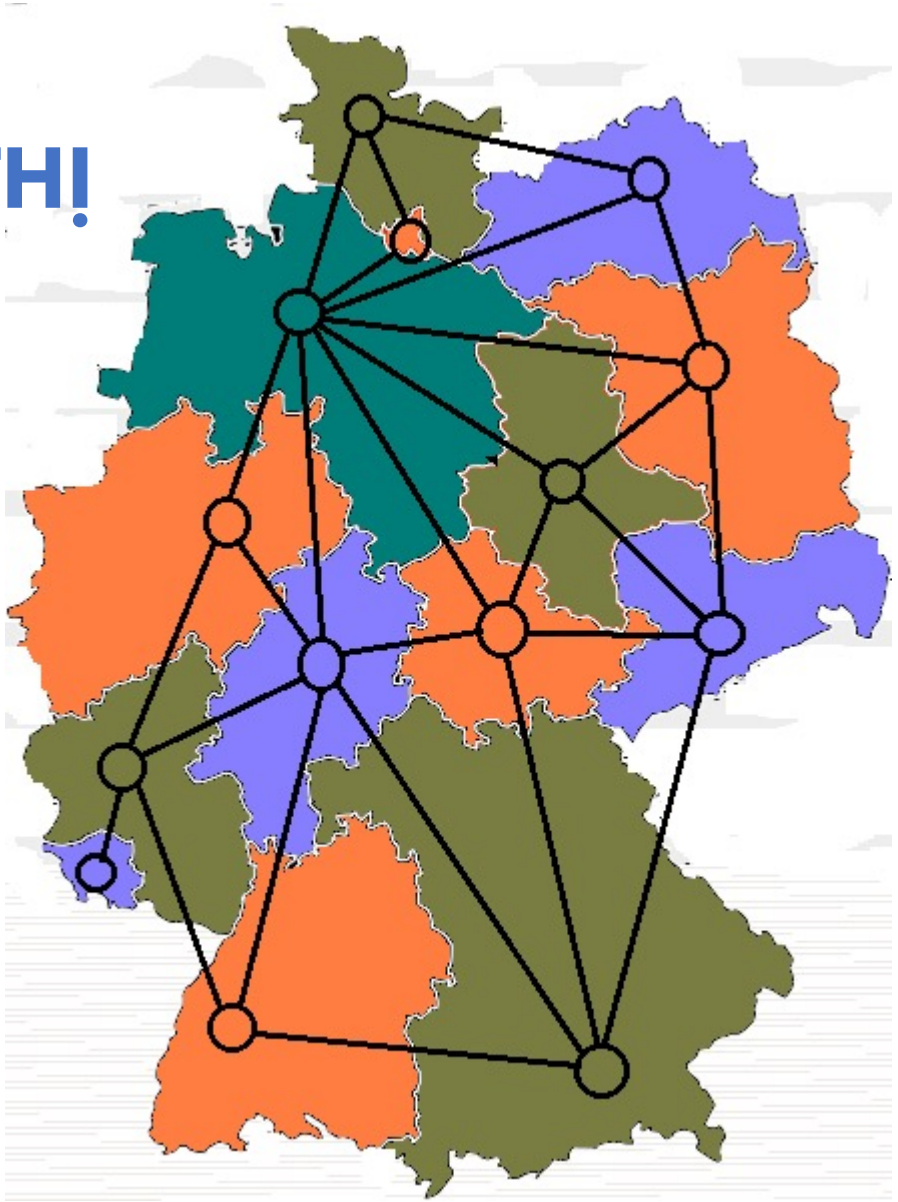
CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ

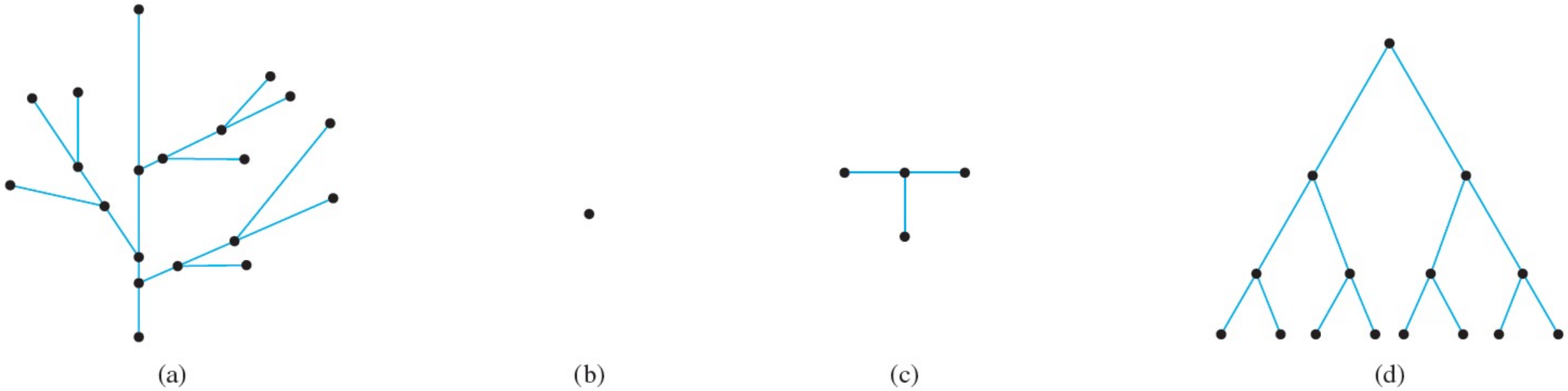


Vũ Ngọc Thanh Sang

CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ

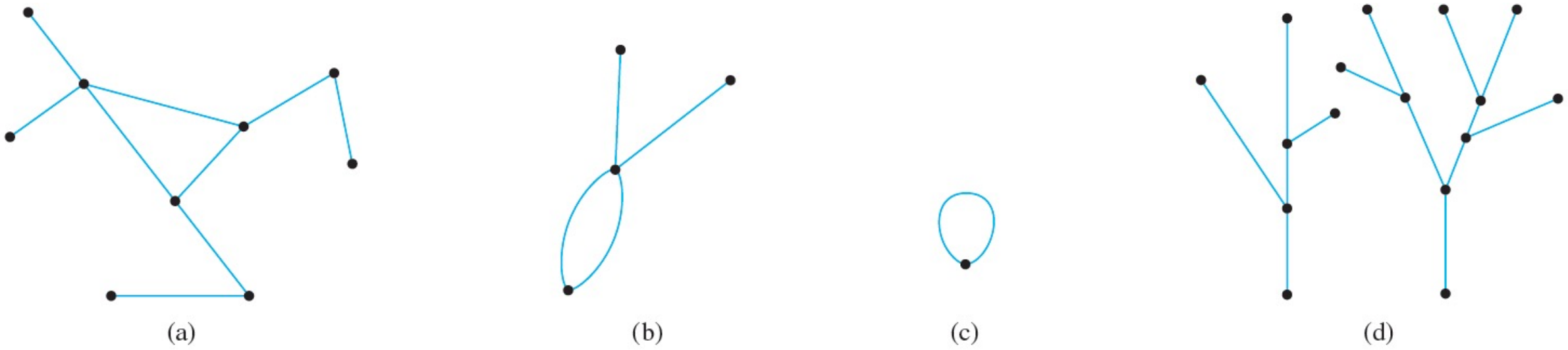
- Một đồ thị được gọi là **cây** khi và chỉ khi nó không có cạnh khuyên và liên thông.
- **Cây tầm thường** (trivial tree) là một đồ thị bao gồm một đỉnh duy nhất.
- **Rừng** (forest) là đồ thị bao gồm các cây không liên thông.

CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ



Ví dụ về đồ thị là cây

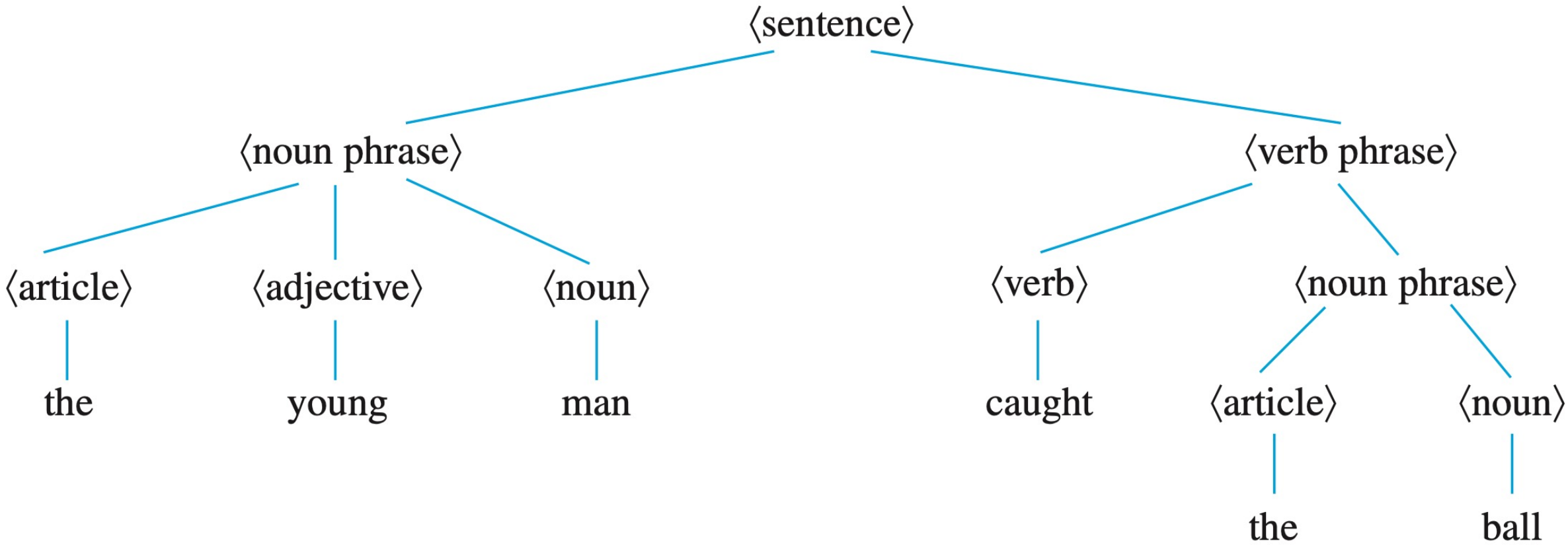
CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ



Ví dụ về đồ thị không phải là cây, đồ thị (d) là ví dụ của rừng

CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ

Cây phân tích cú pháp (A Parse Tree)



1. CÂY VÀ CÁC TÍNH CHẤT CỦA CÂY

Định lý 1: Đồ thị T vô hướng **n đỉnh** là một cây nếu thỏa **một** trong các tính chất sau

- T không chứa chu trình và có n -1 cạnh
- T liên thông và có n -1 cạnh
- T liên thông và mỗi cạnh của nó đều là cạnh cầu
- Hai đỉnh bất kỳ được nối với nhau bằng một đường đi duy nhất
- T không chứa chu trình nhưng nếu thêm vào một cạnh thì có một chu trình (circuit) duy nhất

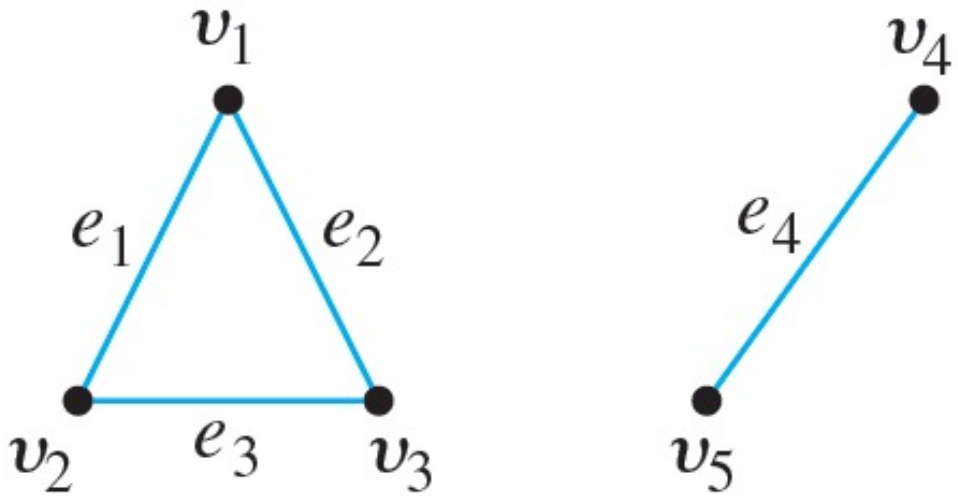
1. CÂY VÀ CÁC TÍNH CHẤT CỦA CÂY

Định lý 1: Đồ thị T vô hướng **n đỉnh** là một cây nếu thỏa **một** trong các tính chất sau

- T không chứa chu trình và có n -1 cạnh
- T liên thông và có n -1 cạnh
- T liên thông và mỗi cạnh của nó đều là cạnh cầu
- Hai đỉnh bất kỳ được nối với nhau bằng một đường đi duy nhất
- T không chứa chu trình nhưng nếu thêm vào một cạnh thì có một chu trình (circuit) duy nhất

1. CÂY VÀ CÁC TÍNH CHẤT CỦA CÂY

Cho ví dụ về một đồ thị 5 đỉnh 4 cạnh nhưng không phải là cây



Hệ quả 1: Nếu đồ thị G có n đỉnh và m cạnh và $m \geq n$ thì đồ thị G có chu trình (circuit).

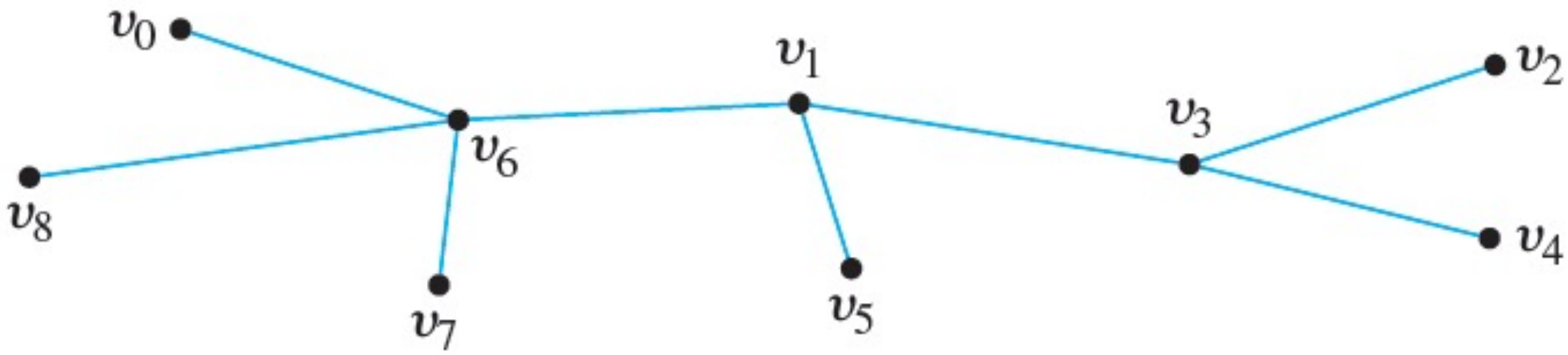
1. CÂY VÀ CÁC TÍNH CHẤT CỦA CÂY

Bổ đề 1: Một cây có nhiều hơn một định sẽ có ít nhất một đỉnh có bậc 1.

Định lý 2: Nếu cây T có ít nhất hai đỉnh, thì đỉnh bậc 1 trong T được gọi là nút lá (leaf node), và đỉnh có bậc lớn hơn 1 trong T được gọi là nút bên trong (internal node).

1. CÂY VÀ CÁC TÍNH CHẤT CỦA CÂY

Xác định nút là và nút bên trong của cây sau đây



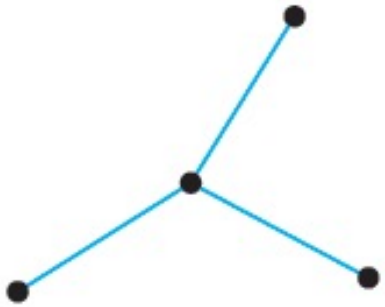
1. CÂY VÀ CÁC TÍNH CHẤT CỦA CÂY

Định lý 2: Một cây có n đỉnh (n nguyên dương) sẽ có n – 1 cạnh.

Tìm tất cả các cây không đồng phôi có 4 đỉnh.

Theo định lý 2: một cây có 4 đỉnh sẽ có 3 cạnh. Vậy số tổng bậc của cây có 4 đỉnh sẽ là 6. Theo bổ đề 1: Một cây có nhiều hơn một định sẽ có ít nhất một đỉnh có bậc 1, ta có các tổ hợp sau:

1, 1, 1, 3 và 1, 1, 2, 2

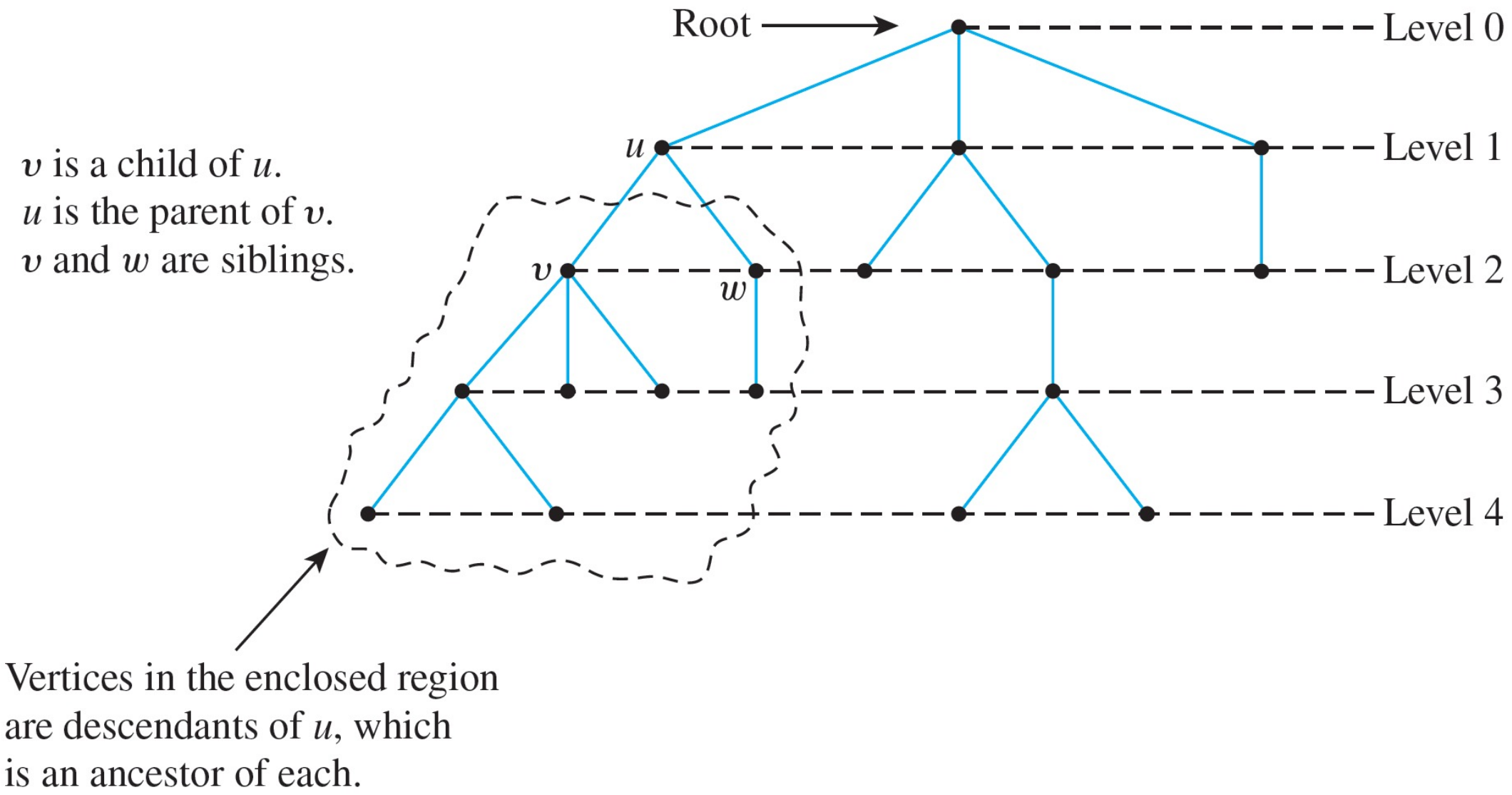


and

2. CÂY GỐC

- **Cây gốc** (rooted tree) là cây trong đó có một đỉnh được phân biệt với các đỉnh khác và được gọi là gốc.
- **Mức** của một đỉnh là số cạnh dọc theo đường đi duy nhất giữa nó và gốc.
- **Chiều cao** của cây gốc là mức tối đa của bất kỳ đỉnh nào của cây.
- Cho gốc hoặc bất kỳ đỉnh bên trong v của cây gốc, các **nút con** của v là tất cả các đỉnh kề với v và cách gốc một bậc so với v.
- Nếu nút w là con của nút v thì v được gọi là **nút cha** của w.

