TOÁN HỌC TỔ HỢP

Chương 2.

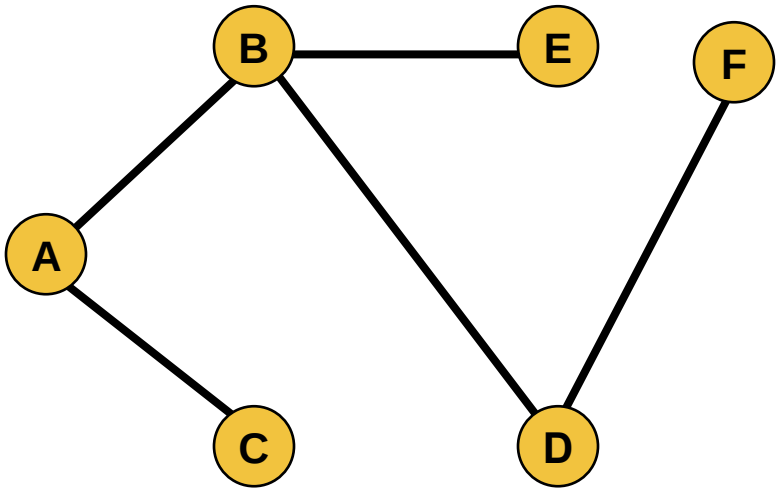
CÂY

Nội dung

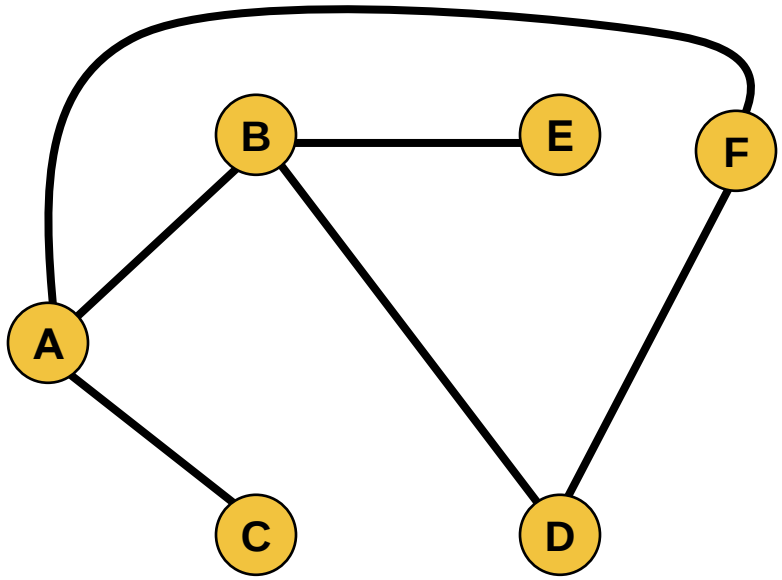
- Định nghĩa và tính chất
- Cây khung ngắn nhất
- Cây có gốc
- Phép duyệt cây

