Weekly Homework 8 Report

24120113 Trần Hoàng Nhi

Exercises

1. The file contains an adjacency matrix. Read the file and output the corresponding adjacency list.

---

vector<vector<int>> convertMatrixToList(const string& filename);
Đọc file đầu vào ma trận kề (kích thước n x n).
Duyệt ma trận, với mỗi vị trí [i][j] = 1, thêm đỉnh j vào danh sách kề của đỉnh i.
Trả về vector chứa danh sách kề của từng đỉnh.

---

2. The file contains an adjacency list. Read the file and output the corresponding adjacency matrix.

---

vector<vector<int>> convertListToMatrix(const string& filename);
Đọc file đầu vào danh sách kề.
Khởi tạo ma trận n x n với giá trị 0.
Với mỗi đỉnh i, đánh dấu vị trí [i][j] = 1 nếu j nằm trong danh sách kề của i.

---

adjacency matrix	adjacency list
9 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0	9 2 2 5 1 6 1 6 1 4 1 5 2 3 7 0 2 2 8 0

Bảng 1: Adjacency matrix and corresponding Adjacency list

3. Implement functions to provide the following information about a given graph:

// Directed or Undirected Graph. bool isDirected(const vector<vector<int>>& adjMatrix);

So sánh từng cặp (i, j) và (j, i) trong ma trận kề. Nếu có vị trí khác nhau, đồ thị có hướng.

// The number of vertices.
int countVertices(const vector<vector<int>>& adjMatrix);

Số đỉnh chính là kích thước của ma trận.

// The number of edges.
int countEdges(const vector<vector<int>>& adjMatrix);

Đếm số phần tử 1 trong ma trận. Nếu đồ thị vô hướng, số cạnh bằng tổng chia 2.

// List of isolated vertices. vector<int> getIsolatedVertices(const vector<vector<int>>& adjMatrix);

Tìm các đỉnh không có cạnh đi ra cũng không có cạnh đi vào.

// Undirected Graph. bool isCompleteGraph(const vector<vector<int>>& adjMatrix);

Kiểm tra mọi cặp đỉnh khác nhau có cạnh nối hay không. Đường chéo bằng 0.

// Undirected Graph bool isBipartite(const std::vector<std::vector<int>>& adjMatrix);

Dùng BFS tô màu 2 màu cho các đỉnh, nếu có đỉnh kề cùng màu trả về false.

// Undirected Graph
bool isCompleteBipartite(const vector<vector<int>>& adjMatrix);
Tương tự bipartite, nhưng kiểm tra đầy đủ các cạnh giữa 2 tập.

---

4. Generate a base undirected graph from a given directed graph.

---

vector<vector<int>> convertToUndirectedGraph(const vector<vector<int>>& adjMatrix);

Với mỗi cạnh (i, j), nếu có cạnh một chiều, thêm cạnh ngược lại.

---

5. Generate a complement graph from a given undirected graph and output its adjacency matrix (*undirected graph).

---

vector<vector<int>> getComplementGraph(const vector<vector<int>>& adjMatrix);
Với đồ thị vô hướng, tạo ma trận bù bằng cách đảo 0 thành 1 (ngoại trừ đường chéo).

---

6. Determine the Euler cycle from a given graph using Hierholzer's Algorithm.

---

vector<int> findEulerCycle(const vector<vector<int>>& adjMatrix);
Sử dụng thuật toán Hierholzer, bắt đầu từ đỉnh bất kỳ có bậc lớn hơn 0, đi theo cạnh chưa đi và quay lại tạo chu trình Euler.

---

7. Find the spanning tree of a given graph using(*undirected graph):

---

vector<vector<int>> dfsSpanningTree(const vector<vector<int>>& adjMatrix, int start); vector<vector<int>>
Dùng DFS để tạo cây khung.
bfsSpanningTree(const vector<vector<int>>& adjMatrix, int start);
Dùng BFS để tạo cây khung.

---

8. Verify the connection between two vertices of a given graph.

---

bool isConnected(int u, int v, const vector<vector<int>>& adjMatrix);
Dùng BFS hoặc DFS từ đỉnh u, kiểm tra xem có thể đến đỉnh v không.

---

9. Find the shortest path between two vertices of a given graph using (*Weighted Graph):

---

vector<int> dijkstra(int start, int end, const vector<vector<int>>& adjMatrix);
Dùng để tìm đường đi ngắn nhất từ đỉnh nguồn đến các đỉnh khác trong đồ thị có trọng số không âm. Thuật toán duy trì tập đỉnh đã biết khoảng cách ngắn nhất, mỗi bước mở rộng từ đỉnh gần nhất chưa xử lý, cập nhật khoảng cách tới các đỉnh kề.
vector<int> bellmanFord(int start, int end, const vector<vector<int>>& adjMatrix);

Dùng cho đồ thị có trọng số âm, tìm đường đi ngắn nhất từ nguồn đến các đỉnh khác bằng cách lặp nhiều lần, mỗi lần cập nhật khoảng cách qua tất cả các cạnh. Nếu sau |V|-1 lần cập nhật mà vẫn thay đổi, báo hiệu đồ thị có chu trình âm.

Git/Github