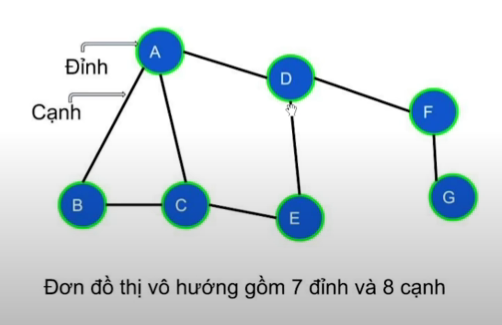
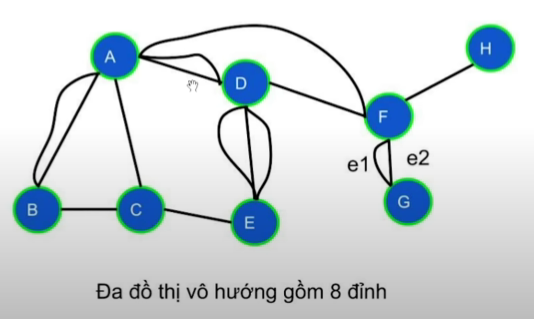
**ĐỒ THỊ**

**BUỔI 1:CÁC KHÁI NIỆM CƠ BẢN**

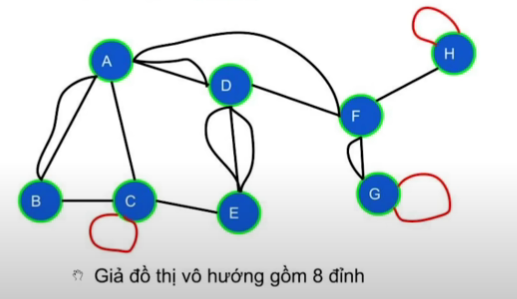
* Đơn đồ thị vô hướng: Đơn đồ thị vô hướng G = <V,E> , gồm V(Vertex) là các tập đỉnh, E (edge) là tập các cặp không có thứ tự bao gồm hai phần tử khác nhau của V gọi là các cạnh.



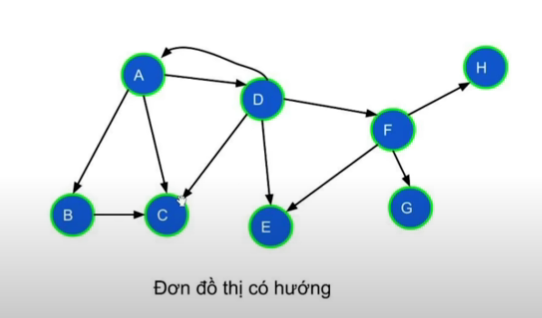
* Đơn đồ thị nghĩa là giữa hai đỉnh có nhiều nhất một cạnh và chỉ một cạnh mà thôi, giữa hai cái đỉnh của đồ thị bắt buộc phải khác nhau.
* Đa đồ thị nghĩa là giữa hai đỉnh có nhiều cạnh , giữa hai đỉnh của đồ thị bắt buộc phải khác nhau.
* Đa đồ thị vô hướng G = <V,E> gồm V là tập các đỉnh , E là tập các cạnh hay gọi là họ các cặp không có thứ tự gồm hai phần tử khác nhau của V . Hai cạnh e1,e2 được gọi là cạnh bội nếu chúng cùng tương ứng với một cặp đỉnh.



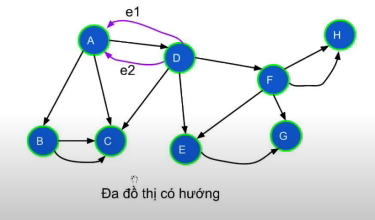
* Giả đồ thị vô hướng G = <V,E> , gồm V là tập các đỉnh, E là tập các cặp không có thứ tự gồm hai phần tử không nhất thiết phải khác nhau của V gọi là các cạnh.
* Cạnh e = (u,u) được gọi là khuyên.
* Cứ nhớ rằng giả đồ thị thì phải chứa khuyên.

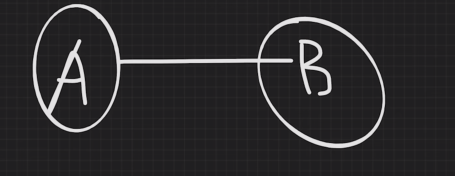


* Đơn đồ thị có hướng G = <V,E> gồm V là tập các đỉnh, E là tập các cặp có thứ tự gồm hai phần tử V gọi là các cung.



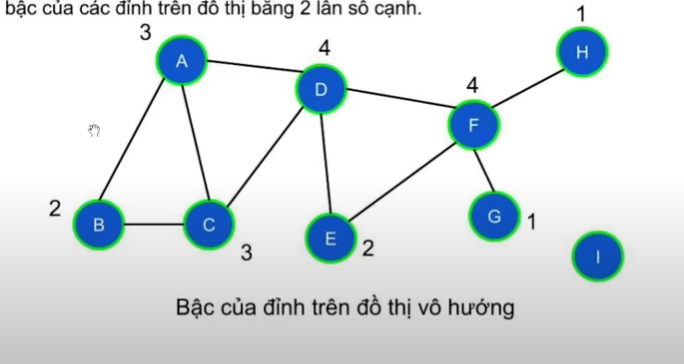
* Đa đồ thị có hướng G = <V,E> gổm V là tập các đỉnh, E là họ các cặp có thứ tự gồm hai phần tử V gọi là các cung.Hai cung e1,e2 tương ứng với cùng một cặp đỉnh được gọi là cung lặp.

