Mục lục:

ĐỀ CƯƠNG CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Chương 5: Đồ thị

1. Các khái niệm:

1.1. Định nghĩa đồ thị:

- Đồ thị G(V, E) bao gồm một tập hữu hạn V các đỉnh (nút), và một tập hữu hạn E các cặp đỉnh mà ta gọi là cung (cạnh).

1.2. Định nghĩa đồ thị vô hướng:

- Đồ thị vô hướng G(V, E) bao gồm V là tập các đỉnh, và E là tập các cặp đỉnh không có thứ tự gọi là các cung.

1.3. Định nghĩa đồ thị có hướng:

- Đồ thị có hướng G(V, E) bao gồm V là tập các đỉnh, và E là tập các cặp đỉnh có thứ tự gọi là các cung.

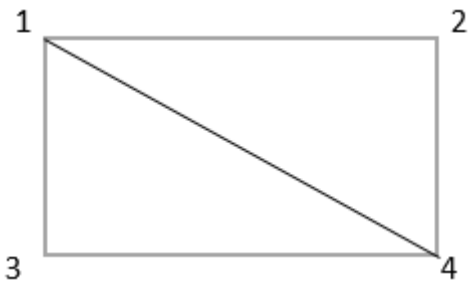
1.4. Một số khái niệm khác:

- Lân cận của nhau: Nếu v(v_1, v_2) là một cung trong tập E(G).
- Một đường đi từ đỉnh u đến đỉnh v trong đồ thị là một dãy các đỉnh u = x_0, x_1, …, x_{n-1}; x_n = v mà dãy các cạnh (x_0, x_1), (x_1, x_2), …, (x_{n-1}, x_n) là các cung thuộc E(G).
- Độ dài của đường đi: Là số lượng cung trên đường đi.
- Đường đi đơn: Là đường đi mà mọi đỉnh trên đó, trừ đỉnh đầu và đỉnh cuối đều khác nhau.
- Một chu trình: Là một đường đi đơn mà đỉnh đầu đỉnh cuối trùng nhau.
- Trong đồ thị G, hai đỉnh u và v gọi là liên thông nếu có một đường đi từ u đến v.
- Đồ thị G gọi là liên thông nếu với mọi cặp đỉnh phân biệt v_i, v_j trong V(G) đều có một đường đi từ v_i đến v_j.
- Đồ thị không liên thông?
- Đồ thị có trọng số?
- Đồ thị không trọng số?

2. Biểu diễn đồ thị: <Cấu trúc lưu trữ>

2.1. Biểu diễn bằng ma trận kề: <Lưu trữ kế tiếp>

- Xét đồ thị G(V, E), V gồm n đỉnh (n>=1). Ma trận kề A biểu diễn G là một ma trận vuông kích thước n x n phần tử. Các phần tử của ma trận có giá trị 0 và 1.
  - + A_{ij} = 1 nếu tồn tại cung (v_i, v_j) trong E.
  - + A_{ij} = 0 nếu không tồn tại cung (v_i, v_j) trong E.
- Lưu trữ kế tiếp của ma trận kề cũng giống như lưu trữ kế tiếp của mảng 2 chiều: Các phần tử trong ma trận kề được lưu trữ trong các ô nhớ liên tiếp nhau, hết hàng 1 đến hàng 2, hết hàng 2 đến hàng 3, …
- Ví dụ: Cho đồ thị vô hướng sau:



Ma trận kề A biểu diễn đồ thị trên là ma trận vuông cấp 4 có giá trị như sau:

	1	**2**	**3**	**4**
1	0	1	1	1
2	1	0	0	1
3	1	0	0	1
4	1	1	1	0

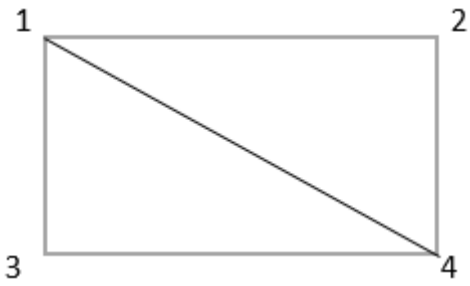
Về mặt lưu trữ thì dùng một vecto V gồm (n x n) ô nhớ có chỉ số từ 1 đến (n x n):

