Lý thuyết đồ thị và bài tập DIMACS Implementation Challenges

Course Project

Ngày 23 tháng 7 năm 2025

Mục lục

1 Giới thiệu về lý thuyết đồ thị

1.1 Khái niệm cơ bản

Đồ thị là một cấu trúc dữ liệu quan trọng trong toán học và khoa học máy tính, được sử dụng để mô hình hóa mối quan hệ giữa các đối tượng.

Định nghĩa: Một đồ thị G được định nghĩa là một cặp có thứ tự $G = (V, E)$ trong đó:

- V là tập hợp các đỉnh (vertices)
- E là tập hợp các cạnh (edges), mỗi cạnh nối hai đỉnh

1.2 Phân loại đồ thị

- **Đồ thị có hướng (Directed Graph):** Các cạnh có hướng từ đỉnh này đến đỉnh khác
- **Đồ thị vô hướng (Undirected Graph):** Các cạnh không có hướng
- **Đồ thị có trọng số (Weighted Graph):** Mỗi cạnh có một giá trị trọng số
- **Đồ thị đầy đủ (Complete Graph):** Mọi cặp đỉnh đều được nối với nhau
- **Đồ thị hai phần (Bipartite Graph):** Tập đỉnh có thể chia thành hai tập con không giao nhau

1.3 Các khái niệm quan trọng

- **Bậc của đỉnh:** Số cạnh kề với đỉnh đó
- **Đường đi (Path):** Chuỗi các đỉnh liên tiếp được nối bởi các cạnh
- **Chu trình (Cycle):** Đường đi khép kín
- **Đồ thị liên thông:** Tồn tại đường đi giữa mọi cặp đỉnh
- **Ma trận kề (Adjacency Matrix):** Biểu diễn đồ thị bằng ma trận 2D

2 Định dạng DIMACS

2.1 Giới thiệu về DIMACS

DIMACS (Discrete Mathematics and Theoretical Computer Science) là một định dạng chuẩn để biểu diễn các bài toán đồ thị trong các cuộc thi lập trình và nghiên cứu.

2.2 Cấu trúc định dạng DIMACS

Định dạng DIMACS cho đồ thị có cấu trúc như sau:

```
c Dòng comment bắt đầu bằng 'c'
p edge n m
e u v
e u v
...
```

Trong đó:

- `c`: Dòng comment
- `p edge n m`: Dòng định nghĩa có n đỉnh và m cạnh
- `e u v`: Dòng mô tả cạnh nối đỉnh u và v

3 Bài tập 1.1: Biểu diễn đồ thị vô hướng theo định dạng DIMACS

3.1 Mô tả bài toán

Biểu diễn chuẩn của đồ thị vô hướng theo định dạng DIMACS bao gồm:

- Dòng định nghĩa vấn đề với số đỉnh n và số cạnh m
- m dòng mô tả cạnh
- Các dòng comment bắt đầu bằng "c"

3.2 Ý tưởng và giải pháp

1. Đọc đồ thị từ input hoặc tạo đồ thị mẫu
2. Xuất ra định dạng DIMACS theo cấu trúc chuẩn
3. Đảm bảo các đỉnh được đánh số từ 1 đến n

3.3 Thuật toán

Algorithm 1 Xuất đồ thị theo định dạng DIMACS

Require: Đồ thị $G = (V, E)$ với $|V| = n$, $|E| = m$
Ensure: Đồ thị được xuất theo định dạng DIMACS
1: In dòng comment mô tả đồ thị
2: In dòng "p edge n m"
3: **for** mỗi cạnh $(u, v) \in E$ **do**
4: In "e u v"
5: **end for**

3.4 Code C++

Listing 1: Xuất đồ thị DIMACS - C++

```cpp
#include <iostream>
#include <vector>
#include <fstream>
using namespace std;

class Graph {
```

```
7  private:
8      int n; // so dinh
9      vector<pair<int, int>> edges; // danh sach canh
10
11 public:
12     Graph(int vertices) : n(vertices) {}
13
14     void addEdge(int u, int v) {
15         edges.push_back({u, v});
16     }
17
18     void exportDIMACS(const string& filename = "") {
19         ostream* out = &cout;
20         ofstream file;
21
22         if (!filename.empty()) {
23             file.open(filename);
24             out = &file;
25         }
26
27         // Comment lines
28         *out << "c Do thi vo huong voi " << n << " dinh\n";
29         *out << "c Xuat theo dinh dang DIMACS\n";
30
31         // Problem line
32         *out << "p edge " << n << " " << edges.size() << "\n";
33
34         // Edge lines
35         for (const auto& edge : edges) {
36             *out << "e " << edge.first << " " << edge.second << "\n";
37         }
38
39         if (file.is_open()) {
40             file.close();
41             cout << "Da xuat do thi ra file: " << filename << endl;
42         }
43     }
44
45     void readFromDIMACS(const string& filename) {
46         ifstream file(filename);
47         string line;
48
49         while (getline(file, line)) {
50             if (line[0] == 'c') continue; // bo qua comment
51
52             if (line[0] == 'p') {
53                 // Doc dong problem
54                 string type;
55                 int vertices, edgeCount;
56                 sscanf(line.c_str(), "p %s %d %d", &type[0],
                       &vertices, &edgeCount);
57                 n = vertices;
58                 edges.clear();
```

```
            }

            if (line[0] == 'e') {
                // Doc canh
                int u, v;
                sscanf(line.c_str(), "e %d %d", &u, &v);
                addEdge(u, v);
            }
        }
        file.close();
    }
};

int main() {
    // Tao do thi mau
    Graph g(5);
    g.addEdge(1, 2);
    g.addEdge(2, 3);
    g.addEdge(3, 4);
    g.addEdge(4, 5);
    g.addEdge(1, 5);

    cout << "=== Xuat do thi theo dinh dang DIMACS ===\n";
    g.exportDIMACS();

    // Xuat ra file
    g.exportDIMACS("graph.dimacs");

    return 0;
}
```