2. CÂY GỐC


Root
Level 0
u
Level 1
v
w
Level 2
Level 3
Level 4
v is a child of u.
u is the parent of v.
v and w are siblings.
Vertices in the enclosed region
are descendants of u, which
is an ancestor of each.

2. CÂY GỐC

Với cây gốc v_0 được hiển thị bên dưới.

một, xác định:

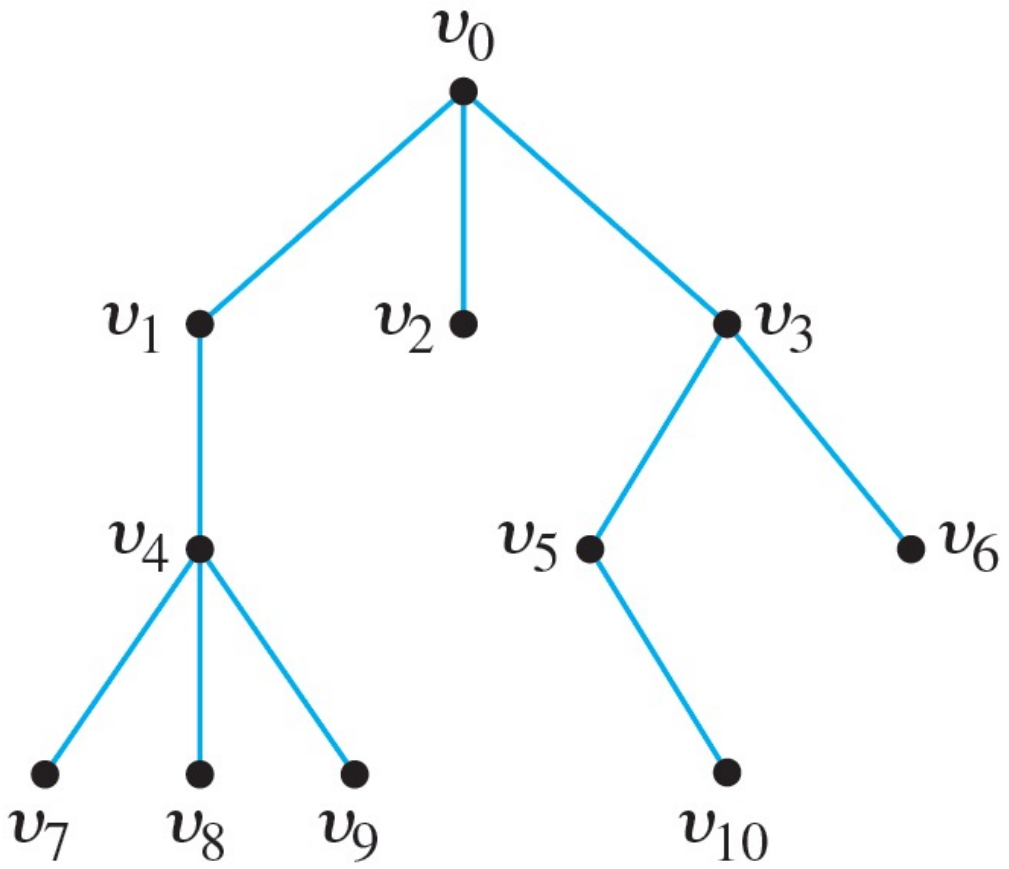
a. Mức của v_5 là bao nhiêu?

b. Mức của v_0 là bao nhiêu?

c. Chiều cao của cây gốc này là bao nhiêu?

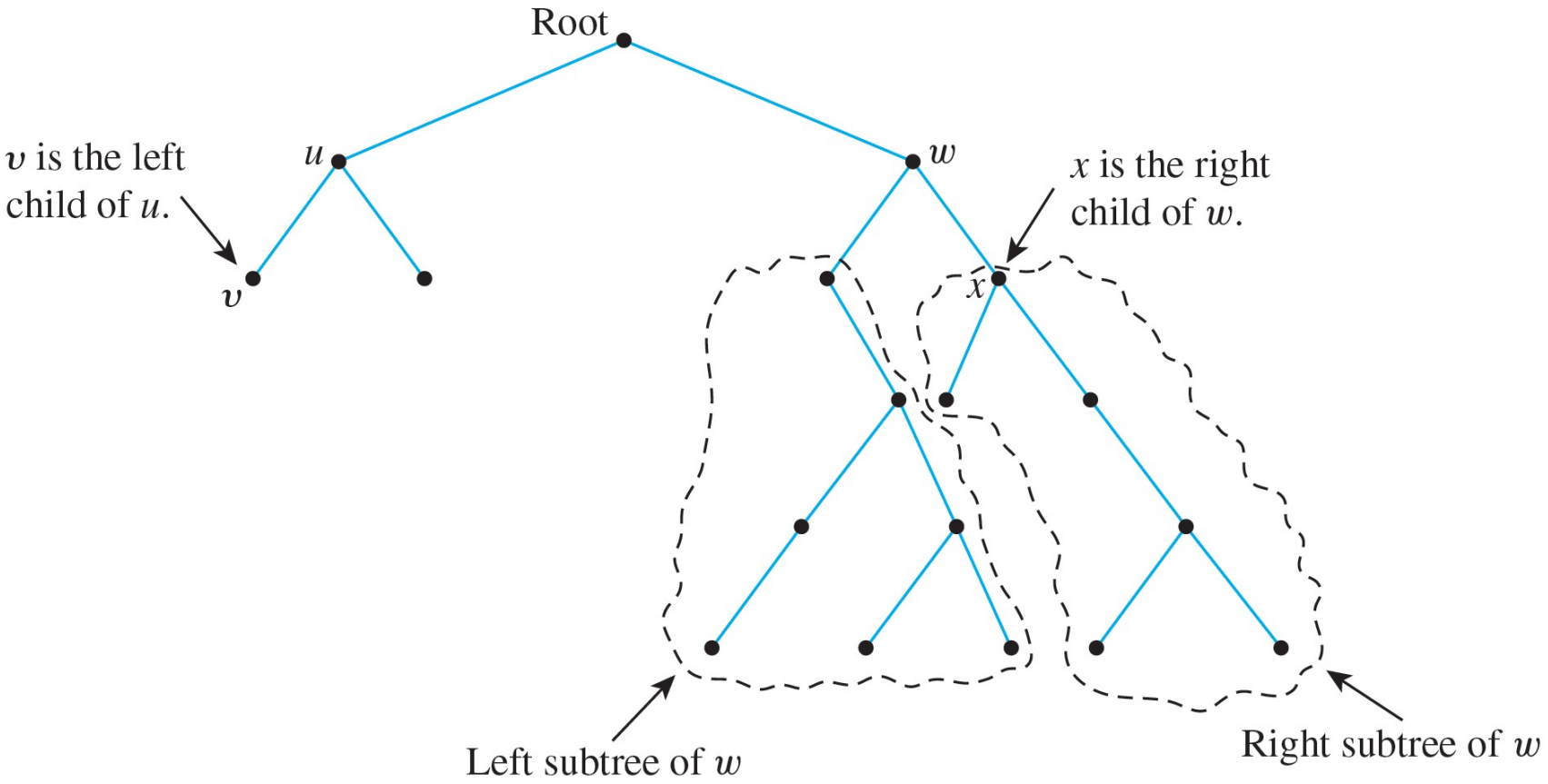
d. Những nút con của v_3 là gì?

e. Nút cha của v_2 là gì?



3. CÂY NHỊ PHÂN

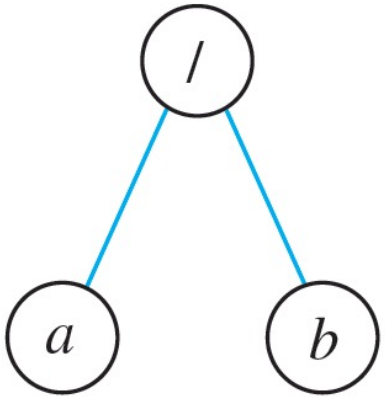
- **Cây nhị phân** là cây gốc trong đó mỗi nút cha có nhiều nhất hai nút con. .
- **Cây nhị phân đầy đủ** là cây nhị phân trong đó mỗi nút cha có đúng hai nút con.