1. Định nghĩa và tính chất

Định nghĩa. **Cây** (tree) là đồ thị vô hướng, liên thông và không có chu trình



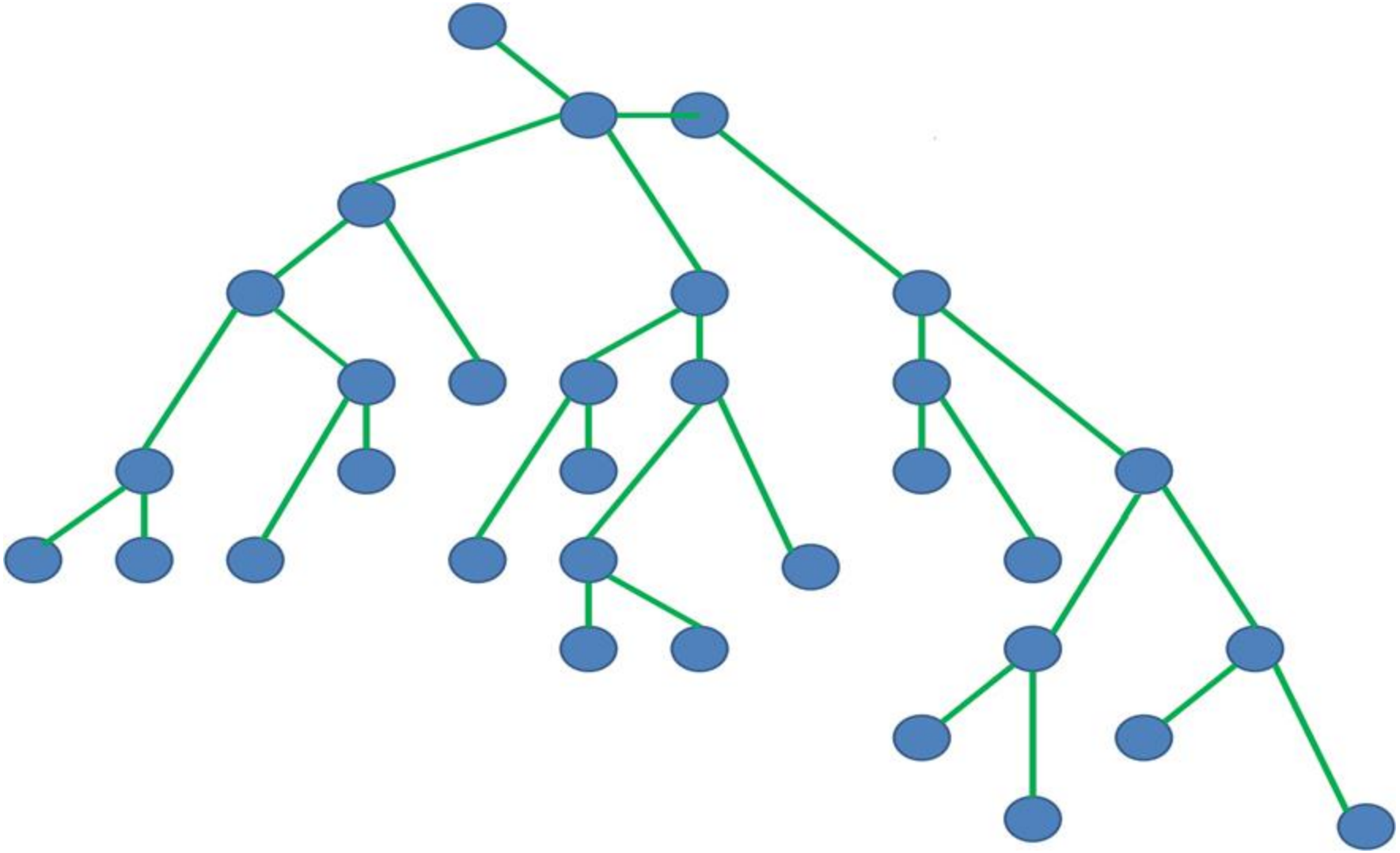
G1



G2

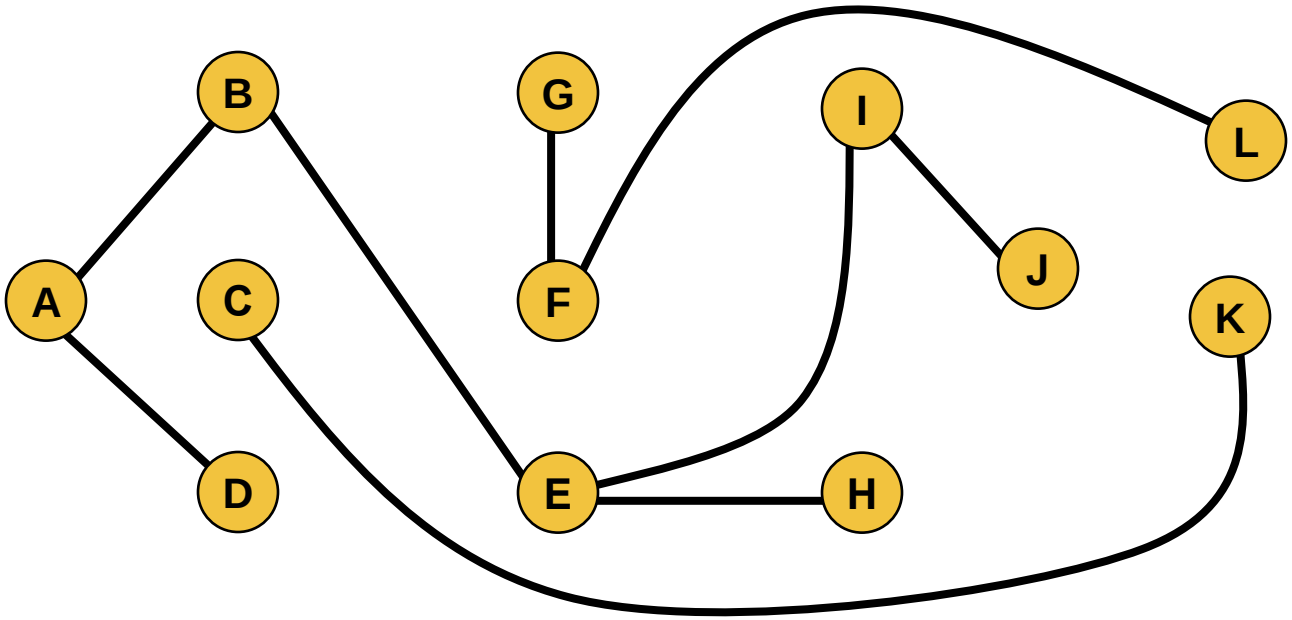
G1 là cây, G2 không phải cây

Cây



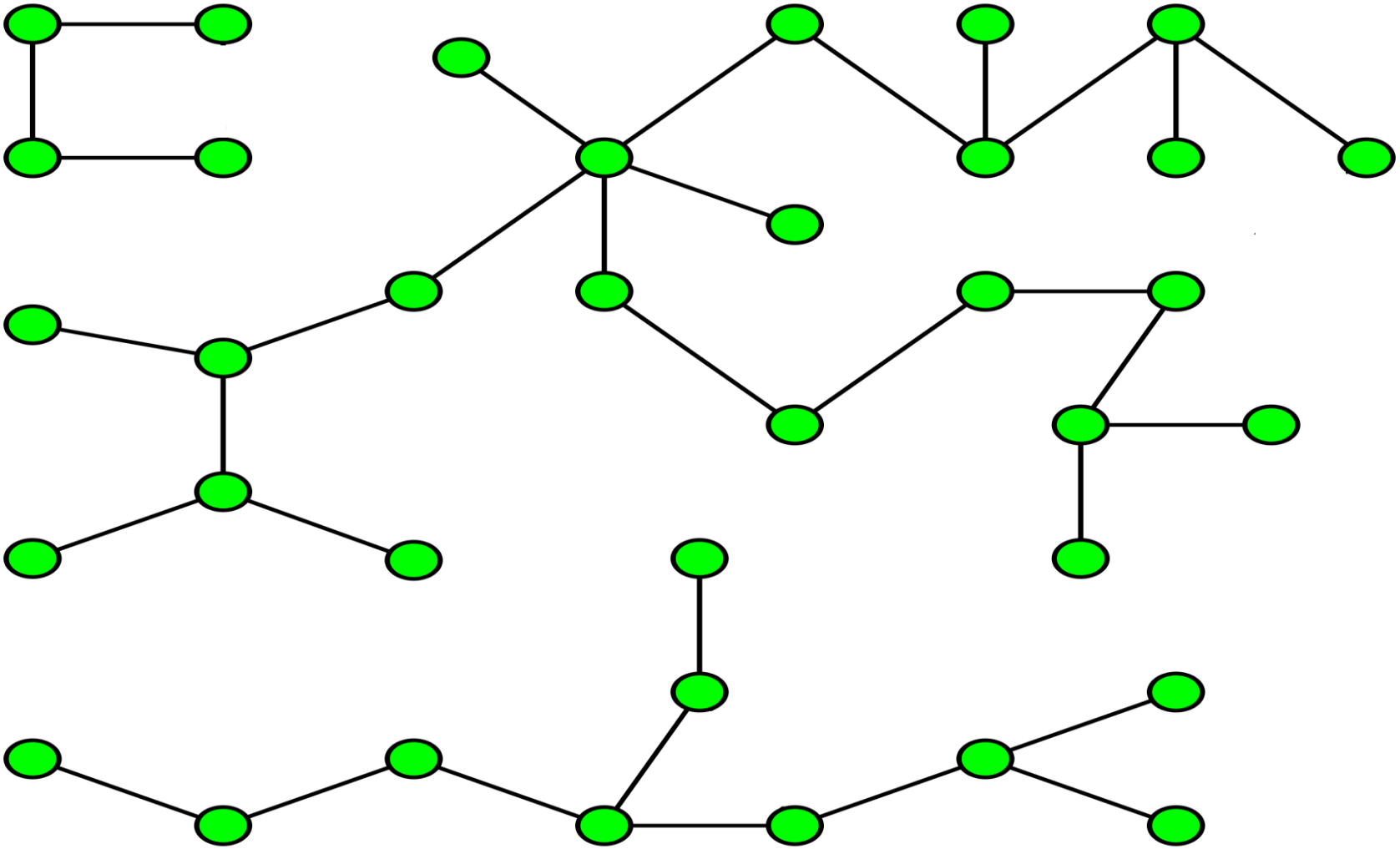
Rừng

Định nghĩa. **Rừng** (forest) là đồ thị vô hướng không có chu trình



Nhận xét. Rừng là đồ thị mà mỗi thành phần liên thông của nó là một cây.

