2.4.2 Graph and Graph Representation (Đồ thị và Biểu diễn của đồ thị)

Created by TUNG DUC NGUYEN tung2.nguyen, last modified about 5 hours ago

1. Các cấu trúc dữ liệu biểu diễn danh sách

Danh sách là tập sắp thứ tự các phần tử cùng một kiểu. Đối với danh sách, người ta có một số thao tác:

- Tìm một phần tử trong danh sách
- Chèn một phần tử vào danh sách
- Xóa một phần tử khỏi danh sách
- Sắp xếp các phần tử theo một trật tự nào đó
- ...

Có rất nhiều cách biểu diễn danh sách trong máy tính, ta đi xem tưng cách một.

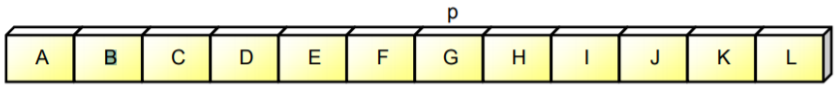
1.1 Biển diễn danh sách bằng mảng một chiều

Khi cài đặt danh sách bằng một mảng, thì có một biến nguyên n lưu số phần tử hiện có trong danh sách. Nếu mảng được đánh dấu từ 1 thì các phần tử trong danh sách được cất giữ trong mảng bằng các phần tử được đánh dấu từ 1 đến n.

Ta có các thao tác với mảng như sau:

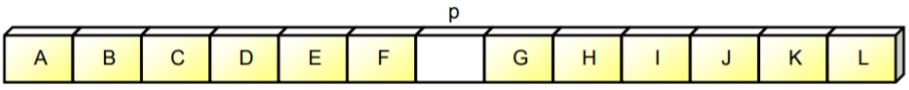
Chèn phần tử vào mảng:

Mảng ban đầu:

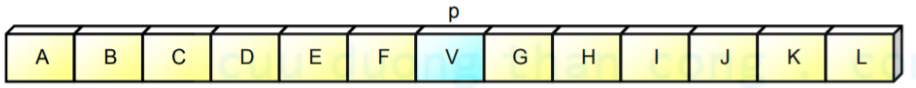


Ta muốn chèn một phần tử V vào mảng tại vị trí p ta phải:

Dồn tất cả các phần tử từ vị trí p tới vị trí n về sau một vị trí:



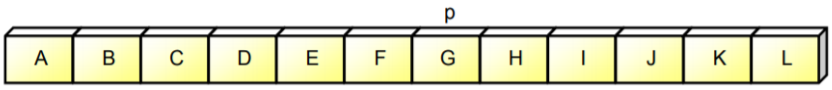
Đặt giá trị V vào vị trí p:



Tăng n lên 1

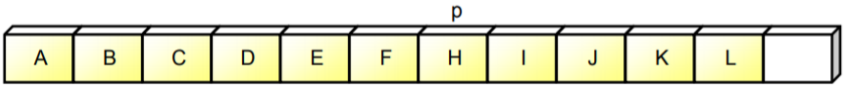
Xóa phần tử khỏi mảng:

Mảng ban đầu:

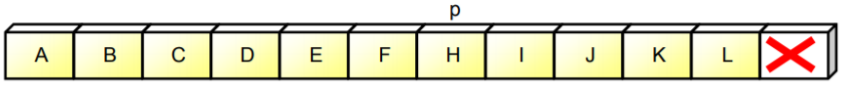


Muốn xóa phần tử p của mảng mà vẫn giữ nguyên thứ tự của các phần tử còn lại, ta phải:

Dồn tất cả phần tử từ vị trí p+1 tới vị trí n lên một vị trí:



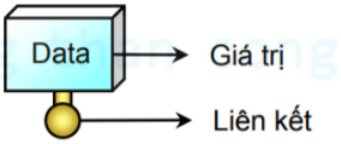
Giảm n đi 1:



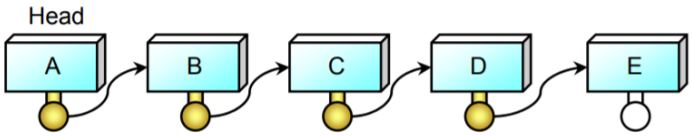
Nếu muốn xóa phần tử p mà không cần duy trì thứ tự của các phần tử khác, ta chỉ cần đảo giá trị của phần tử cần xóa bằng phần tử cuối cùng rồi giảm số phần tử của mảng n đi 1.

