BÀI TẬP THỰC HÀNH 07: ĐỒ THỊ

1 Nội dung

Tập tin "graph.txt" là tập tin chứa thông tin của một đồ thị. Tập tin có thể có nội dung như sau:

```
5
0 1 0 0 1
1 0 1 0 0
0 1 0 0 1
0 0 0 0 1
1 0 1 1 0
```

Trong đó:

- Dòng đầu tiên chứa số nguyên `n`, cho biết số đỉnh của đồ thị.
- `n` dòng tiếp theo, mỗi dòng chứa `n` số nguyên ứng với các phần tử trong ma trận kề.

Sinh viên được yêu cầu đọc thông tin của đồ thị từ tập tin trên vào cấu trúc dữ liệu `Graph` và thực hiện một số thao tác trên đồ thị đó.
Lưu ý: tập tin `graph.txt` sinh viên tự tạo ra theo mẫu trên.

2 Thực hành

Cho struct `Graph` được định nghĩa như sau:

```
#define MAX 100

struct Graph
{
    int num_vertices;               // number of vertices
    int adjacency_matrix[MAX][MAX]; // adjacency matrix
};
```

Không bắt buộc: Sinh viên cài đặt cấu trúc dữ liệu trên và hàm yêu cầu tương ứng sử dụng Lập trình Hướng đối tượng

Yêu cầu

Sinh viên hãy định nghĩa các hàm sau:

1. Hàm tạo đồ thị từ tập tin:
   - `Graph CreateGraphFromFile(string file_name)`
   - Đầu vào: `file_name` là tên tập tin đầu vào (trong bài tập thực hành này là `graph.txt`)
   - Đầu ra: đồ thị đọc từ tập tin, kiểu dữ liệu `Graph`
2. Hàm hiển thị đồ thị:
   - `void DisplayGraph(Graph g)`
   - Đầu vào: `g` là đồ thị muốn hiển thị.

3. Hàm kiểm tra đồ thị có hợp lệ không:
   - `bool IsValidGraph(Graph g)`
   - Đầu vào: `g` là đồ thị muốn kiểm tra
   - Đầu ra: Trả về `true` nếu `g` là đồ thị hợp lệ, `false` nếu ngược lại
4. Kiểm tra một đồ thị có phải là đồ thị vô hướng:
   - `bool IsUndirectedGraph(Graph g)`
   - Đầu vào: `g` là đồ thị muốn kiểm tra
   - Đầu ra: Trả về `true` nếu `g` là đồ thị vô hướng, `false` nếu ngược lại
5. Đếm số lượng cạnh trong đồ thị:
   - `int CountEdge(Graph g)`
   - Đầu vào: `g` là đồ thị muốn đếm số cạnh
   - Đầu ra: số lượng cạnh trong đồ thị
6. Đếm số lượng đỉnh có yêu cầu:
   - `int CountVertices(Graph g, int flag)`
   - Đầu vào: - `g` là đồ thị muốn đếm đỉnh
     - `flag` là cờ hiệu, nếu `flag = 1` hàm thực hiện đếm số lượng đỉnh có bậc lẻ, nếu `flag = 0` hàm thực hiện đếm số lượng đỉnh có bậc chẵn
   - Đầu ra: số lượng đỉnh đếm được theo yêu cầu
7. Duyệt đồ thị theo chiều rộng:
   - `void BFS(Graph g, int start_vertex)`
   - Đầu vào: - `g` là đồ thị muốn duyệt theo chiều rộng
     - `start_vertex` là đỉnh bắt đầu duyệt
8. Duyệt đồ thị theo chiều sâu:
   - `void DFS(Graph g, int start_vertex)`
   - Đầu vào: - `g` là đồ thị muốn duyệt theo chiều sâu
     - `start_vertex` là đỉnh bắt đầu duyệt
9. Kiểm tra đồ thị có liên thông không:
   - `bool IsConnectedGraph(Graph g)`
   - Đầu vào: `g` là đồ thị muốn kiểm tra
   - Đầu ra: Trả về `true` nếu `g` là đồ thị có liên thông, `false` nếu ngược lại
10. Đếm số thành phần liên thông của đồ thị:
    - `int CountConnectedComponents(Graph g)`
    - Đầu vào: `g` là đồ thị muốn đếm số lượng thành phần liên thông
    - Đầu ra: số lượng thành phần liên thông có trong đồ thị
11. Tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh còn lại:
    + Theo thuật toán Dijkstra:
      - `void FindShortestPathDijkstra(Graph g, int start_vertex)`

  - Đầu vào: - `g` là đồ thị muốn tìm kiếm đường đi ngắn nhất
    - `start_vertex` là đỉnh bắt đầu duyệt

+ Theo thuật toán Floyd:
  - `void FindShortestPathFloyd(Graph g, int start_vertex)`
  - Đầu vào: - `g` là đồ thị muốn tìm kiếm đường đi ngắn nhất
    - `start_vertex` là đỉnh bắt đầu duyệt

+ Theo thuật toán Bellman:
  - `void FindShortestPathBellman(Graph g, int start_vertex)`
  - Đầu vào: - `g` là đồ thị muốn tìm kiếm đường đi ngắn nhất
    - `start_vertex` là đỉnh bắt đầu duyệt

3 Quy định nộp bài

- Sinh viên nộp toàn bộ mã nguồn liên quan thông qua tập tin MSSV.zip hoặc MSSV.rar.
- Mỗi phần cần được đặt trong thư mục riêng. Tất cả nằm trong thư mục MSSV (Lưu ý: chỉ nộp file .h và .cpp).
- Các bài nộp sai quy định sẽ bị 0 điểm.
- Các bài làm giống nhau sẽ bị 0 điểm môn học.