Rừng



Tính chất của cây

Định lý: Cho đồ thị vô hướng T có n đỉnh. Khi đó các phát biểu sau là tương đương:

1) T là 1 cây
2) T không chứa chu trình và có n-1 cạnh
3) T liên thông và có n-1 cạnh
4) T liên thông và mỗi cạnh của nó đều là cầu
5) Giữa hai đỉnh bất kỳ của T có đúng một đường đi nối chúng với nhau
6) T không chứa chu trình nhưng khi thêm vào một cạnh nối hai đỉnh của T ta thu được đúng một chu trình

Hệ quả.

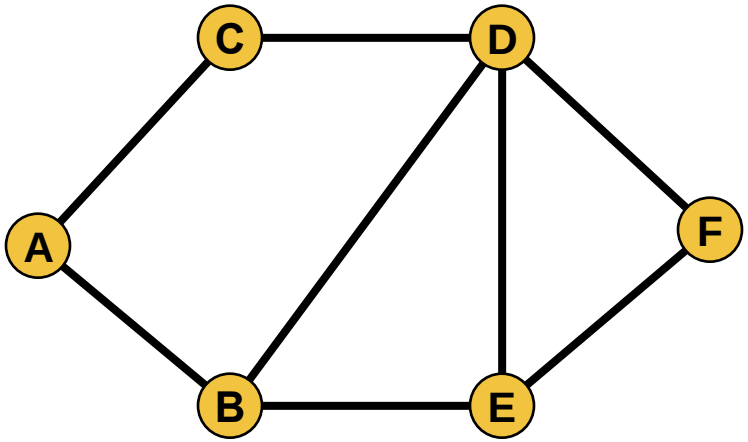
a) Một cây có ít nhất 2 đỉnh treo

b) Nếu G là một rừng có n đỉnh và có p cây thì số cạnh của G là **m=n-p**

Cây khung của đồ thị

Định nghĩa: Một cây T được gọi là **cây khung** (hay cây tối đại, cây bao trùm) của đồ thị G=(V, E) nếu T là đồ thị con của G và chứa tất cả các đỉnh của G.