3.5 Code Python

Listing 2: Xuất đồ thị DIMACS - Python

```
class Graph:
    def __init__(self, vertices):
        self.n = vertices
        self.edges = []

    def add_edge(self, u, v):
        """Them canh vao do thi"""
        self.edges.append((u, v))

    def export_dimacs(self, filename=None):
        """Xuat do thi theo dinh dang DIMACS"""
        output = []

        # Comment lines
        output.append(f"c Do thi vo huong voi {self.n} dinh")
        output.append("c Xuat theo dinh dang DIMACS")

        # Problem line
```

```
        output.append(f"p edge {self.n} {len(self.edges)}")

        # Edge lines
        for u, v in self.edges:
            output.append(f"e {u} {v}")

        result = "\n".join(output)

        if filename:
            with open(filename, 'w', encoding='utf-8') as f:
                f.write(result)
            print(f"Da xuat do thi ra file: {filename}")
        else:
            print(result)

    def read_from_dimacs(self, filename):
        """Doc do thi tu file DIMACS"""
        with open(filename, 'r') as f:
            for line in f:
                line = line.strip()
                if line.startswith('c'):
                    continue  # Bo qua comment
                elif line.startswith('p'):
                    parts = line.split()
                    self.n = int(parts[2])
                    self.edges = []
                elif line.startswith('e'):
                    parts = line.split()
                    u, v = int(parts[1]), int(parts[2])
                    self.add_edge(u, v)

# Su dung
if __name__ == "__main__":
    # Tao do thi mau
    g = Graph(5)
    g.add_edge(1, 2)
    g.add_edge(2, 3)
    g.add_edge(3, 4)
    g.add_edge(4, 5)
    g.add_edge(1, 5)

    print("=== Xuat do thi theo dinh dang DIMACS ===")
    g.export_dimacs()

    print("\n=== Xuat ra file ===")
    g.export_dimacs("graph.dimacs")
```

4 Bài tập 1.2: Biểu diễn đồ thị Stanford GraphBase (SGB)

4.1 Mô tả bài toán

Định dạng SGB (Stanford GraphBase) là một định dạng khác để biểu diễn đồ thị, bao gồm:

- Dòng đầu: "* GraphBase"
- Thông tin về đồ thị
- Danh sách đỉnh và cạnh

4.2 Ý tưởng và giải pháp

1. Chuyển đổi từ định dạng DIMACS sang SGB
2. Tạo cấu trúc dữ liệu phù hợp
3. Xuất ra định dạng SGB

4.3 Code C++

Listing 3: Chuyển đổi DIMACS sang SGB - C++

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <map>
using namespace std;

class SGBGraph {
private:
    int n;
    vector<pair<int, int>> edges;
    map<int, string> vertexLabels;

public:
    SGBGraph(int vertices) : n(vertices) {
        // Tao nhan mac dinh cho cac dinh
        for (int i = 1; i <= n; i++) {
            vertexLabels[i] = "v" + to_string(i);
        }
    }

    void addEdge(int u, int v) {
        edges.push_back({u, v});
    }

    void setVertexLabel(int vertex, const string& label) {
        vertexLabels[vertex] = label;
    }

    void exportSGB() {
```

```
30          cout << "* GraphBase\n";
31          cout << "* Vertices: " << n << "\n";
32          cout << "* Edges: " << edges.size() << "\n";
33          cout << "* Graph\n";
34
35          // Xuat thong tin dinh
36          cout << "* Vertices\n";
37          for (int i = 1; i <= n; i++) {
38              cout << vertexLabels[i] << ",0,0,0\n";
39          }
40
41          // Xuat thong tin canh
42          cout << "* Arcs\n";
43          for (const auto& edge : edges) {
44              cout << vertexLabels[edge.first] << ":"
45                   << vertexLabels[edge.second] << ",1\n";
46          }
47
48          cout << "* Checksum ...\n";
49      }
50
51      void readFromDIMACS(const string& filename) {
52          // Tuong tu nhu bai 1.1
53          // Code doc DIMACS...
54      }
55  };
56
57  int main() {
58      SGBGraph g(4);
59      g.addEdge(1, 2);
60      g.addEdge(2, 3);
61      g.addEdge(3, 4);
62      g.addEdge(1, 4);
63
64      // Dat nhan cho cac dinh
65      g.setVertexLabel(1, "A");
66      g.setVertexLabel(2, "B");
67      g.setVertexLabel(3, "C");
68      g.setVertexLabel(4, "D");
69
70      cout << "=== Xuat do thi theo dinh dang SGB ===\n";
71      g.exportSGB();
72
73      return 0;
74  }
```

5 Bài tập 1.3: Tạo đường đi, chu trình và đồ thị bánh xe

5.1 Mô tả bài toán

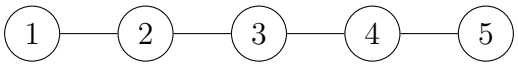
Cần implement các thuật toán để tạo:

- Đường đi P_n với n đỉnh

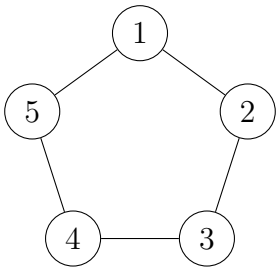
- Chu trình C_n với n đỉnh
- Đồ thị bánh xe W_n với n đỉnh

5.2 Lý thuyết

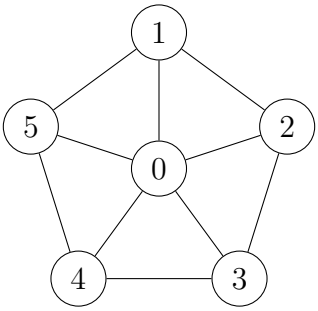
Đường đi P_n: Đồ thị với n đỉnh được sắp xếp thành một dãy tuyến tính, mỗi đỉnh (trừ hai đỉnh đầu cuối) có bậc 2.



Chu trình C_n: Đồ thị vòng tròn với n đỉnh, mỗi đỉnh có bậc 2.



Đồ thị bánh xe W_n: Chu trình C_{n-1} với một đỉnh trung tâm nối với tất cả các đỉnh của chu trình.



5.3 Code C++

Listing 4: Tạo các loại đồ thị đặc biệt - C++

```cpp
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

class SpecialGraphs {
public:
    // Tao duong di P_n
    static vector<pair<int, int>> createPath(int n) {
        vector<pair<int, int>> edges;

        for (int i = 1; i < n; i++) {
            edges.push_back({i, i + 1});
        }

```

```
        return edges;
    }

    // Tao chu trinh C_n
    static vector<pair<int, int>> createCycle(int n) {
        vector<pair<int, int>> edges;

        for (int i = 1; i < n; i++) {
            edges.push_back({i, i + 1});
        }
        edges.push_back({n, 1}); // Noi dinh cuoi voi dinh dau

        return edges;
    }

    // Tao do thi banh xe W_n
    static vector<pair<int, int>> createWheel(int n) {
        vector<pair<int, int>> edges;

        // Tao chu trinh ngoai (tu dinh 1 den n-1)
        for (int i = 1; i < n - 1; i++) {
            edges.push_back({i, i + 1});
        }
        edges.push_back({n - 1, 1}); // Dong chu trinh

