 **Giả sử bạn là nhân viên giao hàng, công ty tại TP.HCM, xếp giao nhiệm vụ giao hàng ở một số tỉnh như sơ đồ trong slide. Để tiết kiệm thời gian, chi phí đi lại, nên xếp yêu cầu giao hàng đi qua mỗi tỉnh 1 lần, xong việc thì quay lại công ty => tổng đường đi ngắn nhất**

- Lê Trịnh Thanh Thúy (N10): HCM – NT – Vinh – HN - ĐN

- Hoàng Quang Vũ (N3): HCM -> NT -> HN -> Vinh -> ĐN -> HCM

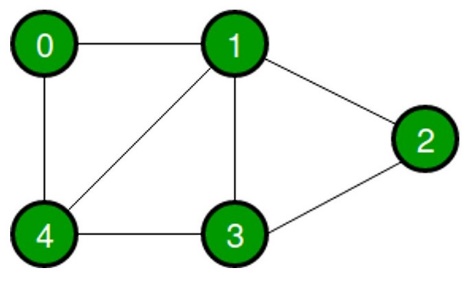
- Nguyễn Trung Tuấn (N7): HCM -> NT -> HN -> Vinh -> ĐN -> HCM

*Trả lời:* Bài này có nhiều hơn 1 đáp án. Mình chọn đường: HCM – NT – V – HN – ĐN – HCM với tổng đường đi 3300km

**Đồ thị là gì?**

- Lê Văn Trí (N2): Đồ thị là tập các đối tượng gồm có đỉnh được nối với nhau bằng cạnh. Tùy cạnh, nếu có hướng, có mũi tên, gọi là đồ thị có hướng; mũi tên không hướng là đồ thị vô hướng

*Trả lời:* Đồ thị là một tập cấu trúc dữ liệu được ký hiệu G(V, E), trong đó: V là một tập hợp hữu hạn các đỉnh còn được gọi là các đỉnh. E: là một tập hữu hạn các cặp có thứ tự có dạng (u, v) được gọi là cạnh. Nói là tập các thứ tự vì, cạnh (u,v) khác cạnh (v,u) ở trong đồ thị có hướng. Ký hiệu u, v cho ta biết tồn tại đỉnh đi từ đỉnh u tới đỉnh v

 **Đồ thị này được học ở môn gì năm nhất?**

- Cát Văn Tài (N4): Cấu trúc rời rạc

Trả lời: Toán rời rạc

**Nêu vài ứng dụng thực tế của dạng đồ thị này**

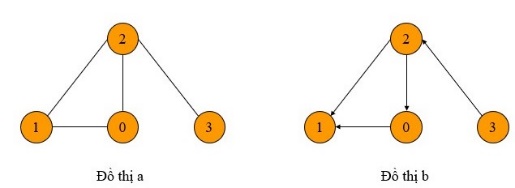
- Cát Văn Tài (N4): google maps(tính quãng đường giữa 2 nơi)

*Trả lời:* ví dụ như mạng lưới internet; mạng điện; google maps. Mạng xã hội (mỗi người sẽ là 1 nút chứa thông tin của mình, là liên kết giữa mỗi người sẽ là các cạnh

**Đồ thị vô hướng. Đồ thị có hướng?**

- Hoàng Quang Vũ (N3): Đồ thị vô hướng là đồ thị có G(V,E) trong đó: V là tập các đỉnh hoặc nút, E là các cặp không thứ tự chứa các đỉnh phân biệt được gọi là cạnh; 2 đỉnh 1 cạnh gọi là đỉnh đầu cuối cạnh đó. Đồ thị có hướng có 1 cạnh xy = E, thì cạnh này có hướng từ x đến y với x là đầu, y là cuối

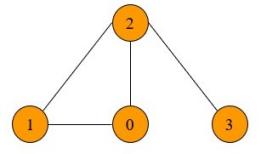
*Trả lời:* Đồ thị vô hướng: có các cạnh không định hướng, mỗi cạnh là 1 mối quan hệ 2 chiều (u,v) ≡ (v, u). Đồ thị có hướng: Các cạnh của nó được định hướng, cạnh gọi là cung để phân biệt với các cạnh của đồ thị vô hướng, mỗi cạnh là một mối quan hệ 1 chiều (u, v) ≠ (v, u).

 **Đồ thị nào vô hướng, đồ thị có hướng?**

**-** Hoàng Quang Vũ(N3): a vô hướng, b có hướng

- Cát Văn Tài (N4): a vô hướng, b có hướng

Trả lời: a vô hướng (không có mũi tên định hướng), b có hướng(cạnh là mũi tên định hướng)

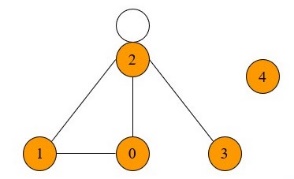


**Lấy ví dụ về cạnh kề, đỉnh kề**

- Nguyễn Khả Tiến (N7): Cạnh kề: 1:2 và 2:3; Đỉnh kề: 1:0 và 2:0

- Lê Trịnh Thanh Thúy (N10): Cạnh kề: 1:2 và 3:2 chung đỉnh 2; Đỉnh kề: 2 và 3 nối với nhau bởi (2,3)