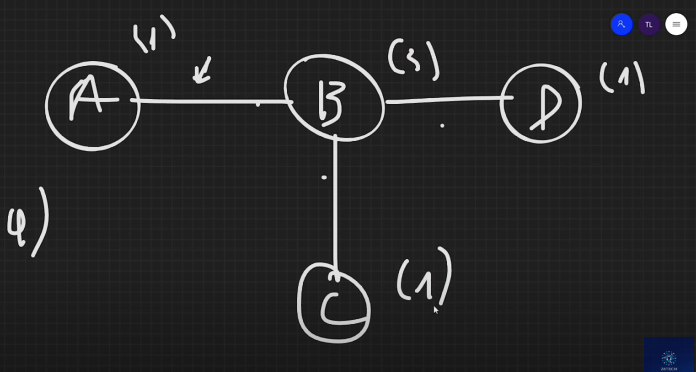




* A và B là hai đỉnh kề nhau . Hai đỉnh u,v được gọi là 2 đỉnh kề nếu tồn tại cạnh e =(u,v) là cạnh của đồ thị. Ví dụ AB =(A,B), AF = (A,F)
* Cạnh liên thuộc : Nếu cạnh e = (u,v) là cạnh của đồ thị thì cạnh e được gọi là cạnh liên thuộc với hai đỉnh u,v . Ví dụ AF là cạnh liên thuộc với hai đỉnh A,F.
* Bậc của đỉnh : Bậc của đỉnh u trên đồ thị vô hướng là số cạnh liên thuộc với đỉnh u , kí hiệu là deg(u)
* Đỉnh có bậc 0 được gọi là đỉnh cô lập . Ví dụ đỉnh I.
* Đỉnh có bậc 1 được gọi là đỉnh treo. Ví đụ đỉnh H, G
* Đồ thị G=<V,E> là đồ thị vô hướng có m cạnh, khi đó tổng bậc của các đỉnh trên đồ thị bằng 2 lần số cạnh.

TỔNG BẬC = 2 \* SỐ\_CẠNH





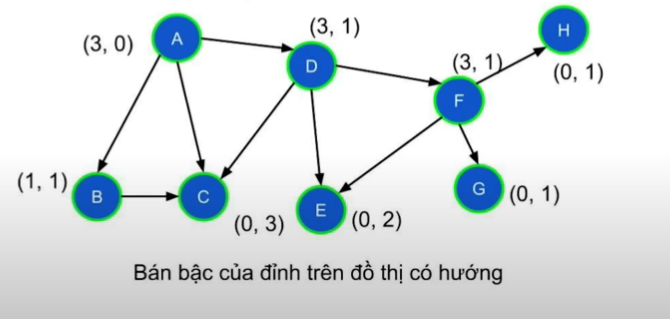
* Bán bậc vào của đỉnh trên đồ thị có hướng .
* Bán bậc ra của đỉnh u trên đồ thị có hướng là số cung của đồ thị đi ra khỏi đỉnh u , kí hiệu là deg+(u).
* Bán bậc vào của đỉnh u trên đồ thị là số cung của đồ thị đi vào đỉnh u , kí hiệu là deg-(u)

VÀO ÂM RA DƯƠNG

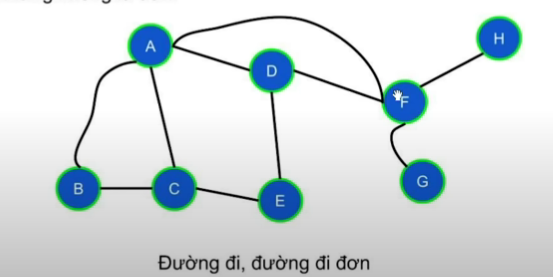
* Định lí : Trên đồ thị có hướng :

Tổng bán bậc ra = Tổng bán bậc vào = Số lượng cạnh.

(BÁN BẬC RA, BÁN BẬC VÀO )



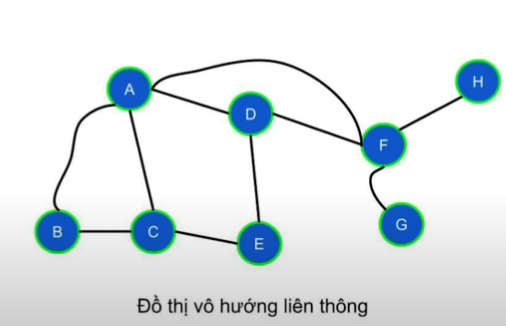
* Tóm lại bán bậc ra của một đỉnh thì xem có bao nhiêu cung đi ra từ đỉnh đó.(Mũi tên hướng ra )
* Bán bậc vào của một đỉnh thì xem có bao nhiêu đỉnh vào từ đỉnh đó. (Mũi tên hướng vào )
* Đường đi : Dãy các đỉnh ( Hoặc các dãy các cạnh ) trong đó 2 đỉnh liên tiếp có cạnh nối . Độ dài đường đi là số cạnh trên đường đi
* Đường đi đơn : Các đỉnh trên đường đi là phân biệt (A,B,C ,D) là 1 đường đi đơn , (A,B,C,A,D ) là đường đi nhưng không là đơn.