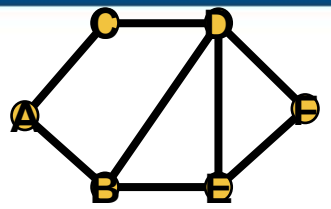
Ví dụ. Cho đồ thị

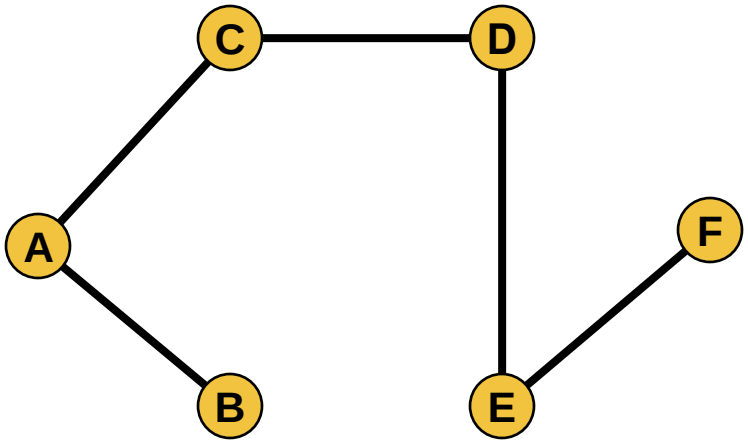


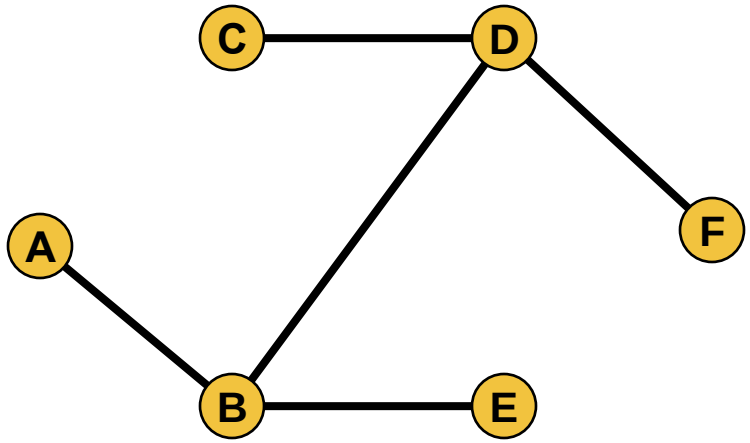
Hãy tìm cây khung của G?

Cây khung của đồ thị

Đáp án. Một số cây khung của G



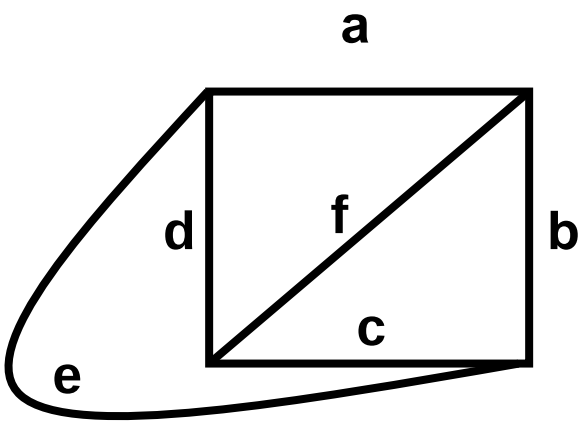




Nhận xét. Với 1 đồ thị cho trước, có thể có vài cây khung của đồ thị đó

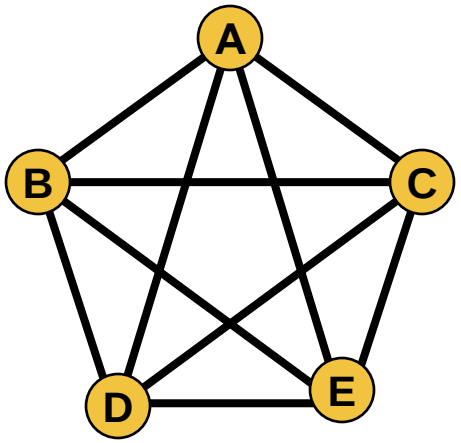
Định lý. Mọi đồ thị liên thông đều có cây khung

Định lý (Cayley) Số cây khung của đồ thị K_n là n^{n-2}



Số cây khung $4^{4-2}=16$

Ví dụ: abc, bcd, cda, dab, abf, acf, bdf, ...



Số cây khung $5^{5-2}=125$

Tìm một cây khung của đồ thị

Bài toán: Cho G là đồ thị vô hướng liên thông, hãy tìm 1 cây khung của đồ thị G.