*Trả lời:* 2 bạn trên đúng. Cạnh kề, ví dụ (0,1) (1,2) 2 cạnh này kề nhau chung đỉnh 1. Đỉnh kề, ví dụ đỉnh 1 vs 2 có cạnh liên thuộc (1,2); 2 vs 3



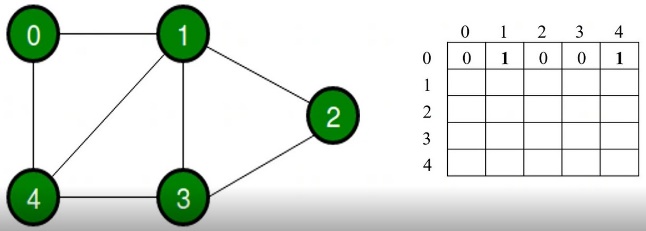
**Tìm bậc của các đỉnh**

- Nguyễn Khả Tiến (N7): 0,1,2: bậc 2; 3: bậc1 và 4 bậc 0

- Lê Trịnh Thanh Thúy(N10): 2,1 bậc 2; 0,3 bậc 1; 4 bậc 0;

- Võ Khánh An (N8): 0,1,2 bậc 2; 3 bậc 1; 4 bậc 0

*Trả lời:* Đỉnh 0 có 2 cạnh liên thuộc (0,1) (0,2) => đỉnh 0 là bậc 2; tương tự đỉnh 1 bậc 2; đỉnh 2 có 1 cái khuyên (khuyên được tính là 2 bậc, vì nó đi được từ trái -> phải, phải -> trái) => đỉnh 2 bậc 5; đỉnh 3 bậc 1; đỉnh 4 bậc 0

 **Hoàn thành ma trận kề**

- Bùi Trí Dũng (N8): 10110-01010-01101-11010

*Trả lời:* Dũng đọc sai vị trí (1,4) sửa lại => 1 (10111-01010-01101-11010)

**Tại sao việc loại bỏ hoặc thêm 1 cạnh chỉ mất O(1) thời gian**

- Bùi Trí Dũng (N8): vertex[a][b] = 0 hoặc = 1, gán luôn nên O(1)

*Trả lời:* vì, trong ma trận, khi ta muốn bỏ đi 1 cạnh ta chỉ cần biến giá 1 thành 0, them 1 cạnh thì đổi 0 thành 1

**Mình đã học cấu trúc dữ liệu nào mà có duyệt theo chiều rộng và duyệt theo chiều sâu**

- Bùi Trí Dũng (N8): 1 cái dùng stack, 1 cái dùng hàng đợi

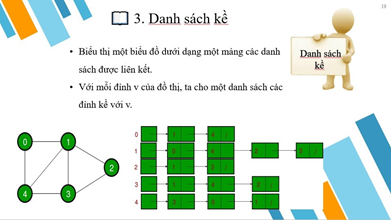
*Trả lời:* thấy ở trong cây nhị phân, ở trong đồ thị cũng duyệt tương tự vậy, tuy nhiên trong đồ thị đòi hỏi phức tạp hơn

**Nghe thuật toán Prim’s, các bạn có nhớ mình học ở đâu không, thuật toán này dùng để làm gì**

- Cát Văn Tài (N4): Học trong toán rời rạc, tìm cây cung nhỏ nhất

*Trả lời:* dùng trong toán rời rạc, tìm cây cung nhỏ nhất

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* Đây là định nghĩa nguyên thủy của biểu diễn đồ thị bằng dnah sách kề, trong cài đặt cụ thể người ta có cài đặt như vậy không? Đây là điều phải lưu ý.

Trong cài đặt cụ thể, thay vì người ta dùng 1 list để biểu diễn cho 1 danh sách kề của 1 đỉnh, trong C++ thì list rất chậm, nên người ta sẽ cài mỗi list là 1 mảng động (C++: vector; python: ?)

\* Cần trả lời câu hỏi What Why. Phần trả lời What khá tốt, How rất tốt, câu hỏi Why:

- Vì sao người ta dùng mô hình đồ thị

- Lưu trữ danh sách kề thì tốt xấu ở chỗ nào? Vì sao tốt/ xấu?

- BFS và DFS được xây dựng dựa trên phương pháp thiết kế giải thuật nào trong các phương pháp đã học? Tại sao người ta làm như vậy? (Nhóm chỉ mới tường thuật người ta làm như vậy và đặt kết quả như vậy)

\* Phải nhìn ra được cách người ta thiết kế nên những thuật toán chúng ta đang tìm hiểu, để khi gặp bài toán mới, chúng ta có thể thiết kế giải thuật mà trước đó chưa biết gì về giải thuật đó (Mục tiêu môn này)

\* Bổ sung slide 2 điều:

- Tại sao như vậy

- Người ta dùng cái này vào đâu, áp dụng như thế nào (Ví dụ BFS, những ứng dụng nào đã dùng BFS). Phần vận dụng đó, nó phải được thể hiện ở trong nghành đang học