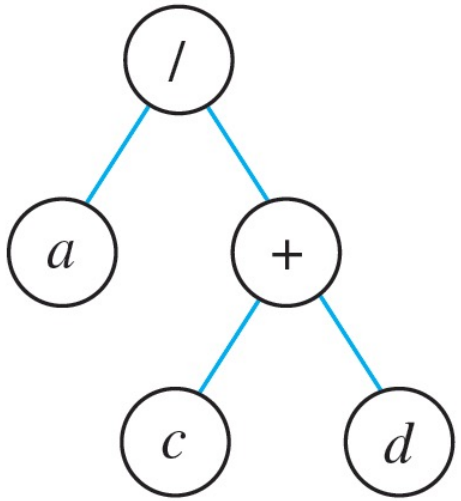
3. CÂY NHỊ PHÂN

- **Ứng dụng:** Biểu diễn biểu thức đại số

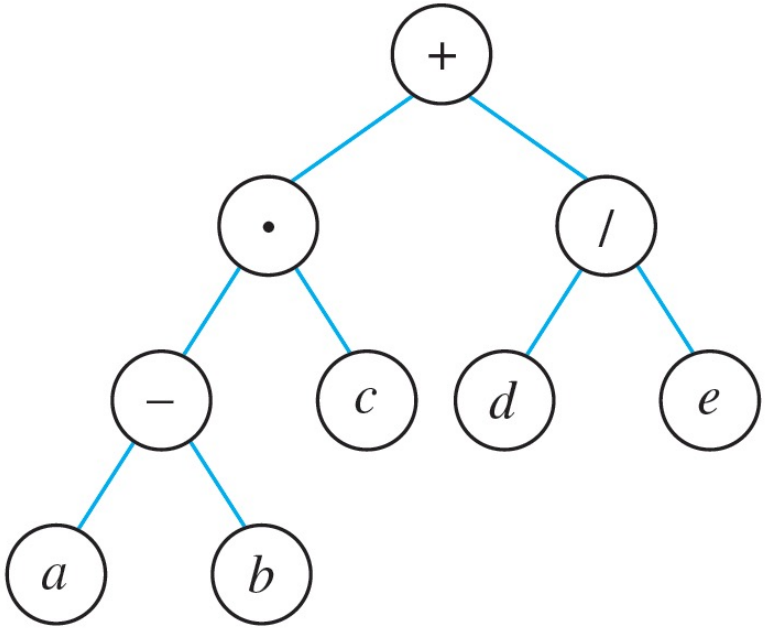
`a/b`



`a/(c+d)`



`((a-b)*c)-(d/e)`



3. CÂY NHỊ PHÂN

Định lý 3: Nếu cây T là cây đầy đủ và có k nút bên trong, thì cây T có:

(1) 2k + 1 đỉnh

(2) k + 1 nút lá

Cây nhị phân đầy đủ có 10 nút bên trong và 13 nút lá có tồn tại hay không?

3. CÂY NHỊ PHÂN

Định lý 4: Cây nhị phân T bất kì có chiều cao h và l nút lá, thõa:

$$l \leq 2^h$$

hoặc

$$\log_2 l \leq h$$

Cây nhị phân có chiều cao là 5 và 38 nút lá có tồn tại hay không?

Hệ quả: Cây nhị phân đầy đủ có chiều cao h có 2^h nút lá.

4. CÂY KHUNG CỦA ĐỒ THỊ

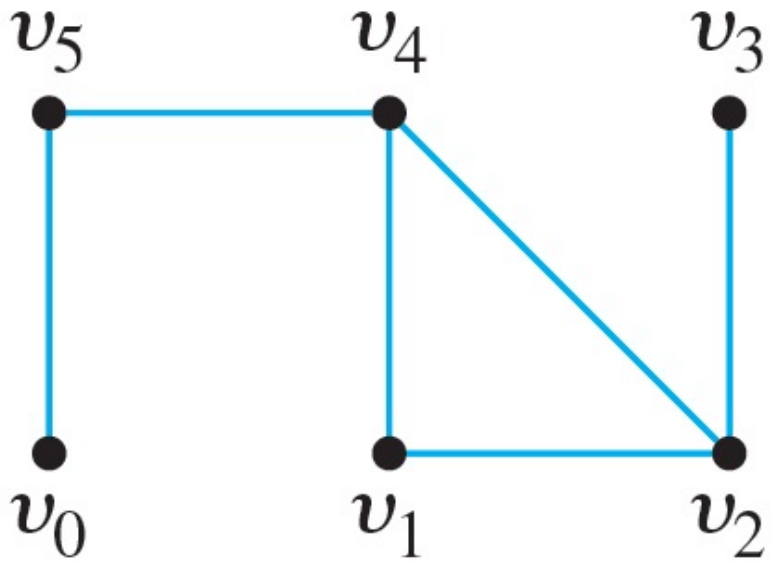
Định lý 5: Cây khung (spanning tree) của đồ thị G là đồ thị con của G bao gồm tất cả các đỉnh của G và là một cây.

Tiên đề:

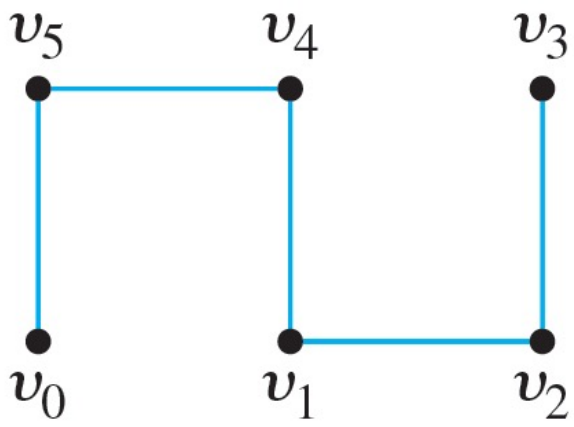
1. Mọi đồ thị liên thông đều có cây khung.
2. Bất kỳ hai cây khung nào của đồ thị đều có cùng số lượng đỉnh.

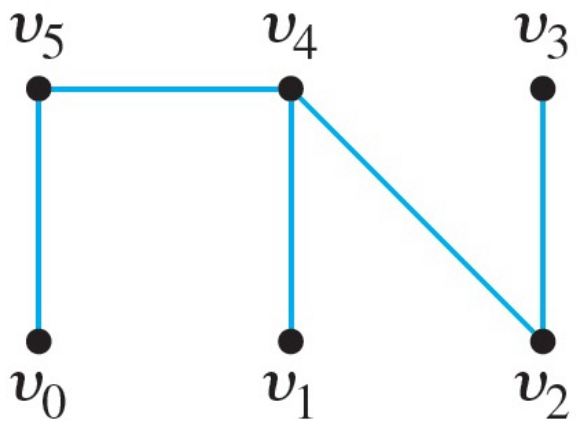
4. CÂY KHUNG CỦA ĐỒ THỊ

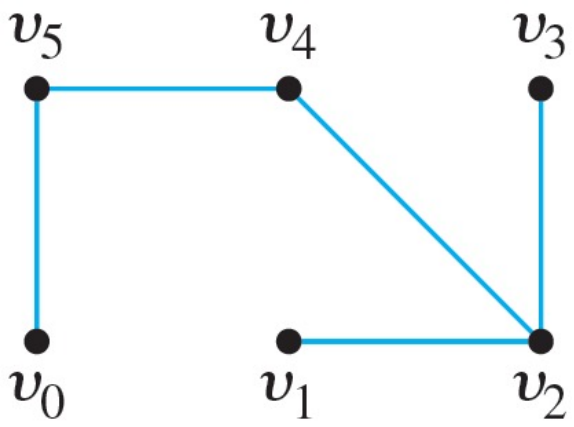
Tìm tất cả cây khung của đồ thị sau



4. CÂY KHUNG CỦA ĐỒ THỊ

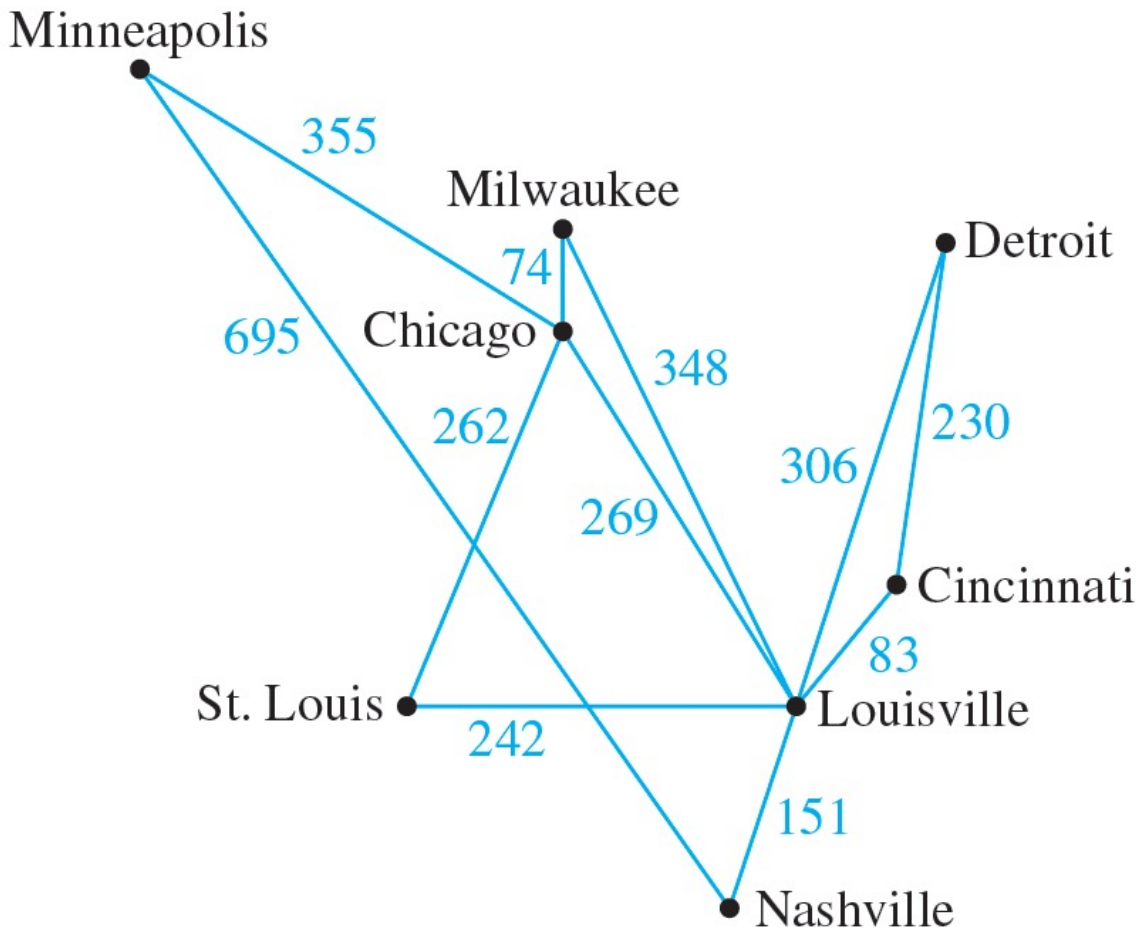






5. CÂY KHUNG NHỎ NHẤT

Giả sử công ty hàng không muốn phục vụ tất cả các thành phố được hiển thị, nhưng với một hệ thống mà trong đó tổng đường đường bay trên toàn hệ thống là nhỏ nhất.