Để giải bài này ta dùng 2 thuật toán sau

- Thuật toán tìm kiếm theo chiều rộng (**BFS**)
  Breadth-first search
- Thuật toán tìm kiếm theo chiều sâu (**DFS**)
  Depth-first search

Tìm kiếm theo chiều rộng (BFS)

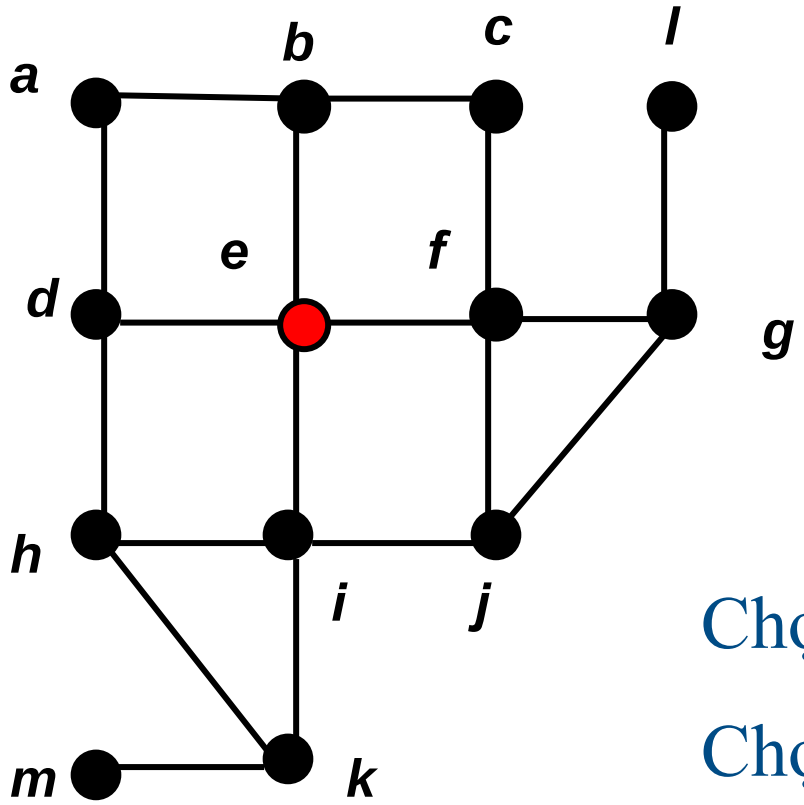
Cho G là đồ thị liên thông với tập đỉnh $\{v_1, v_2, \ldots, v_n\}$

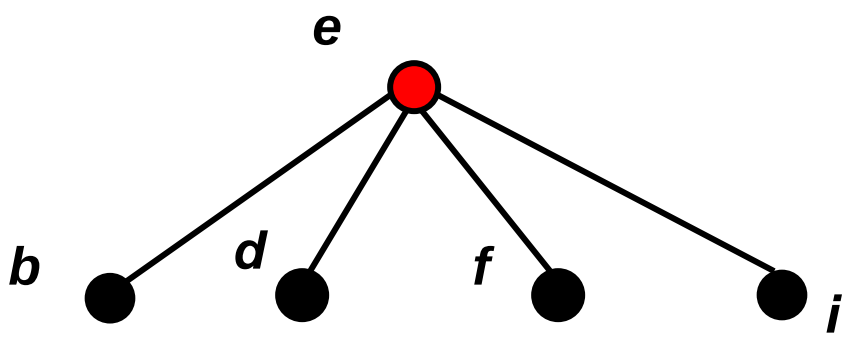
- Bước 0: thêm v_1 như là gốc của cây rỗng.
- Bước 1: thêm vào các đỉnh kề v_1 và các cạnh nối v_1 với chúng. Những đỉnh này là đỉnh mức 1 trong cây.
- Bước 2: đối với mọi đỉnh v mức 1, thêm vào các cạnh kề với v vào cây sao cho không tạo nên chu trình. Ta thu được các đỉnh mức 2.

……

Tiếp tục quá trình này cho tới khi tất cả các đỉnh của đồ thị được ghép vào cây. Cây T có được là cây khung của đồ thị.

Ví dụ. Tìm một cây khung của đồ thị *G*.