V	0	1	1	1	1	0	0	1	…	0
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**…**	**16**

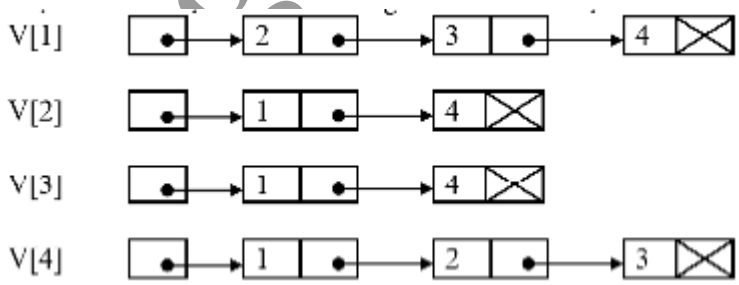
- Trong trường hợp đồ thị có trọng số thì các giá trị 1 được thay bằng trọng số tương ứng của cung đó.

2.2. Biểu diễn bằng danh sách kề: <Lưu trữ danh sách kề>

- Trong cách biểu diễn này, n hàng của ma trận kề thay bằng n danh sách. Mỗi đỉnh của G có một danh sách tương ứng. Các nút trong danh sách i biểu diễn các đỉnh lân cận của nút i.
- Mỗi đỉnh kề trong danh sách kề được lưu trữ trong các nút nhớ, mỗi nút nhớ có cấu trúc gồm 2 trường:
  + Trường Vertex: Chứa thông tin của đỉnh lân cận đỉnh i.
  + Trường Link: Chứa địa chỉ của nút tiếp theo.
- Nút cuối cùng trong danh sách không có nút đứng sau nên trường Link= ∅.
- Ví dụ: Cho đồ thị sau:



Đồ thị trên được biểu diễn bằng danh sách kề như sau:



- Mỗi danh sách có nút đầu danh sách, các nút này được tổ chức lưu trữ kế tiếp (mảng) để truy nhập được nhanh.
- Đồ thị vô hướng có n đỉnh, c cung thì cần n nút đầu danh sách và 2c nút danh sách.
- Đồ thị vô hướng có n đỉnh, c cung thì cần n nút đầu danh sách và c nút danh sách.

3. Phép duyệt đồ thị:

- Là phép thăm tất cả các đỉnh của đồ thị bắt đầu từ một đỉnh nào đó, sao cho mỗi đỉnh chỉ thăm 1 lần.
- Có 2 cách duyệt đồ thị:
  + Phép duyệt theo chiều sâu.
  + Phép duyệt theo chiều rộng.

a. Phép duyệt theo chiều sâu:

- Ý tưởng: Cho đồ thị vô hướng:
  + Đỉnh xuất phát v được thăm.
  + Tiếp theo đó, ta thăm đỉnh w là đỉnh chưa được thăm và là lân cận của v. Và một phép duyệt theo chiều sâu xuất phát từ w lại được thực hiện.
  + Khi một đỉnh u đã được thăm mà mọi đỉnh lân cận của nó đều đã được thăm thì ta sẽ quay ngược lên đỉnh cuối cùng vừa được thăm (mà còn có đỉnh w lân cận với nó chưa được thăm). Và một phép duyệt theo chiều sâu xuất phát từ w lại được thực hiện.
  + Phép duyệt sẽ kết thúc khi không còn một đỉnh nào chưa được thăm mà vẫn có thể thăm được từ đỉnh đã được thăm.
- Giả mã:

```
Procedure DFS(v)
    1.write(v);
    2.visted(v):=1; {Đánh dấu v đã được thăm}
    3.For mỗi đỉnh w lân cận với v do
        If visted(w)=0 then Call DFS(w);
Return
```