5. CÂY KHUNG NHỎ NHẤT

- **Đồ thị có trọng số** là đồ thị mà mỗi cạnh có liên quan đến một trọng số (số thực dương).
- **Cây khung nhỏ nhất**/Cây bao trùm tối thiểu của một đồ thị liên thông, có trọng số là cây khung có tổng trọng số nhỏ nhất so với tất cả các cây khung khác cho đồ thị.
- Nếu G là một đồ thị có trọng số và e là một cạnh của G thì w (e) biểu thị trọng số của e và w (G) biểu thị tổng trọng số của G.

5. CÂY KHUNG NHỎ NHẤT

- **Kruskal's Algorithm**
  - Kruskal's Algorithm xây dựng cây khung bằng cách thêm từng cạnh một vào cây khung đang phát triển.
  - Kruskal's Algorithm tuân theo phương pháp tham lam vì trong mỗi lần lặp lại, nó tìm một cạnh có trọng số nhỏ nhất và thêm nó vào cây khung đang phát triển.
    1. Sắp xếp các cạnh của đồ thị theo trọng số của chúng.
    2. Bắt đầu thêm các cạnh vào cây khung nhỏ nhất từ cạnh có trọng số nhỏ nhất cho đến cạnh có trọng số lớn nhất.
    3. Chỉ thêm các cạnh không tạo thành chu trình (các cạnh chỉ kết nối các thành phần bị ngắt kết nối).

5. CÂY KHUNG NHỎ NHẤT

- **Kruskal's Algorithm**

Input: *G [a connected, weighted graph with n vertices, where n is a positive integer]*

Algorithm Body:

[Build a subgraph T of G to consist of all the vertices of G with edges added in order of increasing weight. At each stage, let m be the number of edges of T.]

1. Initialize T to have all the vertices of G and no edges.
2. Let E be the set of all the edges of G, and let $m := 0$.
3. **while** $(m < n - 1)$
   - 3a. Find an edge e in E of least weight.
   - 3b. Delete e from E.
   - 3c. **if** addition of e to the edge set of T does not produce a circuit **then** add e to the edge set of T and set $m := m + 1$

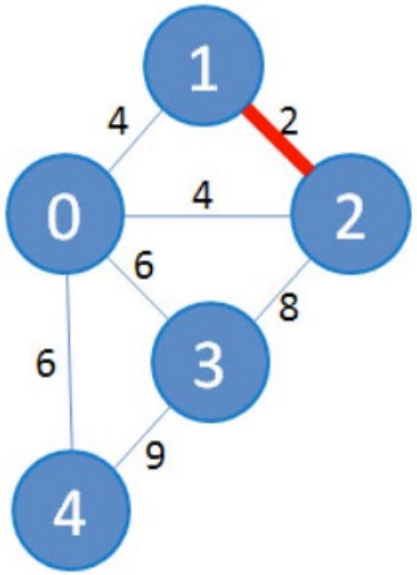
   end while

Output: *T [T is a minimum spanning tree for G.]*

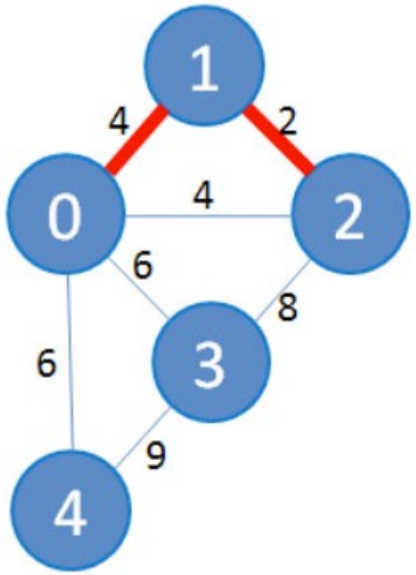
5. CÂY KHUNG NHỎ NHẤT

- **Kruskal's Algorithm**

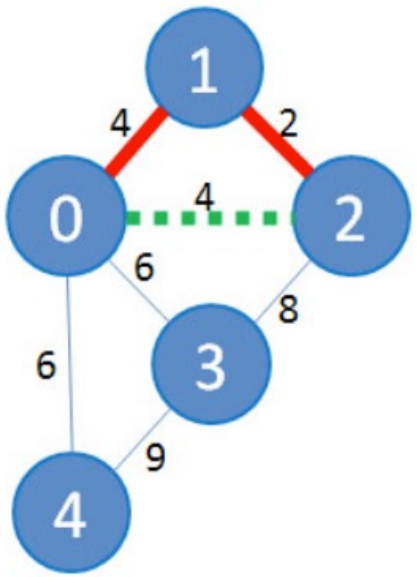
Connect 1 and 2
As this edge is smallest



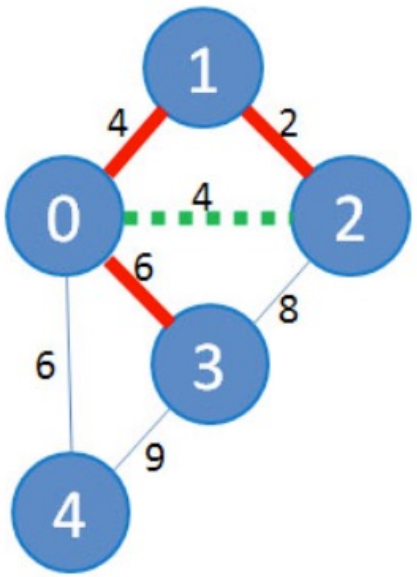
Connect 1 and 0
No cycle is formed



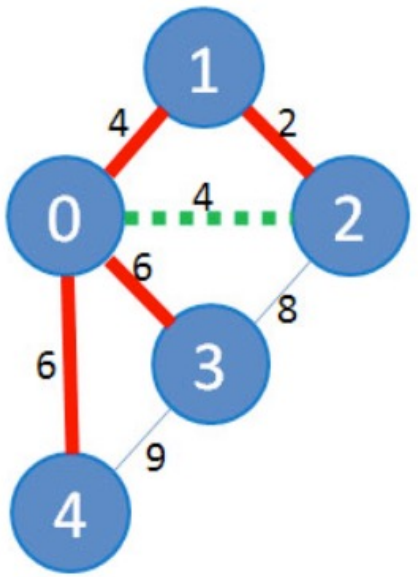
Cannot connect 0 and 2
As it will form a cycle



Connect 0 and 3
The next smallest edge



Connect 0 and 4
MST is formed...



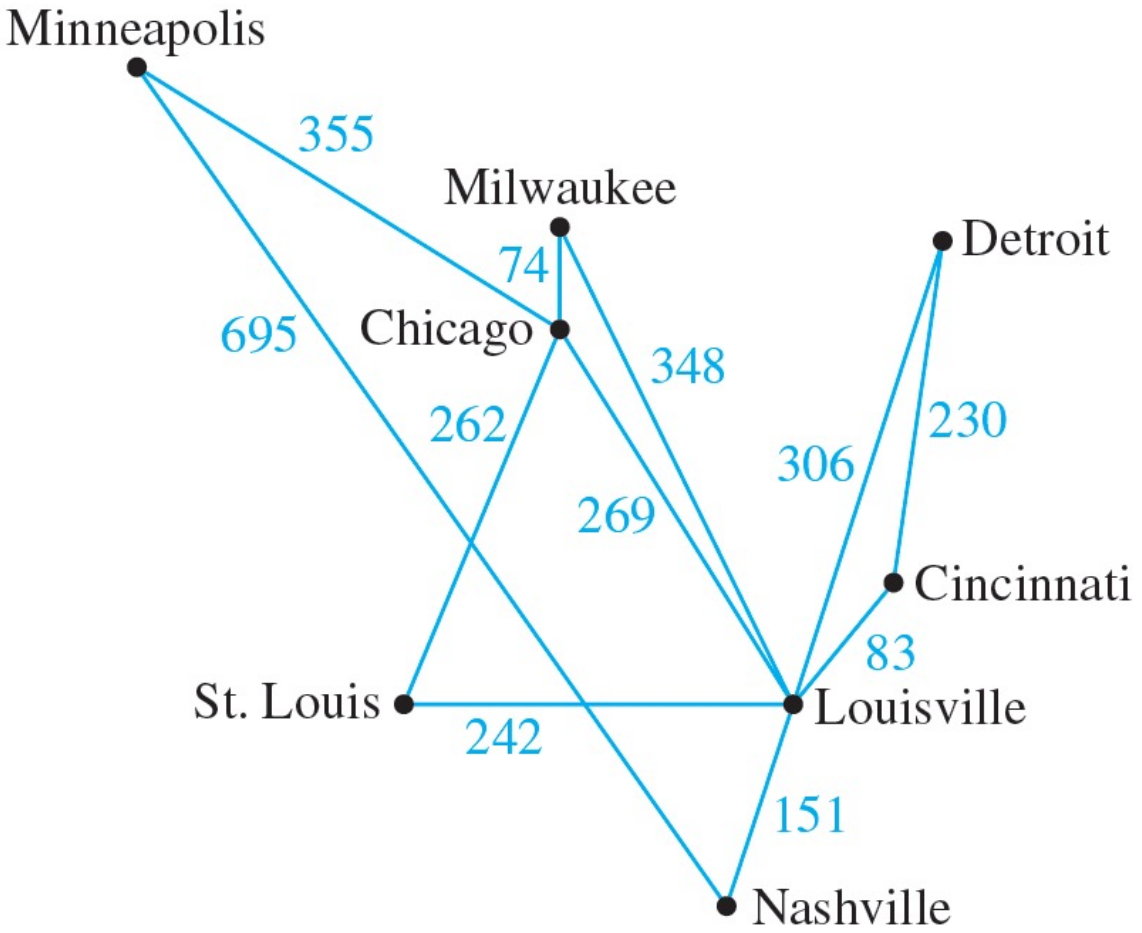
Note: The sorted order of the edges determines how the MST is formed.

Observe that we can also choose to
connect vertex 2 and 0 also with weight 4!