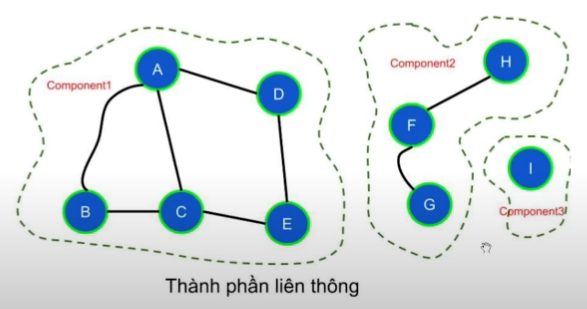
* Chu trình là đường đi gồm các cạnh phân biệt có đỉnh đầu trùng đỉnh cuối . Ví dụ (A,B,C,E,D,A) là một chu trình.
* Chu trình đơn : Ngoại trừ đỉnh đầu và đỉnh cuối trùng nhau, không còn 2 đỉnh nào giống nhau.

(D,A,B,C,A,F,D) là một chu trình nhưng không là đơn.

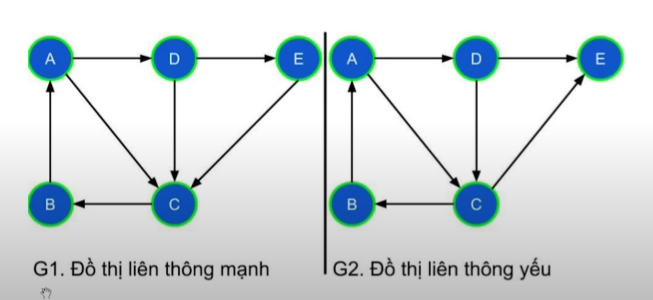
* Chu trình thì phải có sự khép kín.
* Liên thông : Đồ thị vô hướng được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kì của đô thị.



* Một đồ thị không liên thông thì nó sẽ phân rã thành các thành phần liên thông . Như vậy đồ thị vô hướng liên thông nếu nó có số TPLT là 1 . Ví dụ như đồ thị dưới đây có 3 thành phần liên thông ,chú ý 1 đỉnh cô lập cũng là 1 TPLT.



* Liên thông mạnh , yếu của đồ thị có hướng.
* Liên thông mạnh là đồ thị có hướng nếu giữa hai đỉnh bất kì u,v luôn có đường đi từ u tới v (Hình G1).
* Thằng nào cũng có mũi tên vào ra thì là liên thông mạnh.
* Liên thông yếu : Đồ thị có hướng gọi là liên thông yếu nếu đồ thị vô hướng tương ứng của nó liên thông (Hình G2).



============================================================

**NGÀY 2: BIỂU DIỄN ĐỒ THỊ TRÊN MÁY TÍNH**

* Ma trận kề : Với đồ thị vô hướng, ma trận kề của đồ thị có n đỉnh là ma trận vuông cỡ n \* n có các phần tử là 0 hoặc 1.

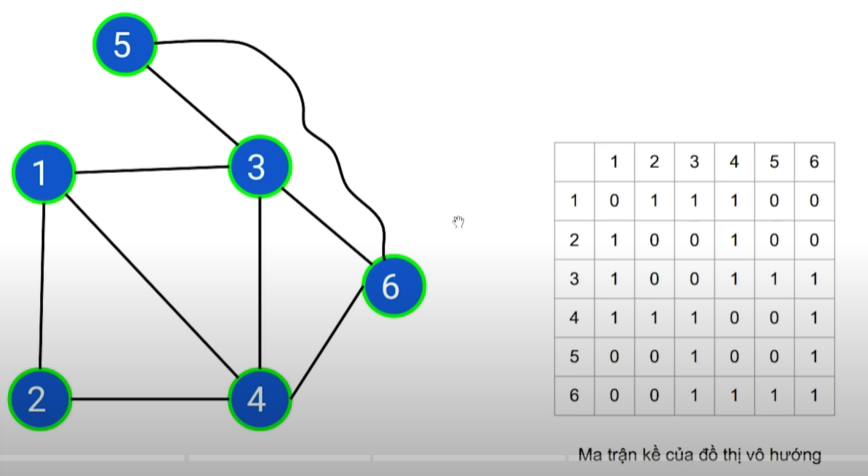
A = {aij , aij =1 nếu cạnh (i,j) là một cạnh của đồ thị , aij = 0 nếu cạnh (i,j) không là một cạnh của đồ thị }. Ví dụ để biểu diễn đồ thị dưới đây, cần ma trận kề có cỡ 6 \* 6.

* Tính chất : Ma trận kề là :

+ ma trận đối xứng.

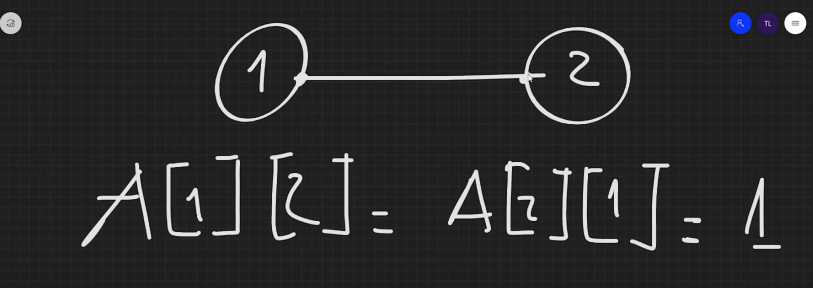
+ tổng các phần tử trên ma trận bằng 2 lần số cạnh.

+ tổng các phần tử trên hàng hoặc cột thứ u là bậc của đỉnh u.

****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **0** | **1** | 1 | **1** | **0** | **0** |
| **2** | **1** | **0** | **0** |  |  |  |
| **3** |  |  |  |  |  |  |
| **5** |  |  |  |  |  |  |
| **5** |  |  |  |  |  |  |
| **6** |  |  |  |  |  |  |

* Từ 1 đến 1 là 0 ,từ 1 -> 2 có cạnh 12 nên là 1.



