**BÀI TẬP TUẦN 9 - ĐỒ THỊ**

Sinh viên tuân thủ theo quy định trong file “**QUY ĐỊNH ĐỐI VỚI MÔN HỌC LẬP TRÌNH C++**”

Sinh viên không cần nộp câu 0, do câu 0 chỉ mang tính chất khởi tạo.

0. Định nghĩa một struct hoặc class dùng để lưu đồ thị có V đỉnh.

1. Viết chương trình cho phép người dùng nhập vào các cạnh của đồ thị và biểu diễn đồ thị dưới dạng một ma trận kề (adjacency matrix).

Với std in có định dạng là:

số đỉnh

số cạnh

n dòng tiếp theo định nghĩa cạnh: <source node> <target node>

Ví dụ: khi người dùng nhập vào:

graph.add\_edge(0, 1)

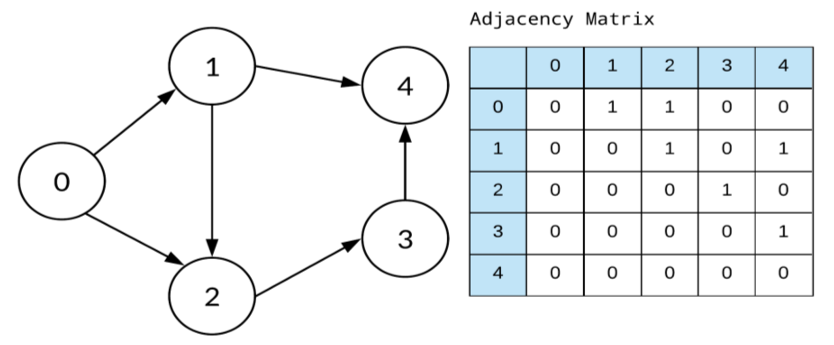
graph.add\_edge(0, 2)

graph.add\_edge(1, 2)

graph.add\_edge(1, 4)

graph.add\_edge(2, 3)

graph.add\_edge(3, 4)



| Std in | Std out |
| --- | --- |
| 5  6  0 1  0 2  1 2  1 4  2 3  3 4 | 0 1 1 0 0  0 0 1 0 1  0 0 0 1 0  0 0 0 0 1  0 0 0 0 0 |

2. Deep first search (duyệt theo chiều sâu). Dùng DFS duyệt qua tất cả các node trong đồ thị. In ra màn hình tương ứng.

Trường hợp không biết mở node nào, thì ưu tiên thứ tự 0, 1, 2, 3, …

Sử dụng đồ thị được định nghĩa ở câu 1. Sinh viên không cần nhập std in.

| Std in | Std out |
| --- | --- |
|  | 0 1 2 3 4 |

3. Breadth first search (duyệt theo chiều rộng). Dùng BFS duyệt qua tất cả các node trong đồ thị. In ra màn hình tương ứng.

Trường hợp không biết mở node nào, thì ưu tiên thứ tự 0, 1, 2, 3, …

Sử dụng đồ thị được định nghĩa ở câu 1. Sinh viên không cần nhập std in.

| Std in | Std out |
| --- | --- |
|  | 0 1 2 4 3 |

4. Viết chương trình tính in degree và out degree.

1. In degree: số cạnh đi vào node đó.
2. Out degree: số cạnh đi ra từ node đó.
3. In ra màn hình với nội dung: <tên node> <in degree> <out degree>

| Std in | Std out |
| --- | --- |
|  | 0 0 2  1 1 2  2 2 1  3 1 1  4 2 0 |

5. Viết chương trình kiểm tra có đường đi từ S đến G hay không. Dùng thuật toán DFS

In ra màn hình **đường đi** hoặc “non-exist path” tương ứng.

Trường hợp không biết mở node nào, thì ưu tiên thứ tự 0, 1, 2, 3, …

| Std in | Std out |
| --- | --- |
| 0  4 | 0 1 2 3 4 |
| 4  0 | non-exist path |

6. Viết chương trình kiểm tra có đường đi từ S đến G hay không. Dùng thuật toán BFS

In ra màn hình **đường đi** hoặc “non-exist path” tương ứng.

Trường hợp không biết mở node nào, thì ưu tiên thứ tự 0, 1, 2, 3, …

| Std in | Std out |
| --- | --- |
| 0  4 | 0 1 4 |
| 4  0 | non-exist path |