Connecting 0 and 4 is also a valid next move

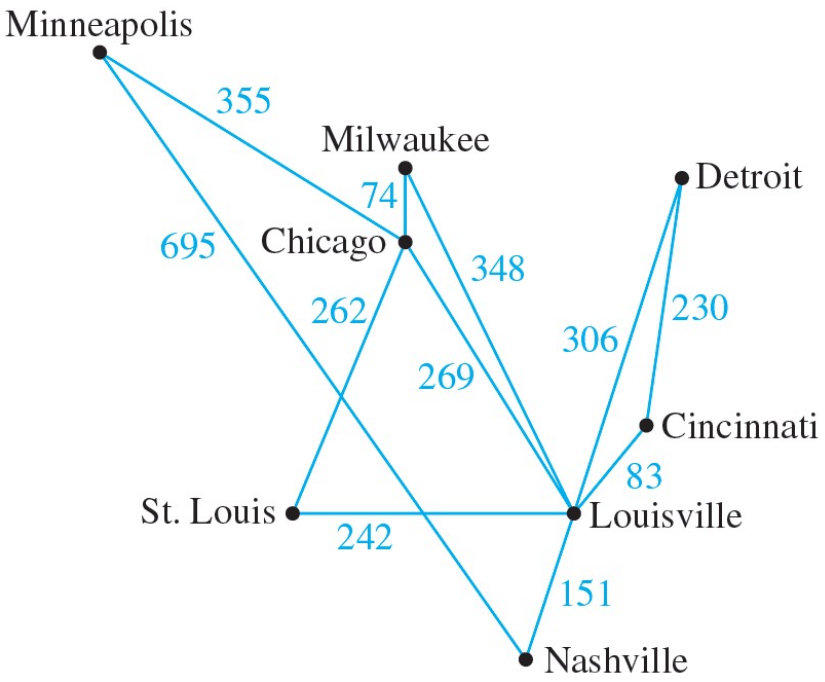
5. CÂY KHUNG NHỎ NHẤT

- **Áp dụng thuật toán Kruskal để xác định cây khung nhỏ nhất của đồ thị sau**



5. CÂY KHUNG NHỎ NHẤT

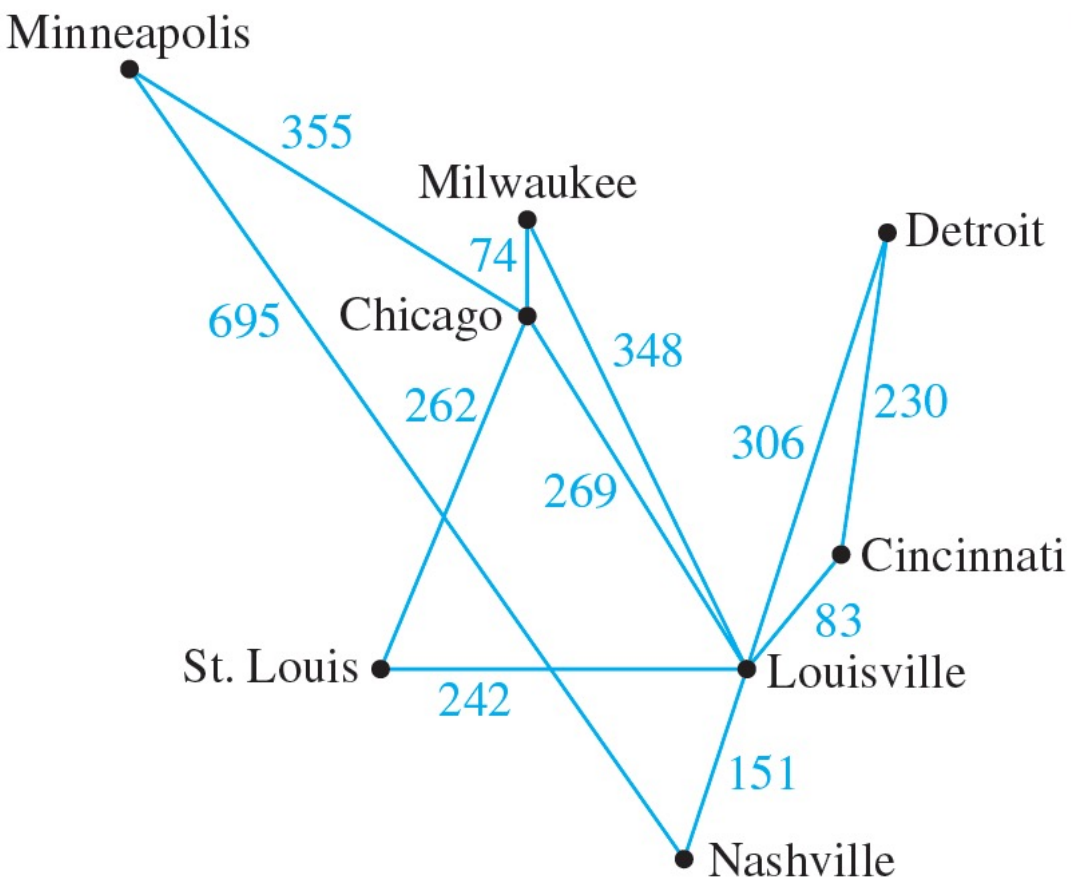
- **Áp dụng thuật toán Kruskal để xác định cây khung nhỏ nhất của đồ thị sau**

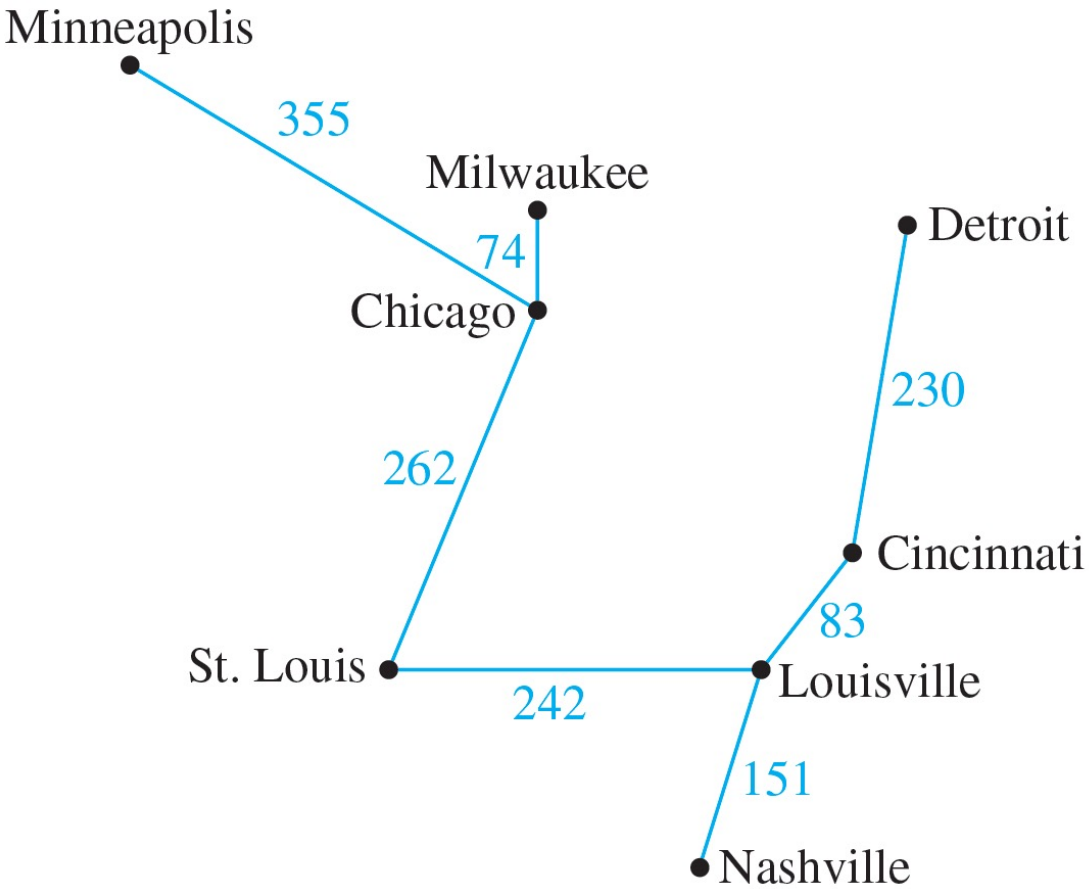


Iteration Number	Edge Considered	Weight	Action Taken
1	Chicago–Milwaukee	74	added
2	Louisville–Cincinnati	83	added
3	Louisville–Nashville	151	added
4	Cincinnati–Detroit	230	added
5	St. Louis–Louisville	242	added
6	St. Louis–Chicago	262	added
7	Chicago–Louisville	269	not added
8	Louisville–Detroit	306	not added
9	Louisville–Milwaukee	348	not added
10	Minneapolis–Chicago	355	added

5. CÂY KHUNG NHỎ NHẤT

- **Áp dụng thuật toán Kruskal để xác định cây khung nhỏ nhất của đồ thị sau**





5. CÂY KHUNG NHỎ NHẤT

- **Prim’s Algorithm**

1. Duy trì hai tập hợp đỉnh rời rạc: một tập hợp chứa các đỉnh trong cây khung đang phát triển và một bộ chứa các đỉnh không nằm trong cây khung đang phát triển.
2. Chèn các đỉnh, được kết nối với câChọn đỉnh có chi phí thấp nhất được kết nối với cây khung đang phát triển và không nằm trong cây khung đang phát triển để thêm nó vào cây khung đang phát triển.
3. y khung đang phát triển, vào hàng đợi ưu tiên.
4. Kiểm tra các chu trình bằng cách đánh dấu các nút đã được chọn và chỉ chèn những nút đó vào hàng đợi nếu nó chưa được duyệt.
5. Trong Thuật toán của Prim, chúng ta sẽ bắt đầu với một nút tùy ý (không quan trọng là nút nào) và đánh dấu nó là đã duyệt.
6. Trong mỗi lần lặp, chúng ta sẽ duyệt một đỉnh mới kề với đỉnh mà chúng ta đã duyệt.

5. CÂY KHUNG NHỎ NHẤT

- **Prim's Algorithm**

Input: *G [a connected, weighted graph with n vertices where n is a positive integer]*

Algorithm Body:

[Build a subgraph T of G by starting with any vertex v of G and attaching edges (with their endpoints) one by one to an as-yet-unconnected vertex of G, each time choosing an edge of least weight that is adjacent to a vertex of T.]