        // Noi dinh trung tam (dinh n) voi tat ca dinh cua chu trinh
        for (int i = 1; i < n; i++) {
            edges.push_back({n, i});
        }

        return edges;
    }

    // Xuat do thi theo dinh dang DIMACS
    static void exportDIMACS(const vector<pair<int, int>>& edges,
                             int n, const string& graphType) {
        cout << "c " << graphType << " voi " << n << " dinh\n";
        cout << "p edge " << n << " " << edges.size() << "\n";

        for (const auto& edge : edges) {
            cout << "e " << edge.first << " " << edge.second << "\n";
        }
    }
};

int main() {
    int n = 6;

    cout << "=== DUONG DI P_" << n << " ===\n";
    auto pathEdges = SpecialGraphs::createPath(n);
    SpecialGraphs::exportDIMACS(pathEdges, n, "Duong di");

    cout << "\n=== CHU TRINH C_" << n << " ===\n";
```

```
    auto cycleEdges = SpecialGraphs::createCycle(n);
    SpecialGraphs::exportDIMACS(cycleEdges, n, "Chu trinh");

    cout << "\n=== DO THI BANH XE W_" << n << " ===\n";
    auto wheelEdges = SpecialGraphs::createWheel(n);
    SpecialGraphs::exportDIMACS(wheelEdges, n, "Do thi banh xe");

    return 0;
}
```

5.4 Code Python

Listing 5: Tạo các loại đồ thị đặc biệt - Python

```
import math

class SpecialGraphs:
    @staticmethod
    def create_path(n):
        """Tao duong di P_n"""
        edges = []
        for i in range(1, n):
            edges.append((i, i + 1))
        return edges

    @staticmethod
    def create_cycle(n):
        """Tao chu trinh C_n"""
        edges = []
        for i in range(1, n):
            edges.append((i, i + 1))
        edges.append((n, 1))  # Noi dinh cuoi voi dinh dau
        return edges

    @staticmethod
    def create_wheel(n):
        """Tao do thi banh xe W_n"""
        edges = []

        # Tao chu trinh ngoai (tu dinh 1 den n-1)
        for i in range(1, n - 1):
            edges.append((i, i + 1))
        edges.append((n - 1, 1))  # Dong chu trinh

        # Noi dinh trung tam (dinh n) voi tat ca dinh cua chu trinh
        for i in range(1, n):
            edges.append((n, i))

        return edges

    @staticmethod
    def export_dimacs(edges, n, graph_type):
        """Xuat do thi theo dinh dang DIMACS"""
```

```
        print(f"c {graph_type} voi {n} dinh")
        print(f"p edge {n} {len(edges)}")

        for u, v in edges:
            print(f"e {u} {v}")

    @staticmethod
    def visualize_coordinates(n, graph_type):
        """Tao toa do de ve do thi"""
        coords = {}

        if graph_type == "path":
            for i in range(1, n + 1):
                coords[i] = (i - 1, 0)

        elif graph_type == "cycle":
            for i in range(1, n + 1):
                angle = 2 * math.pi * (i - 1) / n
                coords[i] = (math.cos(angle), math.sin(angle))

        elif graph_type == "wheel":
            # Dinh trung tam
            coords[n] = (0, 0)
            # Cac dinh chu trinh
            for i in range(1, n):
                angle = 2 * math.pi * (i - 1) / (n - 1)
                coords[i] = (1.2 * math.cos(angle), 1.2 *
                   math.sin(angle))

        return coords

if __name__ == "__main__":
    n = 6

    print(f"=== DUONG DI P_{n} ===")
    path_edges = SpecialGraphs.create_path(n)
    SpecialGraphs.export_dimacs(path_edges, n, "Duong di")

    print(f"\n=== CHU TRINH C_{n} ===")
    cycle_edges = SpecialGraphs.create_cycle(n)
    SpecialGraphs.export_dimacs(cycle_edges, n, "Chu trinh")

    print(f"\n=== DO THI BANH XE W_{n} ===")
    wheel_edges = SpecialGraphs.create_wheel(n)
    SpecialGraphs.export_dimacs(wheel_edges, n, "Do thi banh xe")

    # In toa do de ve
    print(f"\n=== TOA DO CAC DINH DE VE ===")
    print("Chu trinh:")
    coords = SpecialGraphs.visualize_coordinates(n, "cycle")
    for vertex, (x, y) in coords.items():
        print(f"Dinh {vertex}: ({x:.2f}, {y:.2f})")
```

6 Bài tập 1.4: Tạo đồ thị đầy đủ và đồ thị hai phần đầy đủ

6.1 Mô tả bài toán

Cần implement thuật toán để tạo:

- Đồ thị đầy đủ K_n với n đỉnh
- Đồ thị hai phần đầy đủ $K_{p,q}$ với $p + q$ đỉnh

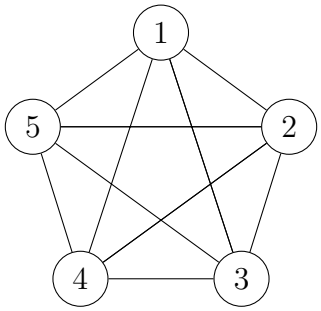
6.2 Lý thuyết

Đồ thị đầy đủ K_n: Đồ thị có n đỉnh và mọi cặp đỉnh đều được nối với nhau. Số cạnh: $\binom{n}{2} = \frac{n(n-1)}{2}$

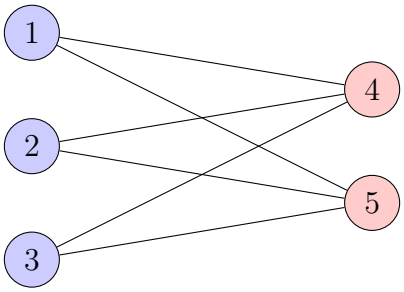
Đồ thị hai phần đầy đủ $K_{p,q}$: Đồ thị có hai tập đỉnh A và B với $|A| = p$, $|B| = q$. Mọi đỉnh trong A đều nối với mọi đỉnh trong B. Số cạnh: $p \times q$

6.3 Hình minh họa

Đồ thị đầy đủ K_5:



Đồ thị hai phần đầy đủ $K_{3,2}$:



6.4 Code C++

Listing 6: Tao do thi day du va do thi hai phan day du - C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

class CompleteGraphs {
public:
```

```
8       // Tao do thi day du K_n
9       static vector<pair<int, int>> createComplete(int n) {
10          vector<pair<int, int>> edges;
11
12          for (int i = 1; i <= n; i++) {
13              for (int j = i + 1; j <= n; j++) {
14                  edges.push_back({i, j});
15              }
16          }
17
18          return edges;
19      }
20
21      // Tao do thi hai phan day du K_{p,q}
22      static vector<pair<int, int>> createCompleteBipartite(int p, int
           q) {
23          vector<pair<int, int>> edges;
24
25          // Tap A: cac dinh tu 1 den p
26          // Tap B: cac dinh tu p+1 den p+q
27          for (int i = 1; i <= p; i++) {
28              for (int j = p + 1; j <= p + q; j++) {
29                  edges.push_back({i, j});
30              }
31          }
32
33          return edges;
34      }
35
36      // Tinh so canh cua do thi day du
37      static int countCompleteEdges(int n) {
38          return n * (n - 1) / 2;
39      }
40
41      // Tinh so canh cua do thi hai phan day du
42      static int countBipartiteEdges(int p, int q) {
43          return p * q;
44      }
45
46      // Xuat do thi theo dinh dang DIMACS
47      static void exportDIMACS(const vector<pair<int, int>>& edges,
48                               int n, const string& graphType) {
49          cout << "c " << graphType << " voi " << n << " dinh\n";
50          cout << "c So canh: " << edges.size() << "\n";
51          cout << "p edge " << n << " " << edges.size() << "\n";
52
53          for (const auto& edge : edges) {
54              cout << "e " << edge.first << " " << edge.second << "\n";
55          }
56      }
57
58      // Kiem tra tinh chat do thi day du
59      static bool isComplete(const vector<pair<int, int>>& edges, int
```

```
        n) {
60          if (edges.size() != countCompleteEdges(n)) {
61              return false;
62          }
63
64          // Tao ma tran ke de kiem tra
65          vector<vector<bool>> adj(n + 1, vector<bool>(n + 1, false));
66
67          for (const auto& edge : edges) {
68              adj[edge.first][edge.second] = true;
69              adj[edge.second][edge.first] = true;
70          }
71
72          // Kiem tra moi cap dinh
73          for (int i = 1; i <= n; i++) {
74              for (int j = i + 1; j <= n; j++) {
75                  if (!adj[i][j]) {
76                      return false;
77                  }
78              }
79          }
80
81          return true;
82      }
83  };
84
85  int main() {
86      cout << "=== DO THI DAY DU K_5 ===\n";
87      int n = 5;
88      auto completeEdges = CompleteGraphs::createComplete(n);
89      CompleteGraphs::exportDIMACS(completeEdges, n, "Do thi day du");
90
91      cout << "\nSo canh ly thuyet: " <<
           CompleteGraphs::countCompleteEdges(n) << "\n";
92      cout << "Kiem tra tinh day du: " <<
           (CompleteGraphs::isComplete(completeEdges, n) ? "Dung" :
           "Sai") << "\n";
93
94      cout << "\n=== DO THI HAI PHAN DAY DU K_{3,4} ===\n";
95      int p = 3, q = 4;
96      auto bipartiteEdges = CompleteGraphs::createCompleteBipartite(p,
           q);
97      CompleteGraphs::exportDIMACS(bipartiteEdges, p + q, "Do thi hai
           phan day du");
98
99      cout << "\nSo canh ly thuyet: " <<
           CompleteGraphs::countBipartiteEdges(p, q) << "\n";
100
101     return 0;
102 }
```