* Ma trận kề với đồ thị có hướng ma trận kề của đồ thị có n đỉnh là ma trận vuông cỡ n \* n có các phần tử là 0 hoặc 1
* A = {aij , aij =1 nếu cạnh (i,j) là một cạnh của đồ thị ,

aij = 0 nếu cạnh (i,j) không là một cạnh của đồ thị}

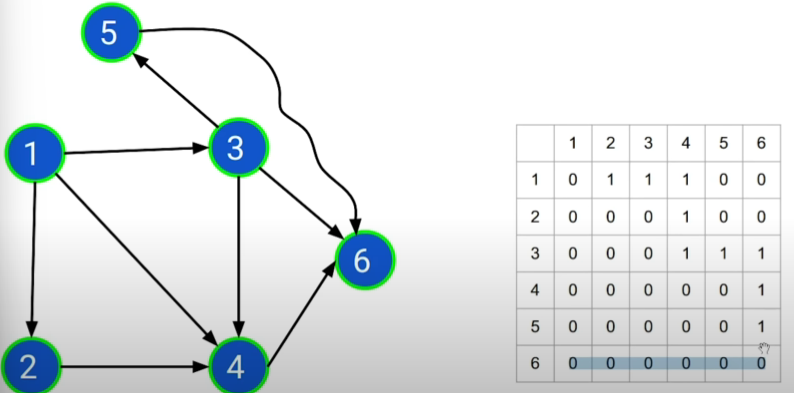
* Ví dụ để biểu diễn đồ thị dưới đây, cần ma trận kề có cỡ 6 \*6
* Tính chất :

+ Có thể không đối xứng

+ Tổng các phần tử của ma trận = số cạnh

+ Tổng các phần tử trên hàng thứ u là bán bậc ra của đỉnh u.

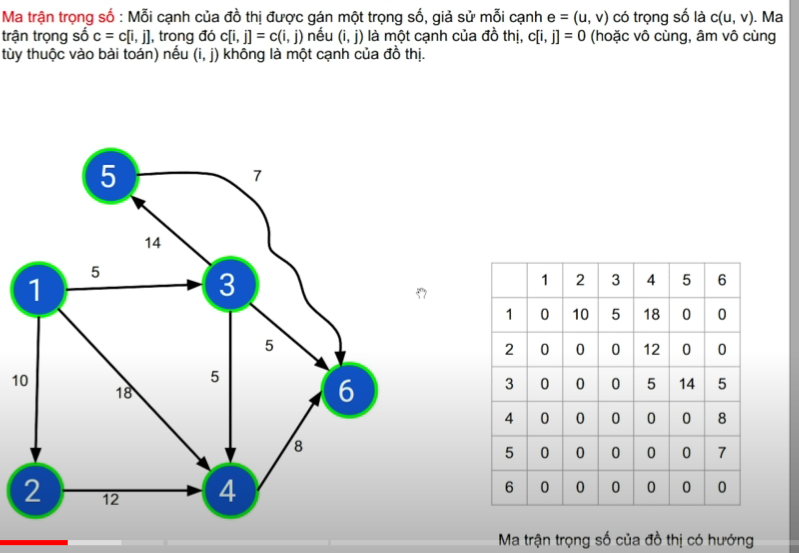
+ Tổng các phần tử trên cột thứ u là bán bậc vào của đỉnh u .



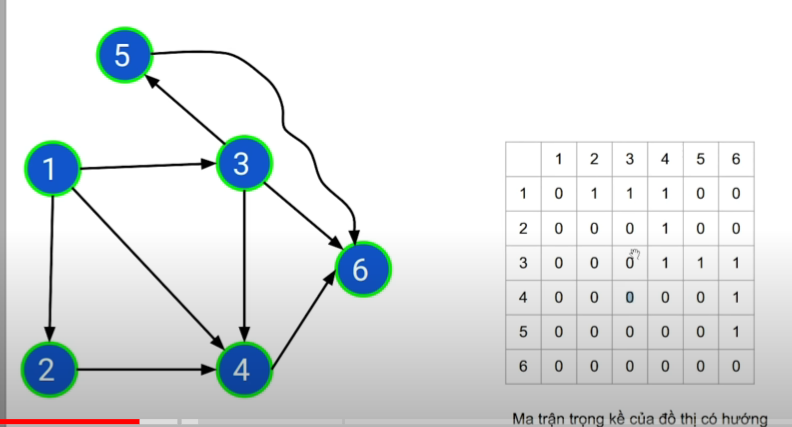
* Bán bậc ra (Số mũi tên chĩa ra) là TỔNG HÀNG . Ví dụ bán bậc ra của 1 là 3
* Bán bậc vào (Số mũi tên chĩa , bắn về ) là TỔNG CỘT. Ví dụ bán bậc vào của 1 là 0.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **0** | **1** | **1** | **1** | **0** | **0** |
| **2** | **0** |  |  |  |  |  |
| **3** | **0** |  |  |  |  |  |
| **5** | **0** |  |  |  |  |  |
| **5** | **0** |  |  |  |  |  |
| **6** | **0** |  |  |  |  |  |

* Với đồ thị có hướng thì ta phải đi theo chiều của mũi tên chứ không có kiểu thích đi như nào thì đi như đồ thị vô hướng.
* Ma trận trọng số thì nếu mà cạnh có trọng số thì lúc này ta sẽ thay số 1 biểu diễn thành cái trọng số đó là được.



