adj[*u*].begin(), adj[*u*].end(), *v*);

    if (it\_u != adj[*u*].end()) adj[*u*].erase(it\_u);

*auto* it\_v = find(adj[*v*].begin(), adj[*v*].end(), *u*);

    if (it\_v != adj[*v*].end()) adj[*v*].erase(it\_v);

}

*void* findEulerCycle(*int* *start*) {

    stack<*int*> st;

    st.push(*start*);

    while (!st.empty()) {

*int* u = st.top();

        if (!adj[u].empty()) {

*int* v = adj[u].back();

            st.push(v);

            removeEdge(u, v);

        } else {

            euler.push\_back(u);

            st.pop();

        }

    }

}

*int* main() {

    ifstream fi(FI);

    ofstream fo(FO);

    if (!fi || !fo) {

        cerr << "Không thể mở file!" << endl;

        return 1;

    }

    readInput(fi);

    if (!isEuler()) {

        fo << 0 << endl;

    } else {

        fo << 1 << endl;

*int* start = 1;

    for (*int* i = 1; i <= n; ++i) {

    if (degree[i] % 2 != 0) {

        start = i;

        break;

    }

}

    findEulerCycle(start);

        for (*int* i = euler.size() - 1; i >= 0; --i)

            fo << euler[i] << " ";

        fo << endl;

    }

    fi.close();

    fo.close();

    return 0;

}

Input: (Y hệt bài 1 và bài 5 nhưng là kiểu dữ liệu của danh sách kề)

5

1 2 3

2 1 3 4 5

3 1 2 4 5

4 2 3 5 5

5 2 3 4 4

Output: (Có in những đỉnh tự nối với chính nó)

1

1 3 5 4 5 4 5 3 4 3 2 5 2 4 2 3 1 2 1