6.5 Code Python

Listing 7: Tao do thi day du va do thi hai phan day du - Python

```python
import itertools
import matplotlib.pyplot as plt
import numpy as np

class CompleteGraphs:
    @staticmethod
    def create_complete(n):
        """Tao do thi day du K_n"""
        edges = []
        for i in range(1, n + 1):
            for j in range(i + 1, n + 1):
                edges.append((i, j))
        return edges

    @staticmethod
    def create_complete_bipartite(p, q):
        """Tao do thi hai phan day du K_{p,q}"""
        edges = []
        # Tap A: cac dinh tu 1 den p
        # Tap B: cac dinh tu p+1 den p+q
        for i in range(1, p + 1):
            for j in range(p + 1, p + q + 1):
                edges.append((i, j))
        return edges

    @staticmethod
    def count_complete_edges(n):
        """Tinh so canh cua do thi day du"""
        return n * (n - 1) // 2

    @staticmethod
    def count_bipartite_edges(p, q):
        """Tinh so canh cua do thi hai phan day du"""
        return p * q

    @staticmethod
    def export_dimacs(edges, n, graph_type):
        """Xuat do thi theo dinh dang DIMACS"""
        print(f"c {graph_type} voi {n} dinh")
        print(f"c So canh: {len(edges)}")
        print(f"p edge {n} {len(edges)}")

        for u, v in edges:
            print(f"e {u} {v}")

    @staticmethod
    def is_complete(edges, n):
        """Kiem tra tinh chat do thi day du"""
        if len(edges) != CompleteGraphs.count_complete_edges(n):
            return False

        # Tao tap hop cac canh de kiem tra nhanh
```

```
53          edge_set = set()
54          for u, v in edges:
55              edge_set.add((min(u, v), max(u, v)))
56
57          # Kiem tra moi cap dinh
58          for i in range(1, n + 1):
59              for j in range(i + 1, n + 1):
60                  if (i, j) not in edge_set:
61                      return False
62
63          return True
64
65      @staticmethod
66      def visualize_complete(n):
67          """Tao toa do de ve do thi day du"""
68          coords = {}
69          for i in range(1, n + 1):
70              angle = 2 * np.pi * (i - 1) / n
71              coords[i] = (np.cos(angle), np.sin(angle))
72          return coords
73
74      @staticmethod
75      def visualize_bipartite(p, q):
76          """Tao toa do de ve do thi hai phan"""
77          coords = {}
78
79          # Tap A (ben trai)
80          for i in range(1, p + 1):
81              coords[i] = (-1, (i - 1) - (p - 1) / 2)
82
83          # Tap B (ben phai)
84          for j in range(p + 1, p + q + 1):
85              coords[j] = (1, (j - p - 1) - (q - 1) / 2)
86
87          return coords
88
89  if __name__ == "__main__":
90      print("=== DO THI DAY DU K_5 ===")
91      n = 5
92      complete_edges = CompleteGraphs.create_complete(n)
93      CompleteGraphs.export_dimacs(complete_edges, n, "Do thi day du")
94
95      print(f"\nSo canh ly thuyet:
            {CompleteGraphs.count_complete_edges(n)}")
96      print(f"Kiem tra tinh day du: {'Dung' if
            CompleteGraphs.is_complete(complete_edges, n) else 'Sai'}")
97
98      print(f"\n=== DO THI HAI PHAN DAY DU K_{{3,4}} ===")
99      p, q = 3, 4
100     bipartite_edges = CompleteGraphs.create_complete_bipartite(p, q)
101     CompleteGraphs.export_dimacs(bipartite_edges, p + q, "Do thi hai
            phan day du")
102
```

```
    print(f"\nSo canh ly thuyet:
       {CompleteGraphs.count_bipartite_edges(p, q)}")

    # In toa do de ve
    print(f"\n=== TOA DO DE VE DO THI HAI PHAN ===")
    coords = CompleteGraphs.visualize_bipartite(p, q)
    print("Tap A (mau xanh):")
    for i in range(1, p + 1):
        x, y = coords[i]
        print(f"Dinh {i}: ({x:.2f}, {y:.2f})")

    print("Tap B (mau do):")
    for j in range(p + 1, p + q + 1):
        x, y = coords[j]
        print(f"Dinh {j}: ({x:.2f}, {y:.2f})")
```

7 Bài tập 1.5: Ma trận kề mở rộng

7.1 Mô tả bài toán

Cần implement ma trận kề mở rộng (extended adjacency matrix) với các tính năng:

- Biểu diễn đồ thị có trọng số
- Hỗ trợ đồ thị có hướng và vô hướng
- Đánh số đỉnh bắt đầu từ 1
- Các phép toán cơ bản trên ma trận

7.2 Lý thuyết

Ma trận kề: Là ma trận vuông A kích thước $n \times n$ với $A[i][j] = 1$ nếu có cạnh từ đỉnh i đến đỉnh j, ngược lại $A[i][j] = 0$.