- Đánh giá giải thuật:

+ Trong trường hợp đồ thị được biểu diễn bởi 1 danh sách kề thì đỉnh w lân cận của v sẽ được xác định bằng cách dựa vào danh sách liên kết ứng với v. Vì giải thuật DFS chỉ xem xét mỗi nút trong một danh sách lân cận nhiều nhất 1 lần thôi mà lại có 2e nút danh sách (ứng với e cung), nên thời gian để hoàn thành phép duyệt này là O(e).

+ Con nếu đồ thị được biểu diễn bởi ma trận kề thì thời gian để xác định lân cận của v là O(n). Vì tối đa có n đỉnh được thăm, nên thời gian duyệt tổng quát là $O(n^2)$.

b. Phép duyệt theo chiều rộng:

- Ý tưởng:

Xét đồ thị vô hướng. Phép tìm kiếm theo chiều rộng thể hiện như sau:

Đỉnh xuất phát v được thăm.

Tiếp theo các đỉnh chưa được thăm mà là lân cận của v sẽ được thăm, rồi đến các đỉnh chưa được thăm là lân cận lần lượt của các đỉnh này và cứ tương tự như vậy.

- Giả mã:

Thủ tục phép duyệt theo chiều rộng như sau:

Cho một đồ thị G(V, E) vô hướng có n đỉnh và vecto Visited(n) gồm n phần tử, ban đầu vecto này có giá trị = 0. Thuật giải này thực hiện thăm mọi đỉnh "với tới được" từ đỉnh v. Bắt đầu từ đỉnh v. Mọi đỉnh I được thăm đánh dấu bằng Visited(i):=1.

Dùng hàng đợi Q có kích thước n; F, R là lối trước và lối sau của hàng đợi. Khi thăm 1 đỉnh thì loại bỏ khỏi hàng đợi; Khi chưa thăm thì bổ sung vào hàng đợi.

Procedure BFS(v)

1)Khởi tạo hàng đợi Q với v được đưa vào.

2)Visited(v):=1 {Đánh dấu v được thăm}

3)While Q không rỗng do

Begin

+)Call CQDelete(v, Q) {Loại bỏ v ra khỏi Q}

+)For mỗi đỉnh w lân cận với v do

If Visited(w)=0 then

Begin

+)Call CQInsert(w, Q); {Bổ sung w vào Q}

+)Visited(w):=1;

End;

End;

Return

- Đánh giá giải thuật:

Nếu biểu diễn đồ thị bằng ma trận lân cận thì thời gian thực hiện là $O(n^2)$.

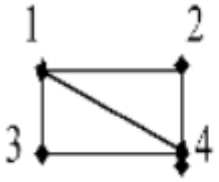
Nếu biểu diễn đồ thị bằng danh sách lân cận thì thời gian thực hiện là O(e).

4. Cây khung và cây khung tối thiểu.

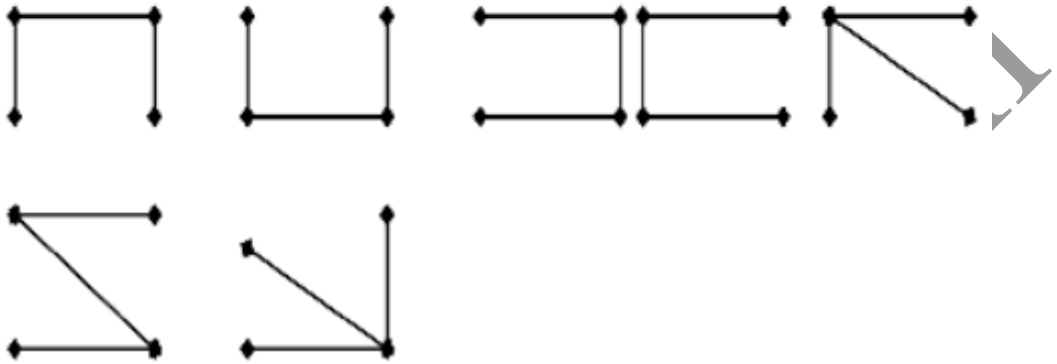
4.1. Cây khung:

- Nếu G là đồ thị liên thông thì phép tìm kiếm theo chiều sâu hoặc theo chiều rộng xuất phát từ 1 đỉnh thăm mọi đỉnh. Như vậy các cung trong G phân thành 2 tập:
  - Tập T chứa các cung đã được duyệt qua.
  - Tập b gồm các cung còn lại.
- Tất cả các cung và các đỉnh trong T sẽ tạo thành một cây con bao gồm mọi đỉnh của G. Cây con như vậy gọi là cây khung của G.

Ví dụ 1: Cho đồ thị sau

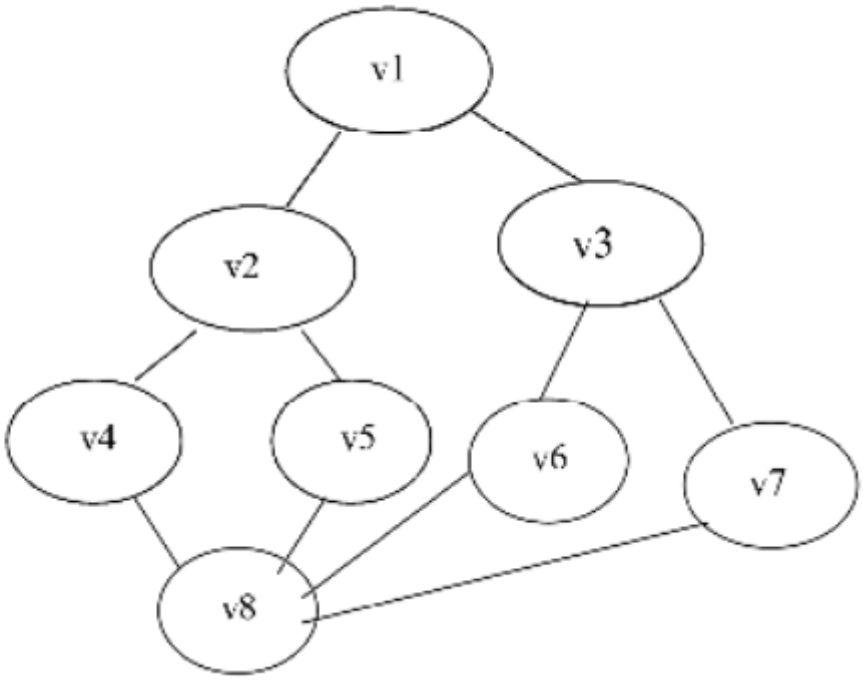


Các cây khung của nó là:

