Họ tên: Huỳnh Công Tính

MSSV: 3124411315

Lớp: DCT124C6

Github: congtinh06

Đề: 5 ý tưởng bài code và 10 câu lí thuyết

Câu 1:

**Main Idea:**

* Chương trình đọc số đỉnh và ma trận kề từ file MaTranKe.inp.
* Tính bậc của từng đỉnh:
  + Nếu có khuyên tại đỉnh i → cộng 2 lần.
  + Nếu có cạnh i—j → cộng thêm 1.
* Ghi kết quả ra file MaTranKe.out.

**Cách làm và hướng giải:**

**Bước 1: Đọc dữ liệu**

* Số đỉnh của đồ thị: **6**.
* Ma trận kề 6×6 được cho trong file input.

**Bước 2: Nguyên tắc tính bậc đỉnh**

* Với đồ thị vô hướng:
  + Bậc của một đỉnh = tổng tất cả các giá trị trên hàng tương ứng.
  + Nếu có khuyên tại đỉnh i (tức a[i][i]=1a[i][i] = 1a[i][i]=1) thì phải cộng 2 × a[i][i].

**Bước 3: Tính toán cụ thể**

* Đỉnh 1: hàng 1 là 0 1 1 0 0 0 → tổng = 2 → bậc = 2.
* Đỉnh 2: hàng 2 là 1 0 1 1 1 0 → tổng = 4 → bậc = 4.
* Đỉnh 3: hàng 3 là 1 1 0 1 0 0 → tổng = 3 → bậc = 3.
* Đỉnh 4: hàng 4 là 0 1 1 0 1 1 → tổng = 4 → bậc = 4.
* Đỉnh 5: hàng 5 là 0 1 0 1 0 1 → tổng = 3 → bậc = 3.
* Đỉnh 6: hàng 6 là 0 0 0 1 1 0 → tổng = 2 → bậc = 2.
* Và xuất ra kết quả ở MaTranKe.inp:

6

2

4

3

4

3

2

Câu 2:

**Main idea:**

* Chương trình đọc ma trận kề của đồ thị có hướng từ file BacVaoBacRa.inp.
* Tính bậc vào (in-degree) và bậc ra (out-degree) cho từng đỉnh.
* Xuất kết quả ra file BacVaoBacRa.out.

**Cách làm và hướng giải**

Bước 1: Đọc dữ liệu

* Đọc số đỉnh nnn.
* Đọc ma trận kề n×nn \times nn×n từ file input.

Bước 2: Nguyên tắc tính bậc

* Với đồ thị có hướng:
  + Bậc ra (out-degree) của đỉnh i = tổng các phần tử trên hàng i (vì hàng i biểu diễn số cung đi ra từ i).
  + Bậc vào (in-degree) của đỉnh j = tổng các phần tử trên cột j (vì cột j biểu diễn số cung đi vào j).

Bước 3: Tính toán cụ thể

0 1 1 0 0 0

0 0 0 1 1 0

0 1 0 0 1 0

0 0 0 0 1 1

0 1 0 0 0 1

0 0 0 0 0 0

Đỉnh 1: hàng 1 có 1 1 → outDeg[1] = 2; cột 1 không có số 1 → inDeg[1] = 0.

Đỉnh 2: hàng 2 có 1 1 → outDeg[2] = 2; cột 2 có (1,3,5) → inDeg[2] = 3.

Đỉnh 3: hàng 3 có 1 1 → outDeg[3] = 2; cột 3 có (1) → inDeg[3] = 1.

Đỉnh 4: hàng 4 có 1 1 → outDeg[4] = 2; cột 4 có (2) → inDeg[4] = 1.

Đỉnh 5: hàng 5 có 1 1 → outDeg[5] = 2; cột 5 có (2,3,4) → inDeg[5] = 3.

Đỉnh 6: hàng 6 toàn 0 → outDeg[6] = 0; cột 6 có (4,5) → inDeg[6] = 2.

* Và kết quả cho file BacVaoBacRa.out là:

6

0 2

3 2

1 2

1 2

3 2

2 0

Câu 3:

**Main idea**

Chương trình đọc dữ liệu đồ thị dưới dạng danh sách kề từ file input. Với mỗi đỉnh, danh sách các đỉnh kề được lưu vào adjList[i]. Sau đó, bậc của đỉnh được tính bằng số phần tử trong danh sách đó (deg[i] = adjList[i].size()). Cuối cùng, chương trình ghi ra file output số đỉnh và bậc của từng đỉnh.

**Cách làm và hướng giải**

Bước 1: Đọc dữ liệu

* Đọc số đỉnh nnn.
* Với mỗi đỉnh iii, đọc danh sách các đỉnh kề (các đỉnh có cạnh nối trực tiếp với iii).
* Danh sách kề của mỗi đỉnh kết thúc bằng giá trị **-1**.
* Lưu vào adjList[i].