Ma trận kề có trọng số: $A[i][j] = w(i, j)$ nếu có cạnh từ i đến j với trọng số $w(i, j)$, ngược lại $A[i][j] = \infty$ hoặc 0.

Tính chất:

- Đồ thị vô hướng: Ma trận đối xứng ($A[i][j] = A[j][i]$)
- Đường chéo chính: $A[i][i] = 0$ (không có khuyên)
- Không gian: $O(n^2)$
- Thời gian kiểm tra cạnh: $O(1)$

7.3 Code C++

Listing 8: Ma trận kề mở rộng - C++

```
#include <iostream>
#include <vector>
#include <iomanip>
#include <limits>
```

```cpp
#include <fstream>
#include <sstream>
using namespace std;

const int INF = numeric_limits<int>::max();

class ExtendedAdjacencyMatrix {
private:
    int n; // so dinh
    vector<vector<int>> matrix;
    bool isDirected;
    bool isWeighted;

public:
    ExtendedAdjacencyMatrix(int vertices, bool directed = false, bool
       weighted = false)
        : n(vertices), isDirected(directed), isWeighted(weighted) {
        matrix.assign(n + 1, vector<int>(n + 1, weighted ? INF : 0));

        // Khoi tao duong cheo chinh = 0
        for (int i = 1; i <= n; i++) {
            matrix[i][i] = 0;
        }
    }

    // Them canh
    void addEdge(int u, int v, int weight = 1) {
        if (u < 1 || u > n || v < 1 || v > n) {
            cout << "Loi: Dinh khong hop le!\n";
            return;
        }

        matrix[u][v] = weight;

        if (!isDirected) {
            matrix[v][u] = weight;
        }
    }

    // Xoa canh
    void removeEdge(int u, int v) {
        if (u < 1 || u > n || v < 1 || v > n) return;

        matrix[u][v] = isWeighted ? INF : 0;

        if (!isDirected) {
            matrix[v][u] = isWeighted ? INF : 0;
        }
    }

    // Kiem tra ton tai canh
    bool hasEdge(int u, int v) const {
        if (u < 1 || u > n || v < 1 || v > n) return false;
```

```
57
58          if (isWeighted) {
59              return matrix[u][v] != INF;
60          } else {
61              return matrix[u][v] != 0;
62          }
63      }
64
65      // Lay trong so canh
66      int getWeight(int u, int v) const {
67          if (!hasEdge(u, v)) return isWeighted ? INF : 0;
68          return matrix[u][v];
69      }
70
71      // Tinh bac dinh
72      int getDegree(int u) const {
73          if (u < 1 || u > n) return -1;
74
75          int degree = 0;
76          for (int v = 1; v <= n; v++) {
77              if (hasEdge(u, v)) degree++;
78          }
79
80          return degree;
81      }
82
83      // Tinh bac vao (do thi co huong)
84      int getInDegree(int u) const {
85          if (!isDirected || u < 1 || u > n) return -1;
86
87          int inDegree = 0;
88          for (int v = 1; v <= n; v++) {
89              if (hasEdge(v, u)) inDegree++;
90          }
91
92          return inDegree;
93      }
94
95      // Tinh bac ra (do thi co huong)
96      int getOutDegree(int u) const {
97          if (!isDirected) return getDegree(u);
98          return getDegree(u);
99      }
100
101     // In ma tran
102     void printMatrix() const {
103         cout << "Ma tran ke " << (isDirected ? "co huong" : "vo
                huong")
104              << (isWeighted ? " co trong so" : "") << ":\n";
105
106         // In header
107         cout << setw(4) << "";
108         for (int j = 1; j <= n; j++) {
```

```
109             cout << setw(4) << j;
110         }
111         cout << "\n";
112 
113         // In ma tran
114         for (int i = 1; i <= n; i++) {
115             cout << setw(4) << i;
116             for (int j = 1; j <= n; j++) {
117                 if (isWeighted && matrix[i][j] == INF) {
118                     cout << setw(4) << "   ";
119                 } else {
120                     cout << setw(4) << matrix[i][j];
121                 }
122             }
123             cout << "\n";
124         }
125     }
126 
127     // Xuat ra dinh dang DIMACS
128     void exportDIMACS() const {
129         cout << "c Ma tran ke chuyen sang DIMACS\n";
130 
131         // Dem so canh
132         int edgeCount = 0;
133         for (int i = 1; i <= n; i++) {
134             for (int j = 1; j <= n; j++) {
135                 if (hasEdge(i, j)) {
136                     if (isDirected || i <= j) {
137                         edgeCount++;
138                     }
139                 }
140             }
141         }
142 
143         cout << "p edge " << n << " " << edgeCount << "\n";
144 
145         // Xuat cac canh
146         for (int i = 1; i <= n; i++) {
147             for (int j = 1; j <= n; j++) {
148                 if (hasEdge(i, j)) {
149                     if (isDirected || i <= j) {
150                         if (isWeighted) {
151                             cout << "e " << i << " " << j << " " <<
                                    matrix[i][j] << "\n";
152                         } else {
153                             cout << "e " << i << " " << j << "\n";
154                         }
155                     }
156                 }
157             }
158         }
159     }
160 
```

```
161     // Doc tu file DIMACS
162     void readFromDIMACS(const string& filename) {
163         ifstream file(filename);
164         string line;
165 
166         while (getline(file, line)) {
167             if (line[0] == 'c') continue;
168 
169             if (line[0] == 'p') {
170                 // Doc thong tin do thi
171                 stringstream ss(line);
172                 string p, edge;
173                 int vertices, edges;
174                 ss >> p >> edge >> vertices >> edges;
175 
176                 if (vertices != n) {
177                     cout << "Canh bao: So dinh khong khop!\n";
178                 }
179             }
180 
181             if (line[0] == 'e') {
182                 // Doc canh
183                 stringstream ss(line);
184                 string e;
185                 int u, v, w = 1;
186 
187                 ss >> e >> u >> v;
188                 if (isWeighted && ss >> w) {
189                     addEdge(u, v, w);
190                 } else {
191                     addEdge(u, v);
192                 }
193             }
194         }
195         file.close();
196     }
197 
198     // Kiem tra tinh lien thong (DFS)
199     bool isConnected() const {
200         if (isDirected) return false; // Can kiem tra lien thong manh
201 
202         vector<bool> visited(n + 1, false);
203         dfs(1, visited);
204 
205         for (int i = 1; i <= n; i++) {
206             if (!visited[i]) return false;
207         }
208 
209         return true;
210     }
211 
212 private:
213     void dfs(int u, vector<bool>& visited) const {
```

```
214             visited[u] = true;
215
216             for (int v = 1; v <= n; v++) {
217                 if (hasEdge(u, v) && !visited[v]) {
218                     dfs(v, visited);
219                 }
220             }
221         }
222     };
223
224     int main() {
225         cout << "=== DEMO MA TRAN KE MO RONG ===\n\n";
226
227         // Do thi vo huong khong trong so
228         cout << "1. Do thi vo huong khong trong so:\n";
229         ExtendedAdjacencyMatrix g1(5, false, false);
230         g1.addEdge(1, 2);
231         g1.addEdge(2, 3);
232         g1.addEdge(3, 4);
233         g1.addEdge(4, 5);
234         g1.addEdge(1, 5);
235
236         g1.printMatrix();
237         cout << "Bac dinh 1: " << g1.getDegree(1) << "\n";
238         cout << "Co canh (1,3)? " << (g1.hasEdge(1, 3) ? "Co" : "Khong")
               << "\n";
239         cout << "Lien thong? " << (g1.isConnected() ? "Co" : "Khong") <<
               "\n\n";
240
241         // Do thi co huong co trong so
242         cout << "2. Do thi co huong co trong so:\n";
243         ExtendedAdjacencyMatrix g2(4, true, true);
244         g2.addEdge(1, 2, 10);
245         g2.addEdge(2, 3, 20);
246         g2.addEdge(3, 4, 30);
247         g2.addEdge(1, 4, 5);
248
249         g2.printMatrix();
250         cout << "Bac ra dinh 1: " << g2.getOutDegree(1) << "\n";
251         cout << "Bac vao dinh 4: " << g2.getInDegree(4) << "\n";
252         cout << "Trong so canh (1,4): " << g2.getWeight(1, 4) << "\n\n";
253
254         cout << "3. Xuat dinh dang DIMACS:\n";
255         g2.exportDIMACS();
256
257         return 0;
258     }
```