* Ma trận kề : Ưu điểm là đơn giản, dễ cài đặt, dễ dàng kiểm tra 2 đỉnh có kề nhau hay không trong O(1) bằng cách kiểm tra giá trị của a[i][j].
* Nhược điểm :Tốn bộ nhớ nên không thể biểu diễn được đồ thị với số đỉnh lớn.

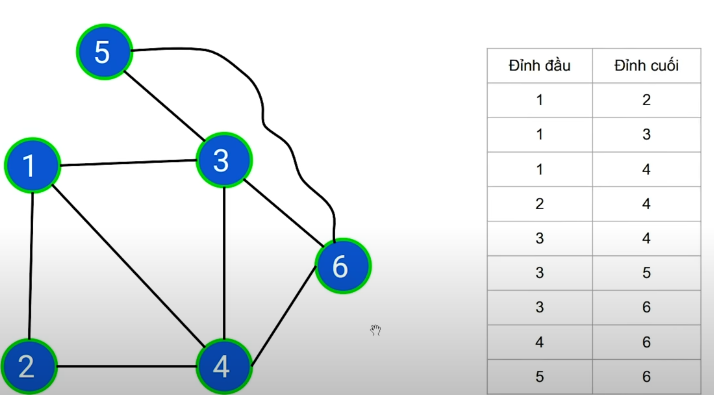


* Biểu diễn theo danh sách cạnh
* Thường được biểu diễn khi đồ thị thưa ( Số lượng cạnh <= 6 lần số đỉnh )

+ Đối với đồ thị vô hướng, nếu tồn tại cạnh giữa 2 đỉnh u và v. Ch

ỉ cần liệt kê cạnh (u,v) mà không cần liệt kê cạnh (v,u). Thường chọn u < v (Ví dụ như chọn 12 chứ không chọn 21 ).

+ Đối với đồ thị có hướng, mỗi cạnh là bộ có tính đến thứ tự của các đỉnh.

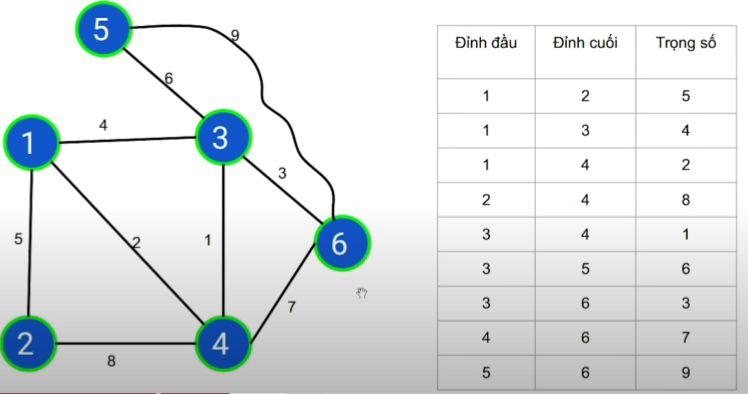


* Để lưu danh sách cạnh thì chúng ta có thể dùng pair.
* Trong trường hợp đồ thị có trọng số, mỗi cạnh sẽ có thêm trọng số đi kèm với đỉnh đầu và đỉnh cuối . Trong trường hợp danh sách không có trọng số có thể dùng pair<int,int> để biểu diễn mỗi cạnh , với cạnh có trọng số có thể dùng tuple hoặc định nghĩa một struct lưu thông tin các cạnh như sau:

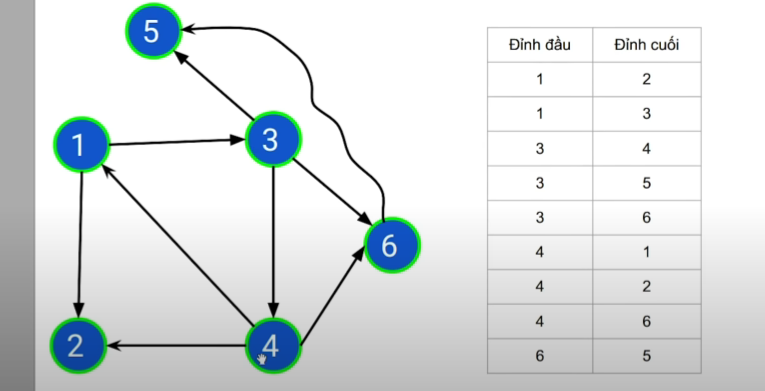
struct edge{

int dau,cuoi;

};



* Trong trường hợp có hướng thì ta chú ý tới hướng của cạnh. Với đồ thị có hướng có trọng số ta làm tương tự như với đồ thị vô hướng có trọng số.

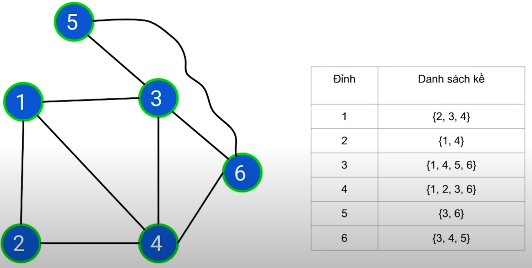


* Danh sách cạnh : Ưu điểm là tiết kiệm được bộ nhớ nếu đồ thị thưa , thuận lợi cho các bài toán chỉ liên quan đến cạnh của đồ thị.
* Nhược điểm : Khi cần duyệt các đỉnh kề với đỉnh nào đó, bắt buộc phải duyệt tất cả các cạnh dẫn tới chi phí tính toán lớn.