1.2 Biển diễn danh sách bằng danh sách nối đơn

Danh sách nối đơn gồm các nút được nối với nhau theo một chiều. Mỗi nút là một bản ghi (record) bao gồm hai trường: Giá trị và Liên kết tới nút kế tiếp (con trỏ):

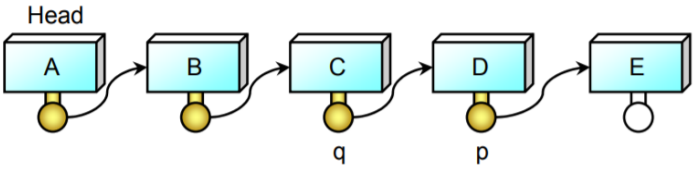


Nút đầu tiên trong danh sách được gọi là chốt của danh sách đơn (Head). Để duyệt danh sách đơn, chúng ta bắt đầu từ head, dựa vào liên kết để đi đến nút kế tiếp, đến khi gặp giá trị đặc biệt hoặc duyệt qua nút cuối thì dừng lại.



Chèn phần từ vào danh sách đơn:

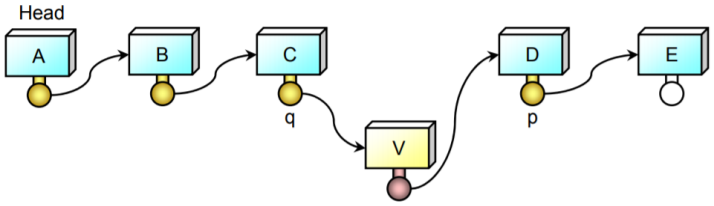
Danh sách ban đầu ta có:



Muốn chèn thêm một nút chứa giá trị V vào vị trí nút p, ta phải:

- Tạo ra một nut mới NewNode chứa giá trị V:

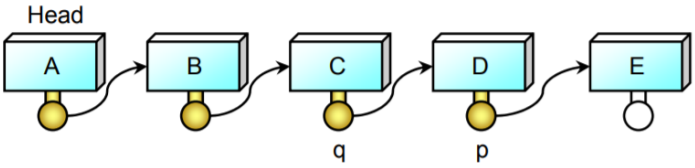
- Tìm nút q đứng trước nút p trong danh sách:
  - Nếu tồn tại nút q, cho q liên kết tới NewNode, NewNode liên kết đến p:



- Nếu không tồn tại nút q → p là Head, chỉnh lại NewNode liên kết đến Head (cũ) và đặt lại Head là NewNode

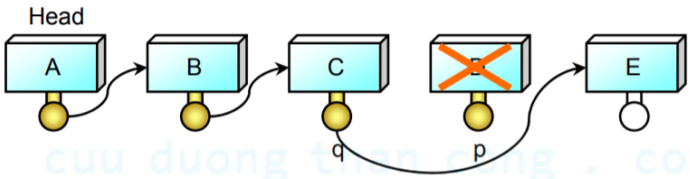
Xóa phần từ khỏi danh sách đơn:

Danh sách ban đầu ta có:



Muốn xóa nút p khỏi danh sách đơn, ta phải:

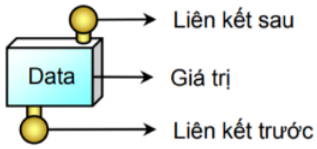
- Tìm nút q đứng trước nút p trong danh sách:
  - Nếu tồn tại nút q, cho q liên kết tới nút liên sau p



- Nếu không tồn tại nút q → p đang là head, ta chỉ cần đặt head bằng nút đứng kế tiếp head (cũ) trong danh sách.

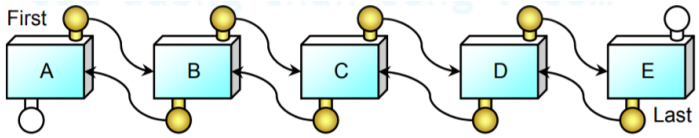
1.3 Biểu diễn danh sách bằng danh sách nối kép

Danh sách nối kép gồm các nút được nối với nhau theo hai chiều. Một nút là một bản ghi gồm 3 trường: Giá trị, Liên kết đến nút kết tiếp (Next) và liên kết đến nút liền trước(Pre):



Khác với danh sách đơn, danh sách kép có hai chốt: Nút đầu tiên của danh sách được gọi là First, Nút cuối cùng trong danh sách được gọi là Last.