7.4 Code Python

Listing 9: Ma trận kề mở rộng - Python

```
1 import numpy as np
```

```python
from typing import List, Tuple, Optional
import matplotlib.pyplot as plt
import networkx as nx

class ExtendedAdjacencyMatrix:
    def __init__(self, vertices: int, directed: bool = False,
       weighted: bool = False):
        """
        Khoi tao ma tran ke mo rong

        Args:
            vertices: So dinh
            directed: Do thi co huong hay khong
            weighted: Do thi co trong so hay khong
        """
        self.n = vertices
        self.is_directed = directed
        self.is_weighted = weighted

        # Khoi tao ma tran
        if weighted:
            self.matrix = np.full((vertices + 1, vertices + 1),
               np.inf)
            # Duong cheo chinh = 0
            np.fill_diagonal(self.matrix, 0)
        else:
            self.matrix = np.zeros((vertices + 1, vertices + 1),
               dtype=int)

    def add_edge(self, u: int, v: int, weight: int = 1) -> None:
        """Them canh vao do thi"""
        if not (1 <= u <= self.n and 1 <= v <= self.n):
            print("Loi: Dinh khong hop le!")
            return

        self.matrix[u][v] = weight

        if not self.is_directed:
            self.matrix[v][u] = weight

    def remove_edge(self, u: int, v: int) -> None:
        """Xoa canh khoi do thi"""
        if not (1 <= u <= self.n and 1 <= v <= self.n):
            return

        self.matrix[u][v] = np.inf if self.is_weighted else 0

        if not self.is_directed:
            self.matrix[v][u] = np.inf if self.is_weighted else 0

    def has_edge(self, u: int, v: int) -> bool:
        """Kiem tra ton tai canh"""
        if not (1 <= u <= self.n and 1 <= v <= self.n):
```

```
52              return False
53
54          if self.is_weighted:
55              return not np.isinf(self.matrix[u][v])
56          else:
57              return self.matrix[u][v] != 0
58
59      def get_weight(self, u: int, v: int) -> float:
60          """Lay trong so canh"""
61          if not self.has_edge(u, v):
62              return np.inf if self.is_weighted else 0
63          return self.matrix[u][v]
64
65      def get_degree(self, u: int) -> int:
66          """Tinh bac cua dinh"""
67          if not (1 <= u <= self.n):
68              return -1
69
70          degree = 0
71          for v in range(1, self.n + 1):
72              if self.has_edge(u, v):
73                  degree += 1
74
75          return degree
76
77      def get_in_degree(self, u: int) -> int:
78          """Tinh bac vao (do thi co huong)"""
79          if not self.is_directed or not (1 <= u <= self.n):
80              return -1
81
82          in_degree = 0
83          for v in range(1, self.n + 1):
84              if self.has_edge(v, u):
85                  in_degree += 1
86
87          return in_degree
88
89      def get_out_degree(self, u: int) -> int:
90          """Tinh bac ra (do thi co huong)"""
91          if not self.is_directed:
92              return self.get_degree(u)
93          return self.get_degree(u)
94
95      def print_matrix(self) -> None:
96          """In ma tran ke"""
97          print(f"Ma tran ke {'co huong' if self.is_directed else 'vo
               huong'}"
98                f"{' co trong so' if self.is_weighted else ''}:")
99
100         # In header
101         print("    ", end="")
102         for j in range(1, self.n + 1):
103             print(f"{j:4}", end="")
```

```
104         print()
105
106         # In ma tran
107         for i in range(1, self.n + 1):
108             print(f"{i:4}", end="")
109             for j in range(1, self.n + 1):
110                 if self.is_weighted and np.isinf(self.matrix[i][j]):
111                     print("    ", end="")
112                 else:
113                     print(f"{int(self.matrix[i][j]):4}", end="")
114             print()
115
116     def export_dimacs(self) -> None:
117         """Xuat ra dinh dang DIMACS"""
118         print("c Ma tran ke chuyen sang DIMACS")
119
120         # Dem so canh
121         edge_count = 0
122         for i in range(1, self.n + 1):
123             for j in range(1, self.n + 1):
124                 if self.has_edge(i, j):
125                     if self.is_directed or i <= j:
126                         edge_count += 1
127
128         print(f"p edge {self.n} {edge_count}")
129
130         # Xuat cac canh
131         for i in range(1, self.n + 1):
132             for j in range(1, self.n + 1):
133                 if self.has_edge(i, j):
134                     if self.is_directed or i <= j:
135                         if self.is_weighted:
136                             print(f"e {i} {j}
                                {int(self.matrix[i][j])}")
137                         else:
138                             print(f"e {i} {j}")
139
140     def read_from_dimacs(self, filename: str) -> None:
141         """Doc tu file DIMACS"""
142         with open(filename, 'r') as file:
143             for line in file:
144                 line = line.strip()
145                 if line.startswith('c'):
146                     continue
147                 elif line.startswith('p'):
148                     parts = line.split()
149                     vertices = int(parts[2])
150                     if vertices != self.n:
151                         print("Canh bao: So dinh khong khop!")
152                 elif line.startswith('e'):
153                     parts = line.split()
154                     u, v = int(parts[1]), int(parts[2])
155                     if self.is_weighted and len(parts) > 3:
```

```
                        w = int(parts[3])
                        self.add_edge(u, v, w)
                    else:
                        self.add_edge(u, v)

    def is_connected(self) -> bool:
        """Kiem tra tinh lien thong (DFS)"""
        if self.is_directed:
            return False  # Can kiem tra lien thong manh

        visited = [False] * (self.n + 1)
        self._dfs(1, visited)

        for i in range(1, self.n + 1):
            if not visited[i]:
                return False

        return True

    def _dfs(self, u: int, visited: List[bool]) -> None:
        """Ham phu DFS"""
        visited[u] = True

        for v in range(1, self.n + 1):
            if self.has_edge(u, v) and not visited[v]:
                self._dfs(v, visited)

    def get_adjacency_list(self) -> dict:
        """Chuyen doi sang danh sach ke"""
        adj_list = {i: [] for i in range(1, self.n + 1)}

        for i in range(1, self.n + 1):
            for j in range(1, self.n + 1):
                if self.has_edge(i, j):
                    if self.is_weighted:
                        adj_list[i].append((j, self.matrix[i][j]))
                    else:
                        adj_list[i].append(j)

        return adj_list

    def visualize(self) -> None:
        """Ve do thi bang matplotlib va networkx"""
        try:
            import matplotlib.pyplot as plt
            import networkx as nx

            # Tao do thi NetworkX
            if self.is_directed:
                G = nx.DiGraph()
            else:
                G = nx.Graph()