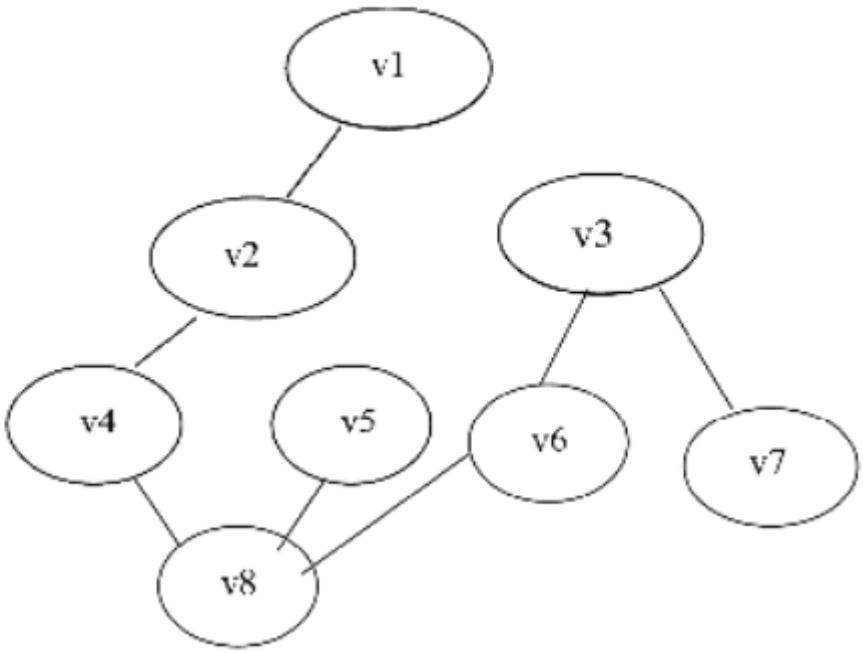


- Phép duyệt cây theo chiều sâu (DFS) cho cây khung theo chiều sâu.
- Phép duyệt theo chiều rộng cho cây khung theo chiều rộng.

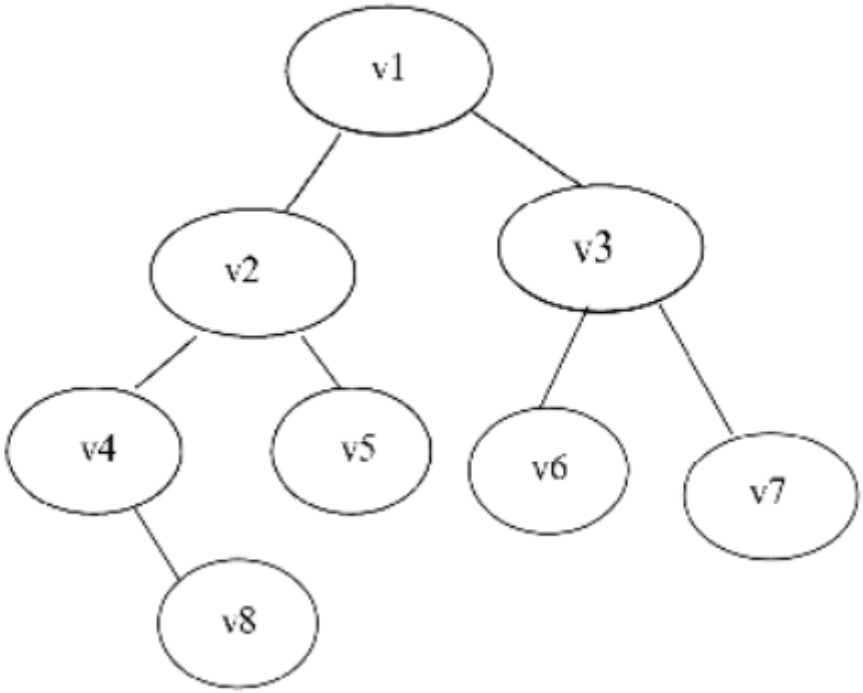
Ví dụ 2: cho đồ thị sau



+ Cây khung theo chiều sâu:



+ Cây khung theo chiều rộng:



4.2. Cây khung với giá trị cựa tiểu:

- Bài toán: Xác định cây khung với giá trị cực tiểu của đồ thị liên thông có trọng số.

  Giá trị của cây khung là tổng các trọng số ứng với các cạnh của cây khung.

- Có nhiều giải thuật xác định cây khung với giá trị cực tiểu nhưng trong phần này ta chỉ xét giải thuật Kruskal. Với giải thuật này cây khung T sẽ được xây dựng dần dần từng cung một. Các cung đưa vào T thỏa mãn:
  - Cung có giá trị cực tiểu trong các cung còn lại.
  - Không tạo ra chu trình với các cung đã có của T.
- Giả mã của giải thuật Kruskal được viết như sau:

1)T=NULL;

2)While T chứa ít hơn (n-1) cung do

Begin

Chọn 1 cung (v, w) từ E có giá trị nhỏ nhất.