1. Pick a vertex v of G and let T be the graph with one vertex, v, and no edges.
2. Let V be the set of all vertices of G except v.
3. **for** $i := 1$ **to** $n - 1$
   - 3a. Find an edge e of G such that (1) e connects T to one of the vertices in V, and (2) e has the least weight of all edges connecting T to a vertex in V. Let w be the endpoint of e that is in V.
   - 3b. Add e and w to the edge and vertex sets of T, and delete w from V.

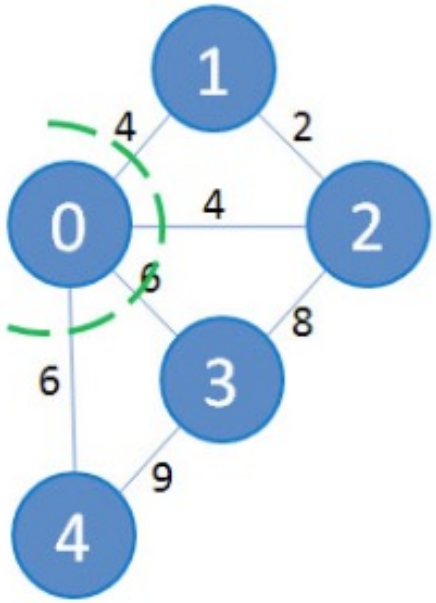
   next i

Output: *T [T is a minimum spanning tree for G.]*

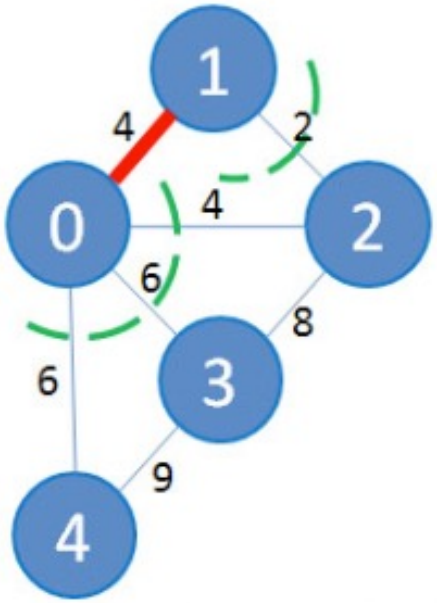
5. CÂY KHUNG NHỎ NHẤT

- **Prim's Algorithm**

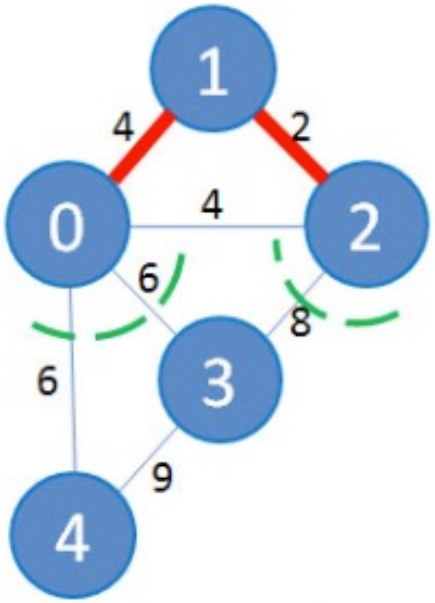
The original graph, start from vertex 0



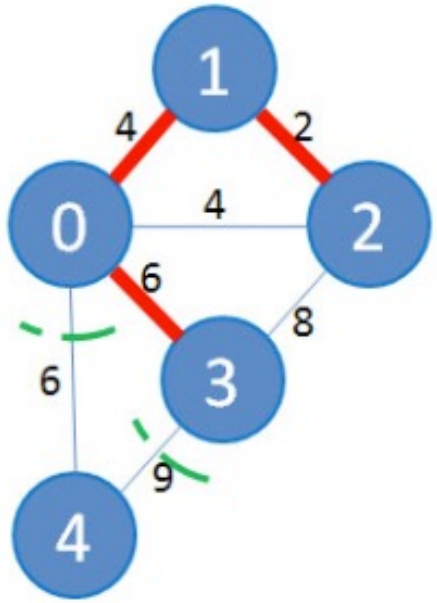
Connect 0 and 1
As this edge is smallest



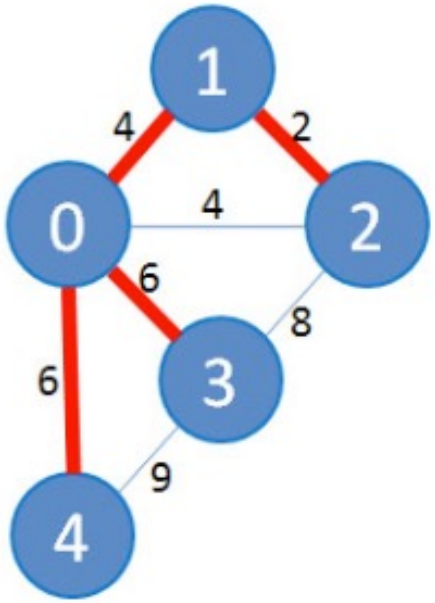
Connect 1 and 2
As this edge is smallest



Connect 0 and 3
As this edge is smallest



Connect 0 and 4
MST is formed



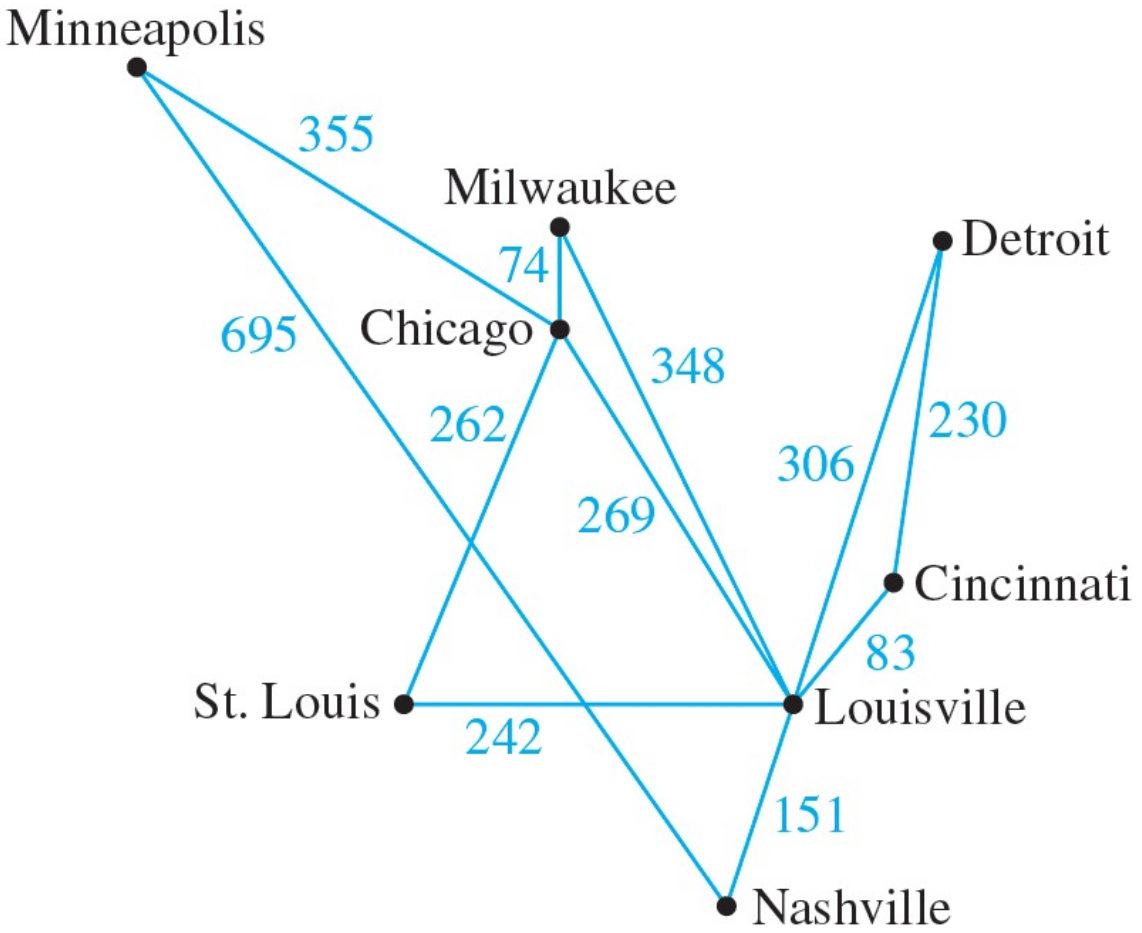
Note: The sorted order of the edges determines how the MST is formed.

Observe that we can also choose to connect vertex 0 and 2 also with weight 4!

Observe that we can also choose to connect vertex 0 and 4 also with weight 6!