```

```
209             # Them dinh
210             G.add_nodes_from(range(1, self.n + 1))
211
212             # Them canh
213             for i in range(1, self.n + 1):
214                 for j in range(1, self.n + 1):
215                     if self.has_edge(i, j):
216                         if self.is_directed or i <= j:
217                             if self.is_weighted:
218                                 G.add_edge(i, j,
                                        weight=self.matrix[i][j])
219                             else:
220                                 G.add_edge(i, j)
221
222             # Ve do thi
223             plt.figure(figsize=(10, 8))
224             pos = nx.spring_layout(G, k=1, iterations=50)
225
226             # Ve cac dinh va canh
227             nx.draw_networkx_nodes(G, pos, node_color='lightblue',
228                                    node_size=500, alpha=0.8)
229             nx.draw_networkx_labels(G, pos, font_size=12,
                    font_weight='bold')
230
231             if self.is_directed:
232                 nx.draw_networkx_edges(G, pos, edge_color='gray',
233                                        arrows=True, arrowsize=20,
                                           alpha=0.6)
234             else:
235                 nx.draw_networkx_edges(G, pos, edge_color='gray',
                        alpha=0.6)
236
237             # Ve trong so (neu co)
238             if self.is_weighted:
239                 edge_labels = {}
240                 for i in range(1, self.n + 1):
241                     for j in range(1, self.n + 1):
242                         if self.has_edge(i, j) and (self.is_directed
                                or i <= j):
243                             if not np.isinf(self.matrix[i][j]):
244                                 edge_labels[(i, j)] =
                                        int(self.matrix[i][j])
245
246                 nx.draw_networkx_edge_labels(G, pos, edge_labels,
                        font_size=10)
247
248             plt.title(f"Do thi {'co huong' if self.is_directed else
                    'vo huong'}"
249                       f"{' co trong so' if self.is_weighted else ''}")
250             plt.axis('off')
251             plt.tight_layout()
252             plt.show()
253
```

```
        except ImportError:
            print("Can cai dat matplotlib va networkx de ve do thi")

if __name__ == "__main__":
    print("=== DEMO MA TRAN KE MO RONG ===\n")

    # 1. Do thi vo huong khong trong so
    print("1. Do thi vo huong khong trong so:")
    g1 = ExtendedAdjacencyMatrix(5, directed=False, weighted=False)
    g1.add_edge(1, 2)
    g1.add_edge(2, 3)
    g1.add_edge(3, 4)
    g1.add_edge(4, 5)
    g1.add_edge(1, 5)

    g1.print_matrix()
    print(f"Bac dinh 1: {g1.get_degree(1)}")
    print(f"Co canh (1,3)? {'Co' if g1.has_edge(1, 3) else 'Khong'}")
    print(f"Lien thong? {'Co' if g1.is_connected() else 'Khong'}\n")

    # 2. Do thi co huong co trong so
    print("2. Do thi co huong co trong so:")
    g2 = ExtendedAdjacencyMatrix(4, directed=True, weighted=True)
    g2.add_edge(1, 2, 10)
    g2.add_edge(2, 3, 20)
    g2.add_edge(3, 4, 30)
    g2.add_edge(1, 4, 5)

    g2.print_matrix()
    print(f"Bac ra dinh 1: {g2.get_out_degree(1)}")
    print(f"Bac vao dinh 4: {g2.get_in_degree(4)}")
    print(f"Trong so canh (1,4): {g2.get_weight(1, 4)}\n")

    print("3. Xuat dinh dang DIMACS:")
    g2.export_dimacs()

    print("\n4. Danh sach ke:")
    adj_list = g2.get_adjacency_list()
    for vertex, neighbors in adj_list.items():
        print(f"Dinh {vertex}: {neighbors}")
```

8 Bài tập 1.6: Biểu diễn đồ thị bằng danh sách kề

8.1 Mô tả bài toán

Danh sách kề (adjacency list) là một cách biểu diễn đồ thị hiệu quả về không gian, đặc biệt phù hợp với đồ thị thưa (sparse graph). Cần implement:

- Biểu diễn đồ thị có hướng và vô hướng
- Hỗ trợ đồ thị có trọng số
- Các phép toán cơ bản: thêm/xóa đỉnh, cạnh

- Chuyển đổi giữa các định dạng khác nhau

8.2 Lý thuyết

Danh sách kề: Mỗi đỉnh v có một danh sách chứa tất cả các đỉnh kề với v.