Để duyệt danh sách kép, chúng ta hoặc bắt đầu từ First, hoặc bắt đầu từ Last.

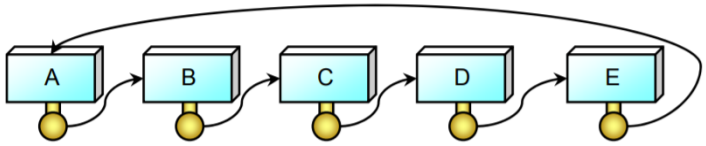


Việc chèn xóa cũng tương tự như danh sách đơn.

1.4 Biển diễn danh sách bằng danh sách nối vòng một hướng

Trong danh sách đơn, phần tử cuối cùng trong danh sách có trường gán một giá trị đặc biệt, thường dùng nhất là null.

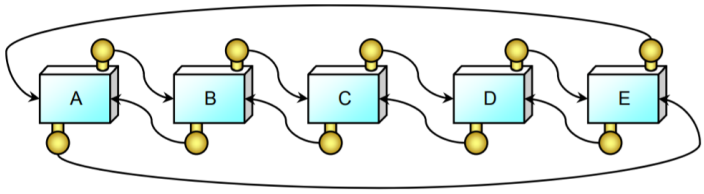
Nếu ta cho phần tử cuối trỏ về Head, thì ta có một danh sách mới được gọi là danh sách vòng một hướng:



Việc chèn/xóa cũng tương tự như danh sách đơn (chỉ khác là không quan tâm đến việc xử lý tại HEAD).

1.4 Biển diễn danh sách bằng danh sách nối vòng hai hướng

Nếu danh sách nối kép ta thực hiện việc Pre của nút First trỏ đến nút Last còn Next của nút Last trỏ đến nút First, ta có danh sách nối vòng hai hướng:



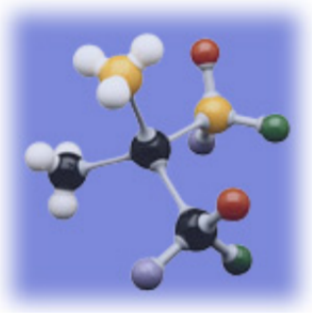
2. Đồ thị

2.1 Định nghĩa đồ thị

Đồ thị là một cấu trúc rời rạc gồm các đỉnh và các cạnh nối các đỉnh đó. Được mô tả hình thức: **G = (V,E)**.

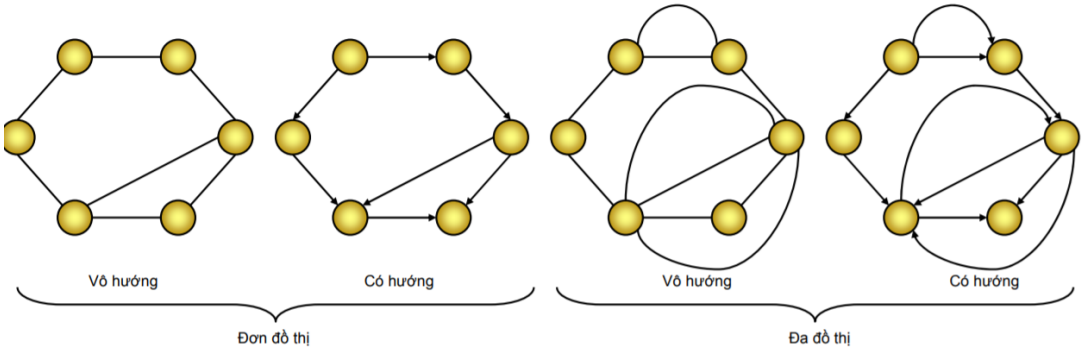
Trong đó, V gọi là tập các đỉnh (vertices) và E là tập các cạnh (Edges).