**Biểu diễn bằng danh sách kề**

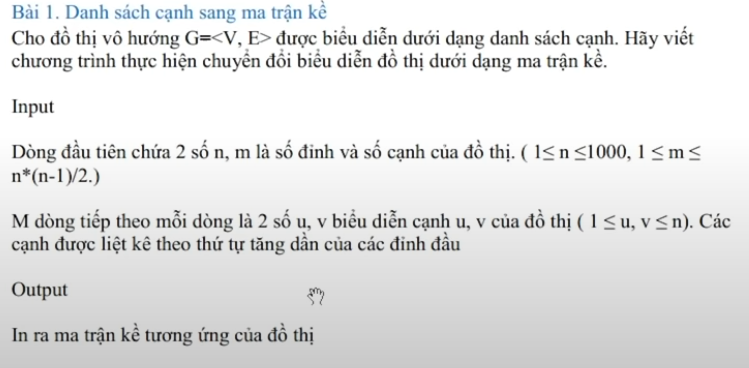
* Đối với mỗi đỉnh u của đồ thị, ta lưu trữ danh sách kề với đỉnh u . Trong C++, để lưu trữ danh sách kề của 1 đỉnh, ta dùng một vector . Khi đó để lưu trữ toàn bộ danh sách kề các đỉnh thì ta dùng một mảng các vector.
* Ví dụ : vector <int> adj[1001];
* Hoặc dùng một vector các vector :

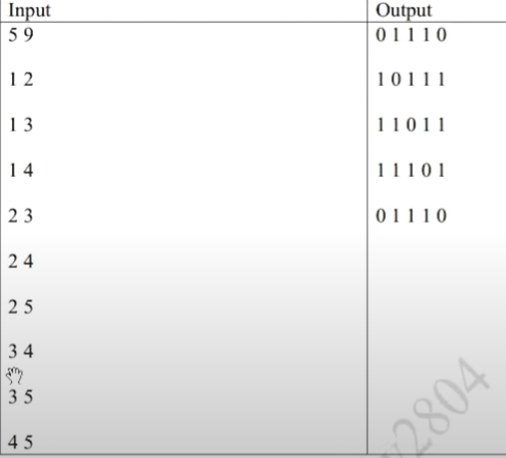
vector<vector<int,>>adj;

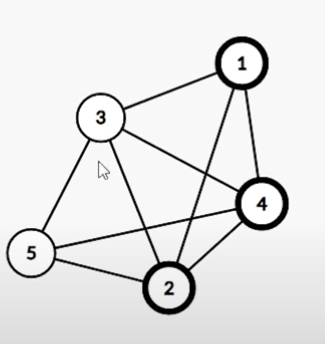


* Tiếp theo chúng ta phải biết chuyển đổi từ thằng này sang thằng kia và ngược lại.

|  |  |  |
| --- | --- | --- |
| MA TRẬN KỀ | DANH SÁCH CẠNH |  |
| MA TRẬN KỀ | DANH SÁCH KỀ |  |
| DANH SÁCH CẠNH | MA TRẬN KỀ |  |
| DANH SÁCH CẠNH | DANH SÁCH KỀ |  |
| DANH SÁCH KỀ | MA TRẬN KỀ |  |
| DANH SÁCH KỀ | DANH SÁCH CẠNH |  |







* Nó cho tất cả 5 đỉnh -> Ma trận kề của chúng ta có kích thước là 5 \* 5 . Thấy có cặp đỉnh nào thì sẽ đánh dấu là 1 tại cái phần tử của ma trận với chỉ số là cặp đỉnh của đỉnh ấy.

-------------

* Bản chất của ma trận kề là kiểm tra xem một đỉnh có kề với các đỉnh còn lại không.

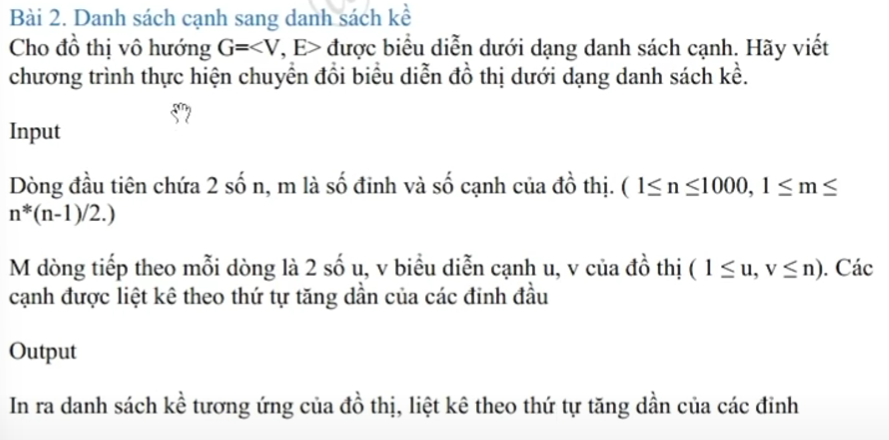
+ Nếu kề thì là 1 (Hai đỉnh mà kề nhau tức tồn tại cạnh nối hai đỉnh)

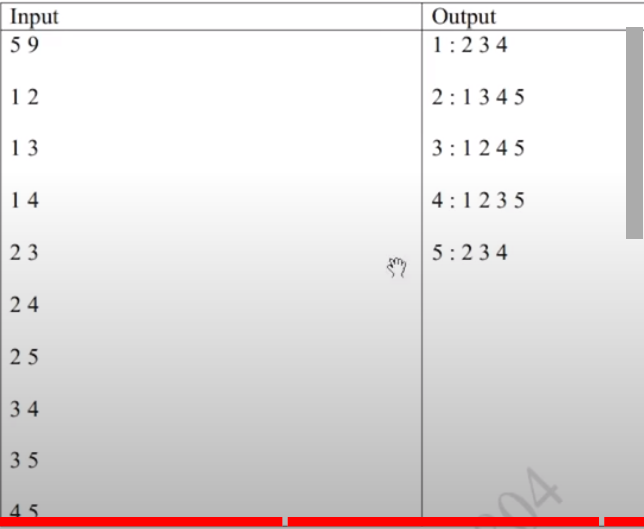
+ Đỉnh đang xét không bao giờ kề với chính nó (Trừ khi có khuyên).