Ưu điểm:

- Không gian: $O(V + E)$ thay vì $O(V^2)$ như ma trận kề
- Hiệu quả với đồ thị thưa
- Duyệt các đỉnh kề nhanh: $O(\deg(v))$

Nhược điểm:

- Kiểm tra tồn tại cạnh: $O(\deg(v))$ thay vì $O(1)$
- Phức tạp hơn trong implementation

8.3 Code C++

Listing 10: Danh sach ke - C++

```cpp
#include <iostream>
#include <vector>
#include <list>
#include <map>
#include <algorithm>
#include <fstream>
#include <sstream>
using namespace std;

template<typename T = int>
class AdjacencyList {
private:
    int n; // so dinh
    vector<list<pair<int, T>>> adj; // adj[u] = {(v1, w1), (v2, w2), ...}
    bool isDirected;
    bool isWeighted;

public:
    AdjacencyList(int vertices, bool directed = false, bool weighted = false)
        : n(vertices), isDirected(directed), isWeighted(weighted) {
        adj.resize(n + 1);
    }

    // Them canh
    void addEdge(int u, int v, T weight = T(1)) {
        if (u < 1 || u > n || v < 1 || v > n) {
            cout << "Loi: Dinh khong hop le!\n";
            return;
        }

```

```
31          // Kiem tra canh da ton tai
32          if (hasEdge(u, v)) {
33              cout << "Canh (" << u << "," << v << ") da ton tai!\n";
34              return;
35          }
36
37          if (isWeighted) {
38              adj[u].push_back({v, weight});
39          } else {
40              adj[u].push_back({v, T(1)});
41          }
42
43          if (!isDirected) {
44              if (isWeighted) {
45                  adj[v].push_back({u, weight});
46              } else {
47                  adj[v].push_back({u, T(1)});
48              }
49          }
50      }
51
52      // Xoa canh
53      void removeEdge(int u, int v) {
54          if (u < 1 || u > n || v < 1 || v > n) return;
55
56          adj[u].remove_if([v](const pair<int, T>& p) { return p.first
               == v; });
57
58          if (!isDirected) {
59              adj[v].remove_if([u](const pair<int, T>& p) { return
                   p.first == u; });
60          }
61      }
62
63      // Kiem tra ton tai canh
64      bool hasEdge(int u, int v) const {
65          if (u < 1 || u > n || v < 1 || v > n) return false;
66
67          for (const auto& edge : adj[u]) {
68              if (edge.first == v) return true;
69          }
70          return false;
71      }
72
73      // Lay trong so canh
74      T getWeight(int u, int v) const {
75          if (u < 1 || u > n || v < 1 || v > n) return T(0);
76
77          for (const auto& edge : adj[u]) {
78              if (edge.first == v) return edge.second;
79          }
80          return T(0);
81      }
```

```
82
83      // Lay danh sach dinh ke
84      vector<int> getNeighbors(int u) const {
85          vector<int> neighbors;
86          if (u < 1 || u > n) return neighbors;
87
88          for (const auto& edge : adj[u]) {
89              neighbors.push_back(edge.first);
90          }
91          return neighbors;
92      }
93
94      // Lay danh sach dinh ke voi trong so
95      vector<pair<int, T>> getNeighborsWithWeight(int u) const {
96          vector<pair<int, T>> neighbors;
97          if (u < 1 || u > n) return neighbors;
98
99          for (const auto& edge : adj[u]) {
100             neighbors.push_back(edge);
101         }
102         return neighbors;
103     }
104
105     // Tinh bac dinh
106     int getDegree(int u) const {
107         if (u < 1 || u > n) return -1;
108         return adj[u].size();
109     }
110
111     // Tinh bac vao (do thi co huong)
112     int getInDegree(int u) const {
113         if (!isDirected || u < 1 || u > n) return -1;
114
115         int inDegree = 0;
116         for (int v = 1; v <= n; v++) {
117             if (hasEdge(v, u)) inDegree++;
118         }
119         return inDegree;
120     }
121
122     // Dem so canh
123     int countEdges() const {
124         int edgeCount = 0;
125         for (int u = 1; u <= n; u++) {
126             edgeCount += adj[u].size();
127         }
128
129         if (!isDirected) edgeCount /= 2;
130         return edgeCount;
131     }
132
133     // In danh sach ke
134     void printAdjacencyList() const {
```

```
135         cout << "Danh sach ke " << (isDirected ? "co huong" : "vo
               huong")
136              << (isWeighted ? " co trong so" : "") << ":\n";
137
138         for (int u = 1; u <= n; u++) {
139             cout << "Dinh " << u << ": ";
140             for (const auto& edge : adj[u]) {
141                 if (isWeighted) {
142                     cout << "(" << edge.first << "," << edge.second
                           << ") ";
143                 } else {
144                     cout << edge.first << " ";
145                 }
146             }
147             cout << "\n";
148         }
149     }
150
151     // Xuat ra dinh dang DIMACS
152     void exportDIMACS() const {
153         cout << "c Danh sach ke chuyen sang DIMACS\n";
154         cout << "p edge " << n << " " << countEdges() << "\n";
155
156         for (int u = 1; u <= n; u++) {
157             for (const auto& edge : adj[u]) {
158                 int v = edge.first;
159                 if (isDirected || u <= v) {
160                     if (isWeighted) {
161                         cout << "e " << u << " " << v << " " <<
                               edge.second << "\n";
162                     } else {
163                         cout << "e " << u << " " << v << "\n";
164                     }
165                 }
166             }
167         }
168     }
169
170     // Duyet DFS
171     void dfs(int start) const {
172         vector<bool> visited(n + 1, false);
173         cout << "DFS tu dinh " << start << ": ";
174         dfsUtil(start, visited);
175         cout << "\n";
176     }
177
178 private:
179     void dfsUtil(int u, vector<bool>& visited) const {
180         visited[u] = true;
181         cout << u << " ";
182
183         for (const auto& edge : adj[u]) {
184             int v = edge.first;
```

```
            if (!visited[v]) {
                dfsUtil(v, visited);
            }
        }
    }
};

int main() {
    cout << "=== DEMO DANH SACH KE ===\n\n";

    // Do thi vo huong co trong so
    cout << "1. Do thi vo huong co trong so:\n";
    AdjacencyList<int> g1(5, false, true);
    g1.addEdge(1, 2, 10);
    g1.addEdge(2, 3, 15);
    g1.addEdge(3, 4, 20);
    g1.addEdge(4, 5, 25);
    g1.addEdge(1, 5, 30);
    g1.addEdge(2, 4, 35);

    g1.printAdjacencyList();
    cout << "So canh: " << g1.countEdges() << "\n";
    cout << "Bac dinh 2: " << g1.getDegree(2) << "\n";
    cout << "Trong so canh (2,4): " << g1.getWeight(2, 4) << "\n\n";

    // DFS
    g1.dfs(1);

    cout << "\n2. Xuat dinh dang DIMACS:\n";
    g1.exportDIMACS();

    return 0;
}
```