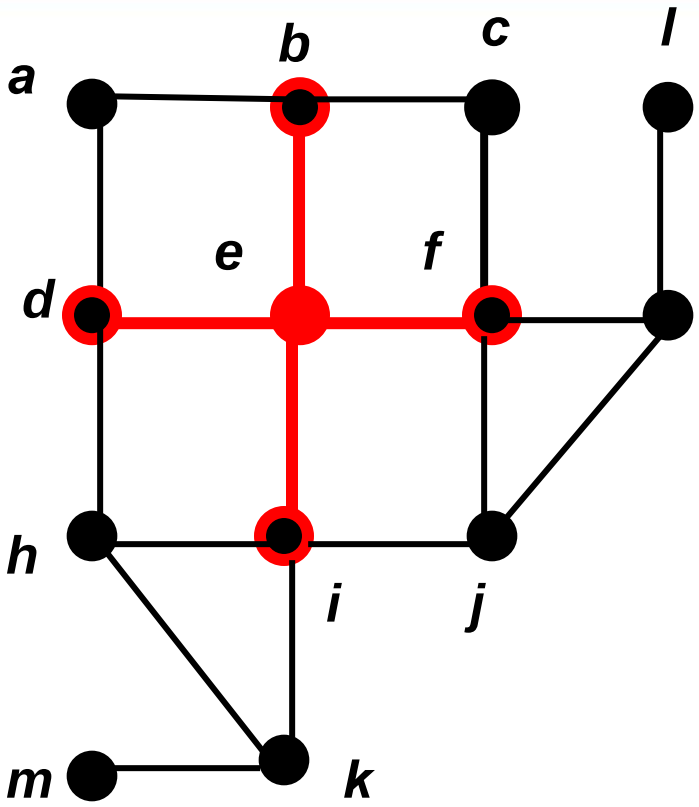


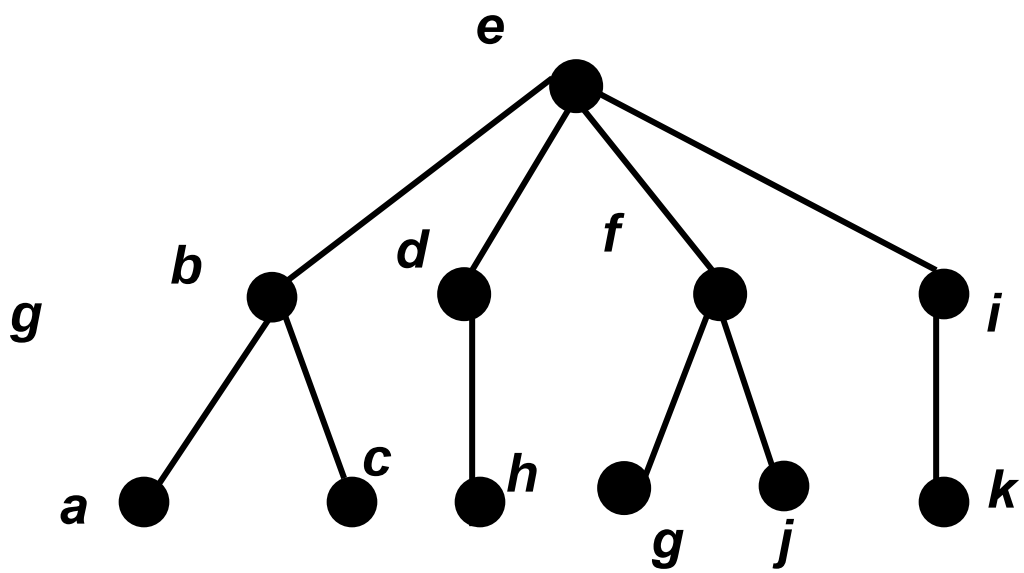


Chọn *e* làm gốc

Chọn các đỉnh kề với *e*.