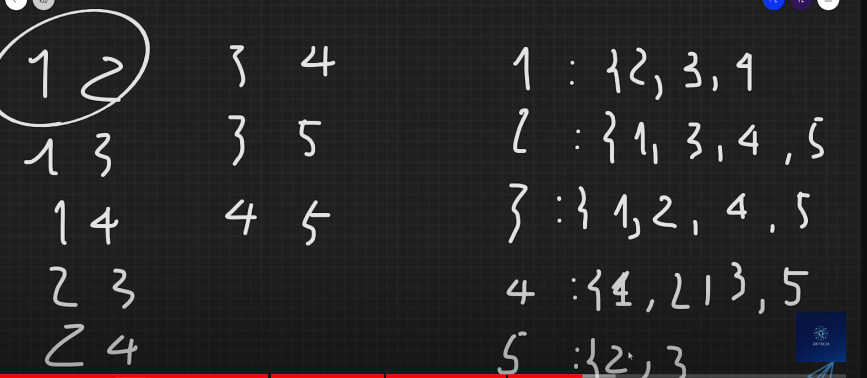
* Từ trong đồ thị nếu tồn tại cạnh AB , hoặc nó hay kí hiệu là một số cụ thể như cạnh 34 thì trong ma trận kề ta sẽ có được biểu diễn :

a[1][2] = a[2][1] = 1

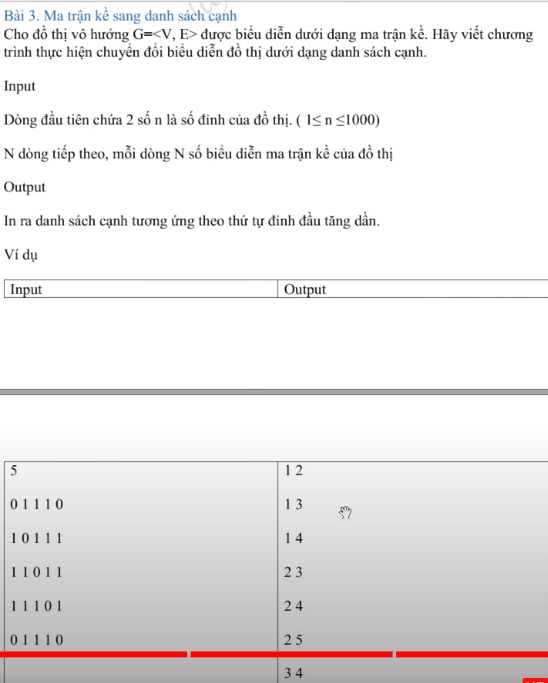
* Từ danh sách cạnh chuyển sang danh sách kề :