Một số ví dụ về đồ thị (giao thông, mạng máy tính, cấu trúc phân tử):



Ta có các loại đồ thị sau:

- **Đơn đồ thị**: nếu giữa hai đỉnh u,v của V có nhiều nhất một cạnh trong E nối từ u đến v.
- **Đa đồ thị**: nếu giữa hai đỉnh u,v của V có nhiều hơn một cạnh trong E nối từ u đến v.
- **Đồ thị vô hướng**: nếu các cạnh trong E là không định hướng
- **Đồ thị có hướng:** nếu các cạnh trong E là có định hướng. Trong đồ thị có hướng, các cạnh được gọi là các cung.



2.2 Các khái niệm

Đồ thị G = (V,E) là một cấu trúc rời rạc. Tức là các tập V và E là các tập hữu hạn và đếm được.

Cạnh liên thuộc (incident) , đỉnh kề (adjacent vertices), bậc (degree)

Đối với đồ thị vô hướng G = (V,E). Xét cạnh e € E, nếu e = (u,v) thì ta nói u và v là hai đỉnh **kề nhau (adjacent)** và cạnh e **liên thuộc (incident)** với đỉnh u và đỉnh v.

Với một đỉnh v trong đồ thị, bậc (degree) là số cạnh liên thuộc với v. Đối với đơn đồ thị, thì số cạch liên thuộc chỉnh là số đỉnh kề với v.

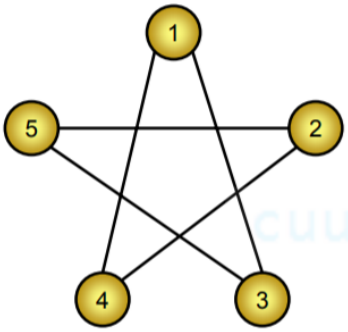
2.3 Biểu diễn đồ thị trên máy tính

2.3.1 Ma trận kề

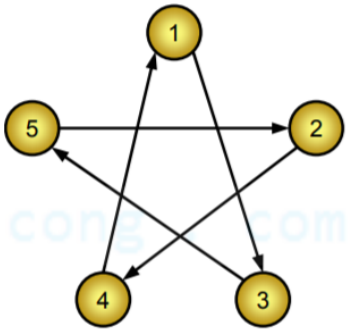
Giả sử một **đơn đồ thị** G = (V,E) có số đỉnh là n. Khi đó ta có thể biểu diễn đồ thị bằng một ma trận vuông A cấp n.

Với A[[i,j] = 1 nếu (i,j) € E và A[i,j] = 0 nếu (i,j) không nằm trong E.

Ví dụ:



$$A=\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$



$$A=\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Ta có nhận xét: Nếu đồ thị G là vô hướng, thì mà trận kề tương ứng là ma trận đối xứng A[i,j] = A[j,i].

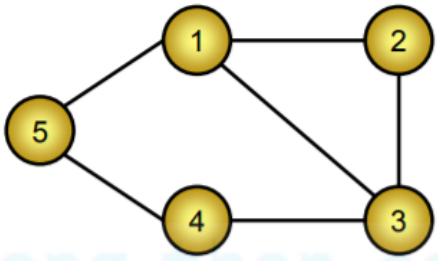
Với cách dùng ma trận kề để biểu diễn đồ thị, ưu điểm là: *Đơn giản, trực quan, dễ cài đặt trên máy tính.*

Tuy nhiên nhược điểm là: *Không gian bộ nhớ cần là n^2 bất kể số cạnh nhiều hay ít. Và khi duyệt đỉnh u, ta phải duyệt tất cả các đỉnh trong đồ thị → lãng phí bộ nhớ và thời gian.*

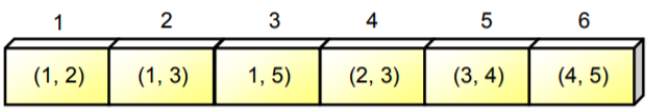
2.3.2 Danh sách cạnh

Trong trường hợp đồ thị có n đỉnh, m cạnh ta có thể biểu diễn đồ thị dưới dạng danh sách cạnh bằng cách liệt kê tất cả các cạnh đồ thị trong một danh sách, mỗi phần tử là một cặp (u,v) tương ứng là một cạnh của đồ thị. Danh sách có thể dùng mảng hoặc danh sách liên kết.