Bước 2: Nguyên tắc tính bậc

* Bậc của một đỉnh = số lượng phần tử trong danh sách kề của đỉnh đó.
* Trong chương trình: deg[i] = adjList[i].size();

Bước 3: Tính toán cụ thể:

5

2 3 -1

1 2 3 -1

1 2 5 -1

5 -1

3 4 -1

Đỉnh 1: danh sách {2, 3} → bậc = 2

Đỉnh 2: danh sách {1, 2, 3} → bậc = 3

Đỉnh 3: danh sách {1, 2, 5} → bậc = 3

Đỉnh 4: danh sách {5} → bậc = 1

Đỉnh 5: danh sách {3, 4} → bậc = 2

* Và kết quả:

5

2

3

3

1

2

Câu 4:

**Main idea**

Chương trình đọc số đỉnh n và số cạnh m của đồ thị, sau đó đọc từng cạnh (u,v)(u, v)(u,v) và lưu vào danh sách cạnh edgeList. Khi duyệt danh sách cạnh, mỗi cạnh sẽ làm tăng bậc của hai đỉnh u và v. Kết quả (số đỉnh và bậc của từng đỉnh) được ghi ra file output.

**Cách làm và Hướng giải**

Bước 1: Đọc dữ liệu

* Đọc số đỉnh n và số cạnh m.
* Với mỗi cạnh, đọc cặp (u,v)(u, v)(u,v).
* Lưu vào danh sách cạnh edgeList.

Bước 2: Nguyên tắc tính bậc

* Với đồ thị vô hướng: mỗi cạnh (u,v)(u, v)(u,v) sẽ làm tăng bậc của cả 2 đỉnh:
  + deg[u]++
  + deg[v]++

Bước 3: Tính toán cụ thể:

5 6

1 2

1 3

2 3

2 5

3 5

4 5

* Đỉnh 1: cạnh (1,2), (1,3) → bậc = 2
* Đỉnh 2: cạnh (1,2), (2,3), (2,5) → bậc = 3
* Đỉnh 3: cạnh (1,3), (2,3), (3,5) → bậc = 3
* Đỉnh 4: cạnh (4,5) → bậc = 1
* Đỉnh 5: cạnh (2,5), (3,5), (4,5) → bậc = 3
* Và kết quả:

2

3

3

1

3

Câu 5:

**Main idea**

* Input: Đọc số đỉnh n, số cạnh m và danh sách các cạnh (u, v) của đồ thị.
* Xử lý: Với mỗi cạnh (u, v), tăng bậc của cả hai đỉnh:
* deg[u]++
* deg[v]++

(vì đồ thị vô hướng, mỗi cạnh góp phần vào bậc của cả 2 đỉnh).

* Output: In ra bậc của từng đỉnh từ 1 → n.

**Cách làm và Hướng giải**

Bước 1: Đọc dữ liệu

* Đọc số đỉnh n và số cạnh m.
* Với mỗi cạnh, đọc cặp (u, v).
* Lưu tất cả vào danh sách cạnh edgeList.

Bước 2: Nguyên tắc tính bậc

* Với đồ thị vô hướng, mỗi cạnh (u, v) sẽ làm tăng bậc của cả 2 đỉnh:
* deg[u]++
* deg[v]++
* Duyệt toàn bộ danh sách cạnh, cập nhật mảng deg[].

Bước 3: Tính toán cụ thể

5 6

1 2

1 3

2 3

2 5

3 5

4 5

* Đỉnh 1: có cạnh (1,2), (1,3) → bậc = 2
* Đỉnh 2: có cạnh (1,2), (2,3), (2,5) → bậc = 3
* Đỉnh 3: có cạnh (1,3), (2,3), (3,5) → bậc = 3
* Đỉnh 4: có cạnh (4,5) → bậc = 1
* Đỉnh 5: có cạnh (2,5), (3,5), (4,5) → bậc = 3
* Và kết quả:

3

3

1

3



**Định nghĩa:**  
Một **đồ thị** GGG là một cặp G=(V,E)G=(V,E)G=(V,E) với VVV là tập các đỉnh (vertices) và EEE là tập các cạnh (edges). Mỗi cạnh nối hai đỉnh (có thể có hướng hoặc không).

**Thành phần:**

* Tập đỉnh V={v1,v2,…,vn}V=\{v\_1,v\_2,\dots,v\_n\}V={v1​,v2​,…,vn​}.
* Tập cạnh EEE. Với đồ thị vô hướng, cạnh là tập 2-phan tử {u,v}\{u,v\}{u,v}. Với đồ thị có hướng, cung là cặp có thứ tự (u,v)(u,v)(u,v).