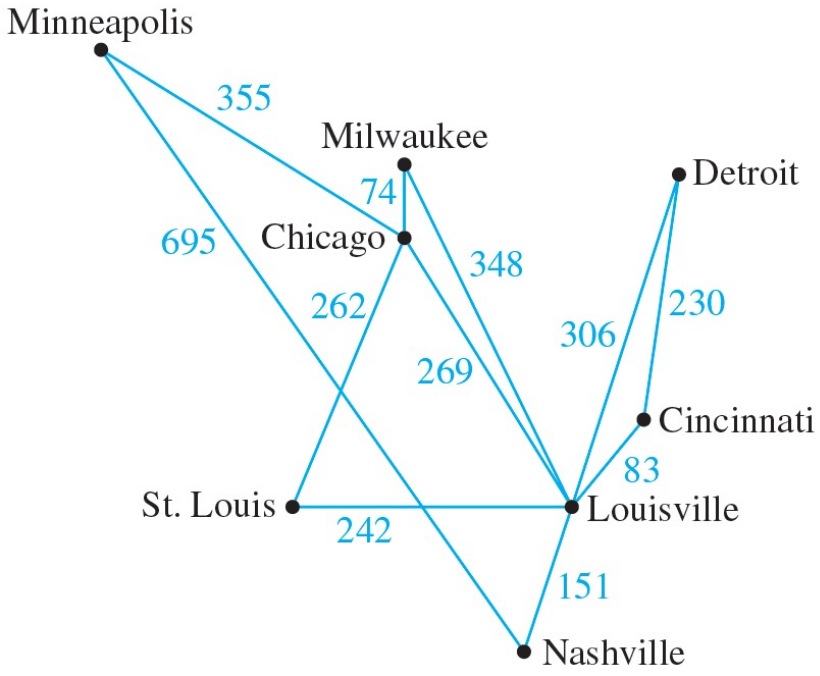
5. CÂY KHUNG NHỎ NHẤT

- Áp dụng thuật toán Prim để xác định cây khung nhỏ nhất của đồ thị sau bieestt đỉnh bắt đầu là Minneapoils



5. CÂY KHUNG NHỎ NHẤT

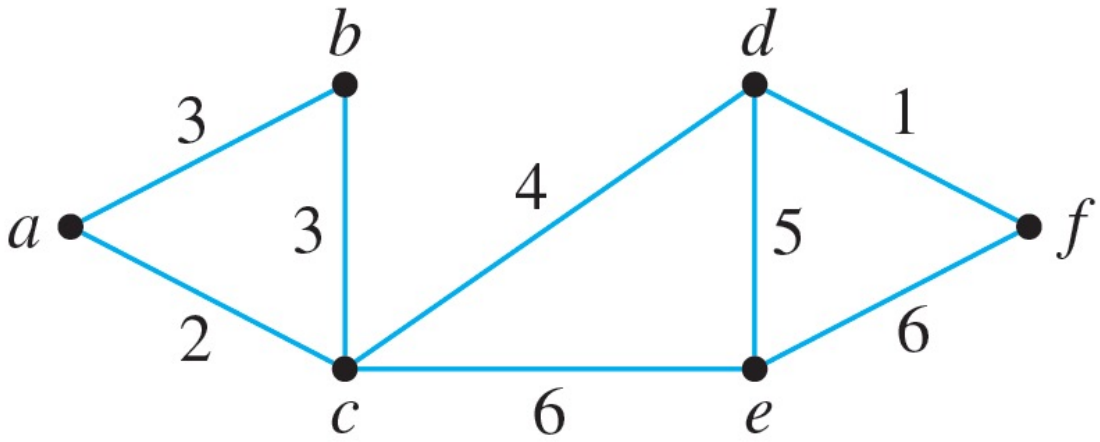
- Áp dụng thuật toán Prim để xác định cây khung nhỏ nhất của đồ thị sau bieestt đỉnh bắt đầu là Minneapoils



Iteration Number	Vertex Added	Edge Added	Weight
0	Minneapolis		
1	Chicago	Minneapolis–Chicago	355
2	Milwaukee	Chicago–Milwaukee	74
3	St. Louis	Chicago–St. Louis	262
4	Louisville	St. Louis–Louisville	242
5	Cincinnati	Louisville–Cincinnati	83
6	Nashville	Louisville–Nashville	151
7	Detroit	Cincinnati–Detroit	230

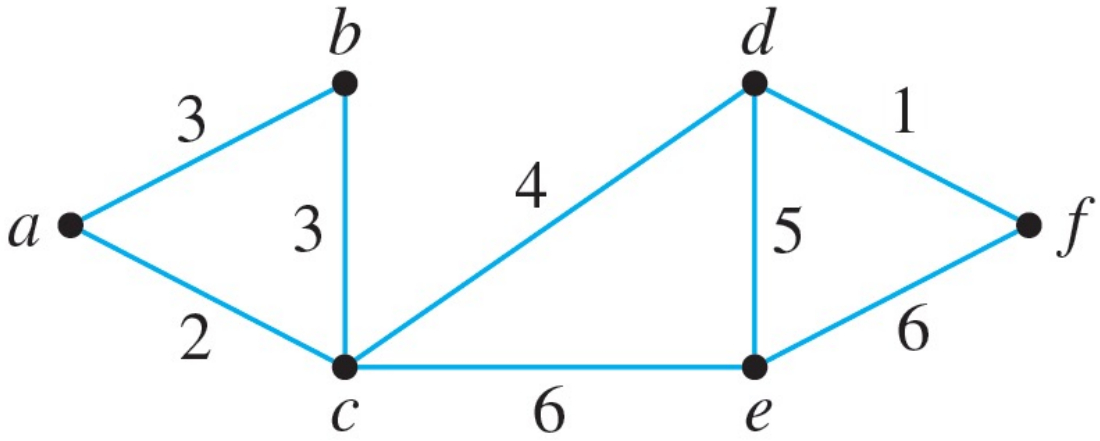
5. CÂY KHUNG NHỎ NHẤT

- Tìm tất cả các cây khung nhỏ nhất cho đồ thị sau. Sử dụng thuật toán Kruskal và thuật toán Prim bắt đầu từ đỉnh a. Cho biết thứ tự các cạnh được thêm vào để tạo thành mỗi cây.

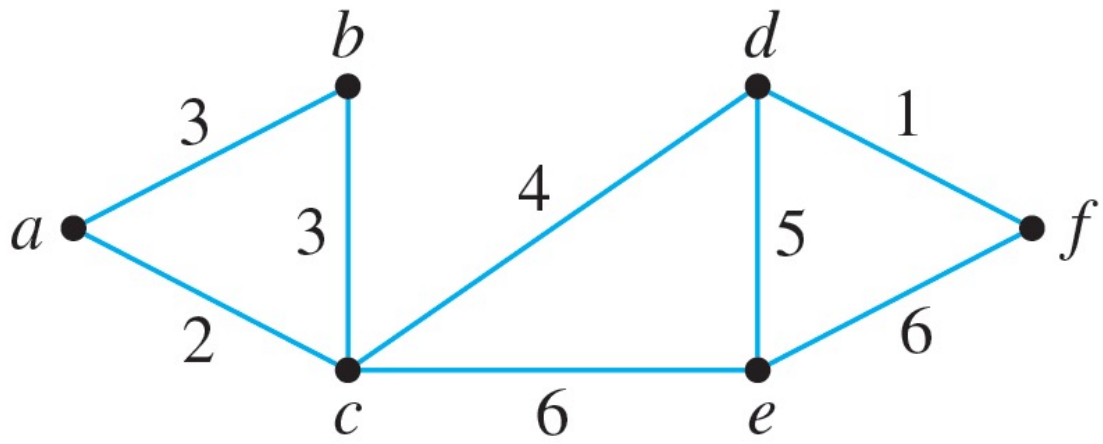


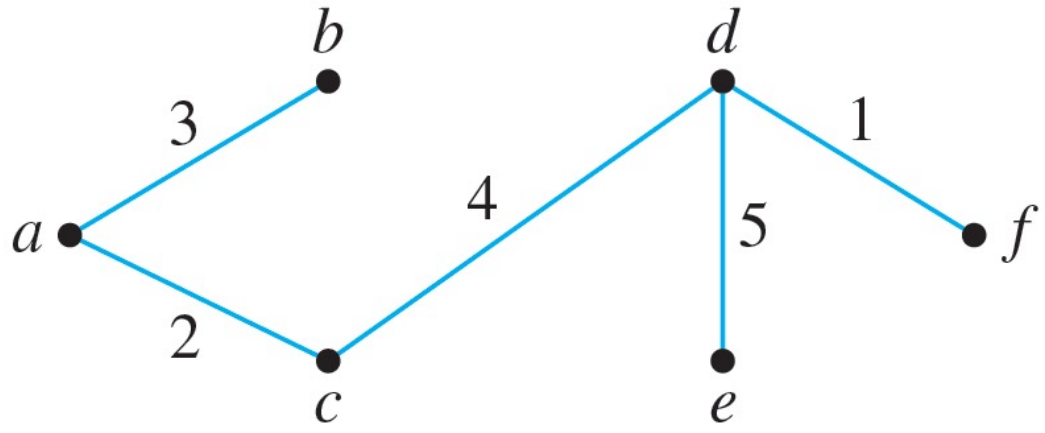
5. CÂY KHUNG NHỎ NHẤT

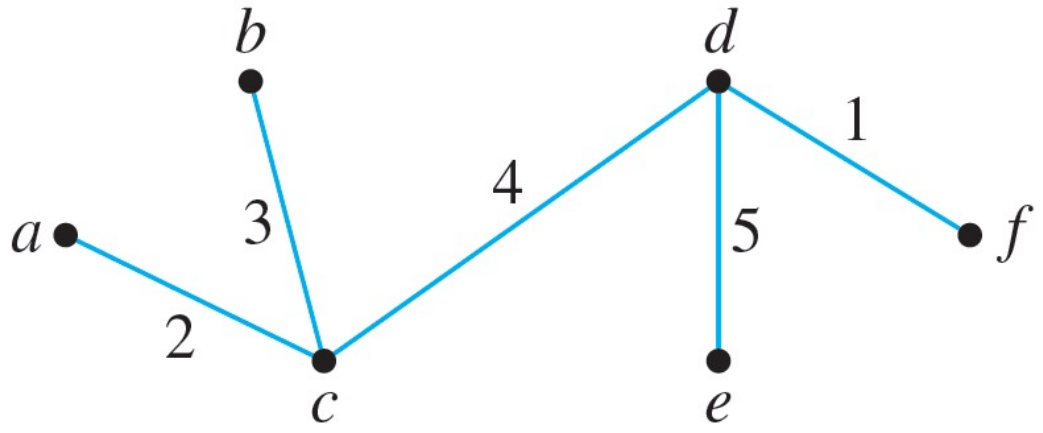
- Khi thuật toán của Kruskal được áp dụng, các cạnh được thêm vào theo một trong hai thứ tự sau:
    1. {d, f}, {a, c}, {a, b}, {c, d}, {d, e}
    2. {d, f}, {a, c}, {b, c}, {c, d}, {d, e}
- Khi thuật toán của Prim được áp dụng bắt đầu từ a, các cạnh được thêm vào một trong hai thứ tự sau:
    1. {a, c}, {a, b}, {c, d}, {d, f}, {d, e}
    2. {a, c}, {b, c}, {c, d}, {d, f}, {d, e}



5. CÂY KHUNG NHỎ NHẤT







THANK YOU FOR YOUR ATTENTION.