Loại (v, w) ra khỏi E.

If (v, w) không tạo nên chu trình trong T then đưa (v, w) vào T.

End

Return

- Đanh giá giải thuật:

Trong trường hợp xấu nhất sẽ là O(c.Log c) trong đó c là số cung của đồ thị G.

5. Bài toán tìm đường đi ngắn nhất.

- Phát biểu bài toán:

Cho đồ thị có hướng G(V, E), một hàm trọng số w(c) cho các cung c của G và một đỉnh nguồn v_0.

Bài toán đặt ra là: Xác định đường đi ngắn nhất từ v_0 đến mọi đỉnh còn lại của G (Độ dài đường đi là tổng các trọng số trên các cung đường đi đó và các trọng số đều dương).

* Gọi S là tập các đỉnh kể cả v0 mà đường đi ngắn nhất xác lập.

Đối với 1 đỉnh w ∈ S, gọi Dist(w) là độ dài của đường đi ngắn nhất từ v0 qua các đỉnh trong S và kết thúc ở w thì sẽ có một số nhận xét sau:

1. Nếu đường đi ngắn nhất tới w thì đường đi đó bắt đầu từ v0 kết thúc ở w và chỉ đi qua những đỉnh thuộc S.
2. Đích của đường đi sinh ra tiếp theo phải là một đỉnh w nào đó ∉ S mà có Dist(w) ngắn nhất so với mọi đỉnh ∉ S .
3. Nếu đã chọn được một đỉnh w như trong nhận xét 2 ở trên và sinh ra một đường đi ngắn nhất từ v0 đến w thì w sẽ trở thành 1 phần tử của S.

* Giải thuật:

```
Procedure   Shortest_Path(v,Cost,Dist,n)
{ Dist(j) 1<= j <=n là độ dài đường đi ngắn nhất từ v đến j trong đồ thị có
hướng G có n đỉnh, Dist(v)=0. G được biểu diễn bởi ma trận lân cận Cost có
kích thước n x n. }
1. For  i:=1 To n Do Begin
            S[i]:=0;  Dist[i]:= Cost[v,i];
        End;
2. S[v]:=1; Dist[v]:=0; k:=1;  { đưa v vào S }
3. While  k<n  { xác định n-1 đường đi từ đỉnh v }
4. Begin
        Chọn u sao cho Dist[u]= min(Dist[i]))  với  S[i]=0;
5. S[u]:=1; k:=k+1;  { đưa u vào S}
6. For  mọi w với S[w]=0 Do
7.    Dist[w]:= min(Dist[w], Dist[u]+Cost[u,w])  { tính lại khoảng cách theo
đường đi ngắn nhất }
  End;
8. Return
```

* Dựa trên các quan điểm như các nhận xét nêu trên Diskstra đưa ra giải thuật tìm đường đi ngắn nhất như sau:
  - Giả thiết n đỉnh của G được đánh số từ 1 tới n.
  - Tập S được thể hiện bằng véc tơ bít: S[i] = 0 nếu đỉnh i $\notin$ S
    S[i] = 1 nếu đỉnh i $\in$S
  - Độ dài trọng số biểu diễn bằng ma trận lân cận Cost:
    Cost[i,j] là trọng số cung (i,j)
    Cost[i,j] = + ∞ nếu cung (i,j) không có.
    Cost[i,j] = 0 nếu i=j

Câu hỏi ôn tập cuối chương:

Câu 27: Trình bày đặc điểm tổ chức của cấu trúc dữ liệu đồ thị.

Câu 28: Trình bày cấu trúc lưu trữ của đồ thị.

Câu 29: Viết giả mã các phép toán duyệt đồ thị. Áp dụng duyệt cây.