**Các loại đồ thị chính:**

* **Đồ thị vô hướng (undirected):** cạnh không có hướng.
* **Đồ thị có hướng (directed / digraph):** mỗi cạnh là cung có hướng.
* **Đồ thị có trọng số (weighted):** mỗi cạnh kèm trọng số/chi phí.
* **Đồ thị đơn (simple graph):** không có nhiều cung/cạnh giữa cùng hai đỉnh, không có vòng tự (loop).
* **Đa đồ thị (multigraph):** cho phép nhiều cạnh giữa cùng hai đỉnh.
* **Đồ thị có vòng (pseudograph):** cho phép vòng tự (edge from a vertex to itself).
* **Đồ thị liên thông / không liên thông, cây, đồ thị đầy đủ (complete), bipartite, v.v.**

**Ví dụ minh họa (vô hướng, 4 đỉnh):**  
Đỉnh: {1,2,3,4}, Cạnh: { {1,2}, {1,3}, {3,4} }.



**Ma trận kề (adjacency matrix)** của đồ thị có nnn đỉnh là ma trận A=[aij]A=[a\_{ij}]A=[aij​] kích thước n×nn\times nn×n với:

* **Đồ thị vô hướng, không trọng số (0/1):**

A black background with white text

AI-generated content may be incorrect.

* **Đồ thị có hướng:**

A black and white text

AI-generated content may be incorrect.

Ma trận có thể **không đối xứng**.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black and white text on a black background

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

**8) Mối quan hệ giữa ma trận kề và đỉnh cô lập (isolated vertex)**

* **Đỉnh cô lập** là đỉnh không có cạnh nào nối tới hoặc ra khỏi nó.
* Trong ma trận kề (cả hướng và vô hướng), **đỉnh i cô lập ⇔ tất cả phần tử ở hàng i và cột i đều bằng 0**.
  + Với đồ thị vô hướng, hàng i = cột i, nên chỉ cần hàng i toàn 0.
  + Bậc (degree) = tổng hàng i = 0.

**Ví dụ:** nếu hàng và cột thứ 3 toàn 0 thì đỉnh 3 là đỉnh cô lập.

**9) So sánh giống nhau và khác nhau giữa đồ thị vô hướng và có hướng**

**Giống nhau:**

* Cả hai đều gồm tập đỉnh VVV và tập cạnh/cung EEE.
* Cả hai có thể biểu diễn bằng **ma trận kề** và **danh sách kề**.
* Nhiều khái niệm như đường đi, chu trình, thành phần liên thông (khái niệm khác nhau với có hướng) vẫn tương tự.

**Khác nhau (điểm chính):**

* **Hướng cạnh:** vô hướng: cạnh không có chiều; có hướng: cung có chiều → thứ tự các đỉnh quan trọng.
* **Ma trận kề:** vô hướng → ma trận **đối xứng**; có hướng → **không đối xứng** nói chung.
* **Độ (degree):** vô hướng có một bậc (degree); có hướng có **bậc vào** và **bậc ra**.
* **Liên thông:** vô hướng: “connected” (một loại). Có hướng: phân biệt **weakly connected** (bỏ hướng) và **strongly connected** (có đường đi theo hướng giữa mọi cặp).
* **Thuật toán/ứng dụng:** nhiều bài toán cho đồ thị có hướng (topological sort, phân lớp chu trình, mạng chảy) khác với đồ thị vô hướng (spanning tree, matching, v.v.).
* **Kiểm tra cạnh tồn tại:** cả hai cùng O(1) với ma trận; nhưng với danh sách kề, vô hướng lưu cả hai chiều hoặc chỉ lưu một chiều tùy cách.

**10) Vai trò và ứng dụng của đồ thị trong thực tế (vài ví dụ minh họa)**

Đồ thị là mô hình rất phổ biến; ứng dụng trải rộng nhiều lĩnh vực:

1. **Mạng xã hội:** đỉnh = người, cạnh = quan hệ bạn bè / follow. Dùng để tìm “influencer” (đỉnh có degree lớn), cộng đồng (community detection).
   * Ma trận kề cho biết ai kết nối với ai; phân tích bậc (degree) cho trung tâm mạng.
2. **Mạng giao thông / bản đồ:** đỉnh = giao lộ, cạnh = đường (trọng số = thời gian/chi phí).
   * Giải bài toán đường đi ngắn nhất (Dijkstra), luồng tối ưu (max-flow) cho lưu lượng.
3. **Mạng máy tính / routing:** nút = router, cạnh = link; xác định đường truyền, phân tích độ tin cậy, redundancy.
4. **Phân tích phụ thuộc / build (topological sort):** biểu diễn quan hệ phụ thuộc (task A phải xong trước B) bằng đồ thị có hướng, dùng topological sort để sắp xếp thực hiện.
5. **Sinh học:** mạng tương tác protein, mạng gen.
6. **Hóa học:** phân tử coi như đồ thị (đỉnh = nguyên tử, cạnh = liên kết) để phân tích cấu trúc.
7. **Search & ranking (PageRank):** biểu diễn web bằng đồ thị có hướng (link từ trang i → j). PageRank dùng ma trận chuyển tiếp dựa trên ma trận kề.
8. **Xử lý hình ảnh, tách vùng (segmentation):** sử dụng đồ thị để tách vùng ảnh.