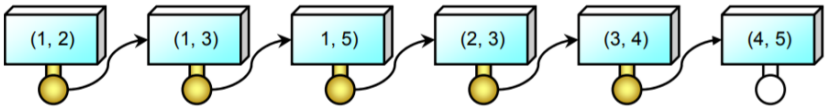
Ví dụ: Với đồ thị



Ta có thể dùng mảng:



Hoặc danh sách liên kết:



Ưu điểm của danh sách cạnh:

Trong trường hợp số cạnh tương đối nhỏ (m < 6n), thì biểu diễn bằng danh sách cạnh sẽ tiết kiệm được không gian lưu trữ.

Trong một số trường hợp mà thuật toán của chúng ta cần duyệt các cạnh, thì danh sách cạnh sẽ làm cho việc duyệt dễ dàng hơn.

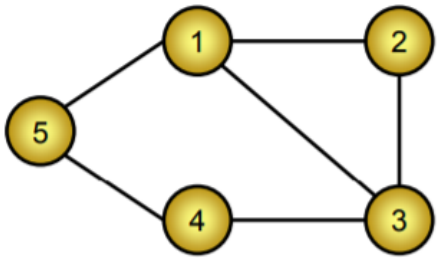
Nhược điểm:

Khi cần duyệt đỉnh kề với đỉnh v nào đó, ta cần duyệt toàn bộ các cạnh. Việc này có thể tốn thời gian.

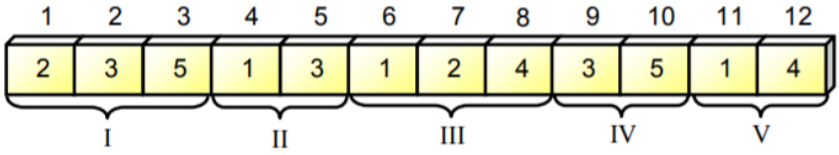
2.3.3 Danh sách kề

Để khắc phục nhược điểm của cả hai phương pháp danh sách cạnh và ma trận kề, người ta đề xuất việc biểu diễn đồ thị bằng danh sách kề. Trong cách biểu diễn này, mỗi đỉnh v của đồ thị có một danh sách các đỉnh kề với v.

Ví dụ với đồ thị:



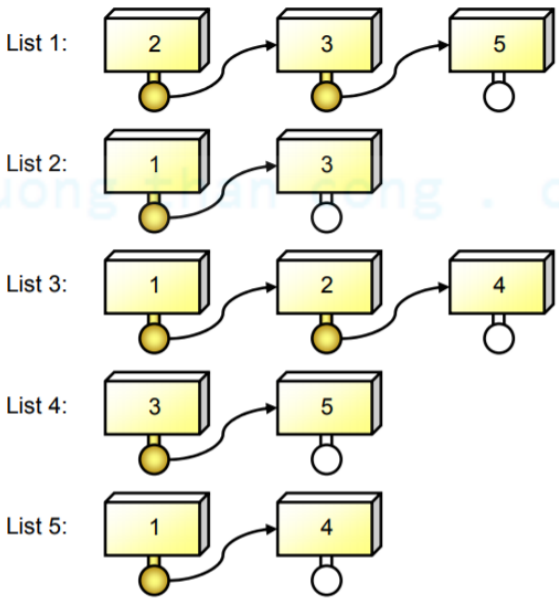
Ta có thể dùng mảng:



Và thêm một mảng phụ Head, trong đó Head[i] lưu các vị trí bắt đầu của của đoạn thứ i.

Như vậy, các đỉnh kề vời đỉnh u sẽ là các đỉnh từ A[Head[u]] đến A[Head[u + 1] -1].

Hoặc ta có thể dùng danh sách liên kết:



Trong đó mỗi đỉnh i là một danh sách các đỉnh kề với i. Head của danh sách của đỉnh thứ i là List[i].

Ưu điểm của việc dùng danh sách kề:

Duyệt tất cả các đỉnh kề của một đỉnh v cho trước dễ dàng và tiết kiệm bộ nhớ

Nhược điểm:

Việc kiểm tra (u.v) có phải là cạnh không thì sẽ mất thời gian hơn sử dụng ma trận kề.