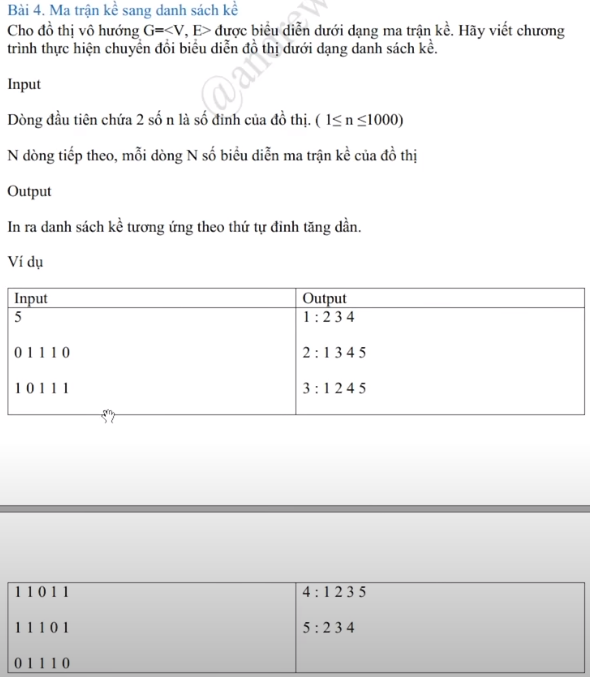


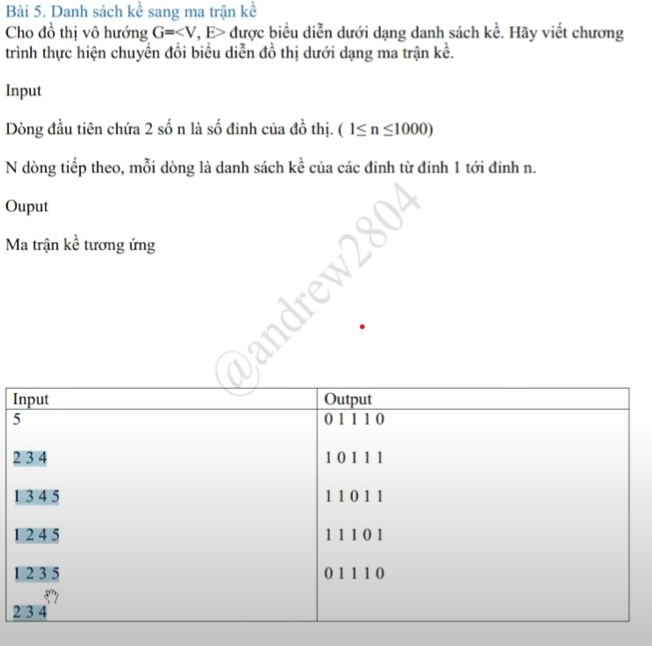


* Ma trận kề sang danh sách cạnh

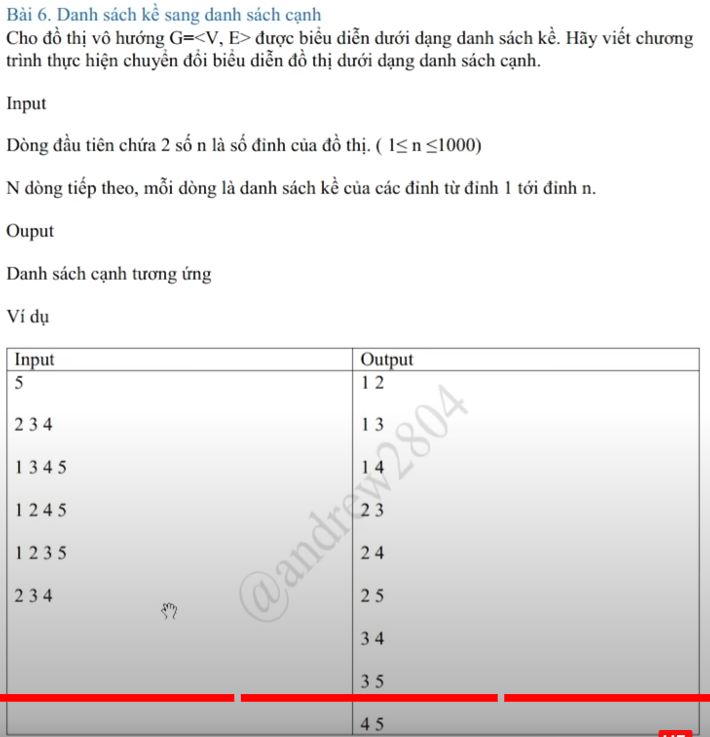


* Duyệt hàng 1 nếu mà đi qua một VỊ\_TRÍ nếu giá trị là bằng 1 thì sẽ có cạnh từ 1 đến VỊ\_TRÍ đó.
* Nhưng nếu có AB rồi thì không được liệt kê BA nữa.
* Ta có thể tạo một vector lưu các cạnh.





* Lần lượt input là danh sách các cái đỉnh kề với đỉnh i. Có một cái khó ở đây là mình không biết trước số lượng đỉnh ở từng dòng (Mỗi dòng có số lượng đỉnh khác nhau ) nên mình phải coi mỗi cái hàng là một cái string và tách nó ra thành cách số.



TỔNG QUÁT CÁC KIẾN THỨC QUAN TRỌNG

* Có 3 cách chuyển đổi biểu diễn một ma trận :

MA TRẬN KỀ :

+ a[i][j] = 1 nếu ij là một cạnh của đồ thị.

+ Bản chất kiểm tra xem các đỉnh có kề với nhau hay không. Nếu kề thì là 1 (Hai đỉnh kề nhau tức tồn tại cạnh nối trực tiếp hai đỉnh). Đỉnh đang xét không bao giờ kề với chính nó.

+ Dùng ma trận cỡ n để lưu: a[1001][1001]

DANH SÁCH CẠNH :

+ Liệt kê tất cả các cạnh (Bao gồm đỉnh đầu và cuối ) nếu trong đồ thị tồn tại cạnh. Nếu có thằng AB thì không được liệt kê thằng BA nữa(Tức là chỉ lấy những thằng i<j ).

+ Dùng vector<pair<int,int> vt để lưu danh sách cạnh.

DANH SÁCH KỀ :

+ Liệt kê xem một cạnh có bao nhiêu đỉnh kề với nó( Tức là từ đỉnh hiện tại tồn tại cạnh trực tiếp ).

+ Dùng một mảng vector để lưu , tức là mỗi phần tử trong mảng là một vector : vector<int> a[1001]

|  |  |
| --- | --- |
| MaTran -> DanhSachCanh | vector<pair<int,int>> vt;  if(a[i][j] ==1 && i< j){              vt.push\_back({i,j});          } |
| MaTran -> DanhSachKe | vector<int> vt[1000];   if(a[i][j]==1){              vt[i].push\_back(j);          } |
| DanhSachCanh -> MaTran | int a[1001][1001];  cin >> x >> y;         a[x][y] = a[y][x] = 1; |
| DanhSachCanh -> DSKe | vector <int> adj[1001];   cin >> x >> y;   adj[x].push\_back(y);   adj[y].push\_back(x); |
| DanhSachKe -> MaTran | int a[1001][1001];  *string* s,num;      getline(cin,s);    *stringstream* ss(s);      while(ss >> num){          a[i][stoi(num)] = 1;          }     } |
| DanhSachKe -> DSCanh | vector<pair<int,int> > edge;  *string* s,num ;      getline(cin,s);  *stringstream* ss(s);      while(ss >> num ){          /\* - Tranh bi lap canh -\*/          if(i < stoi(num)){          edge.push\_back({i,stoi(num)});          }      }     } |