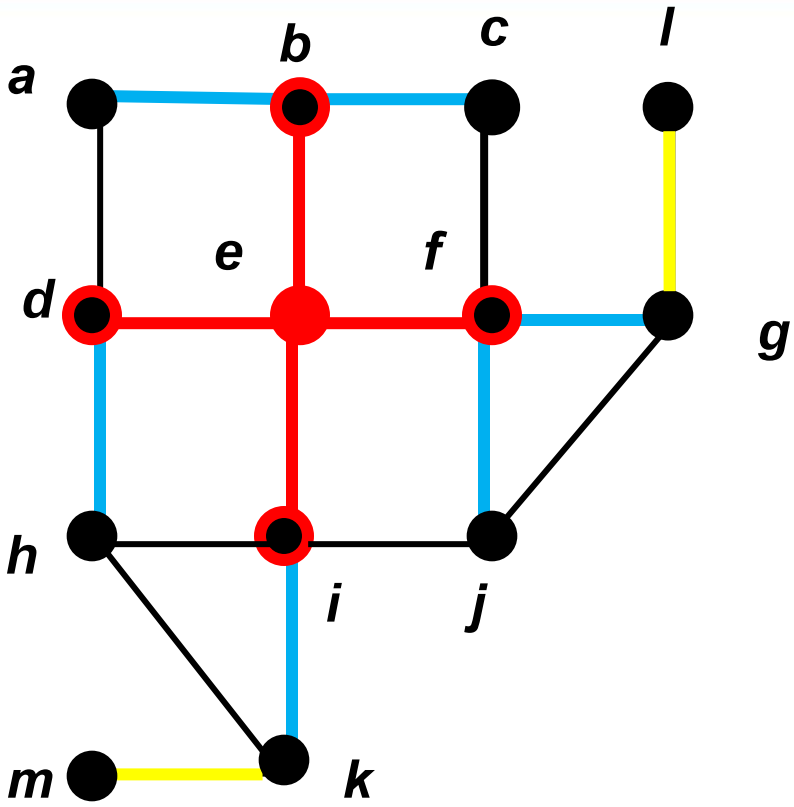
Các đỉnh mức 1 là: *b, d, f, i*

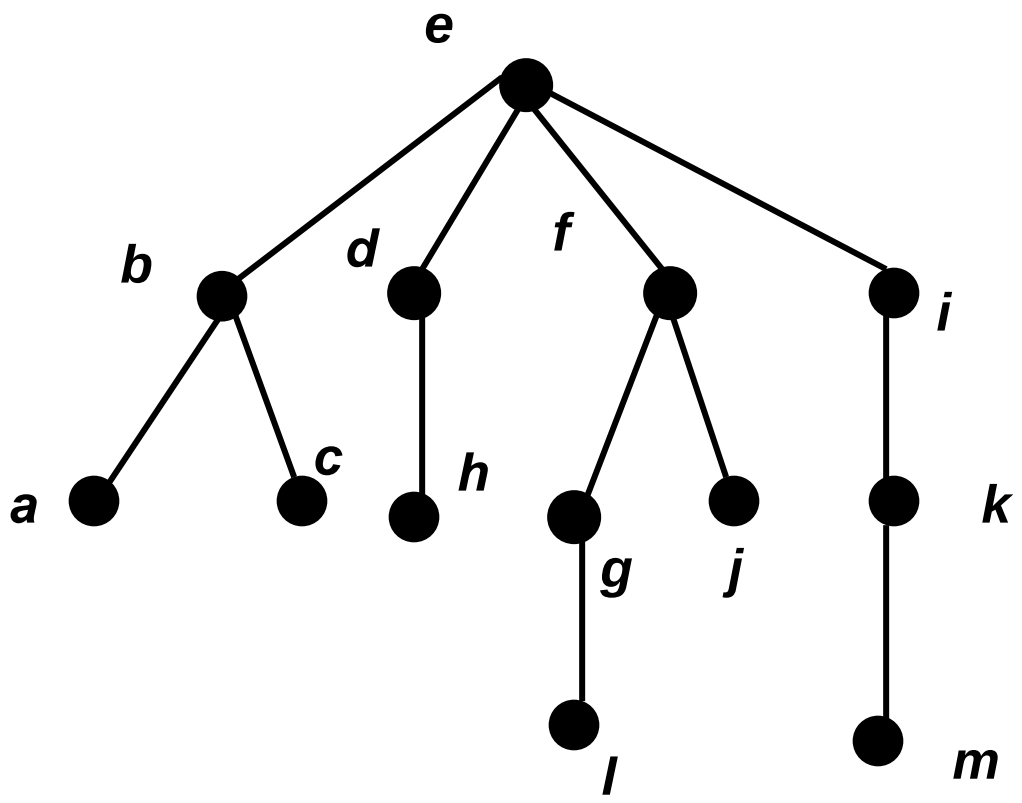




- Thêm *a* và *c* làm con của *b*,
- *h* là con duy nhất của *d*,
- *g* và *j* là con của *f*,
- *k* là con duy nhất của *i*,

Các đỉnh mức 2 là: *a*, *c*, *h*, *g*, *j*, *k*



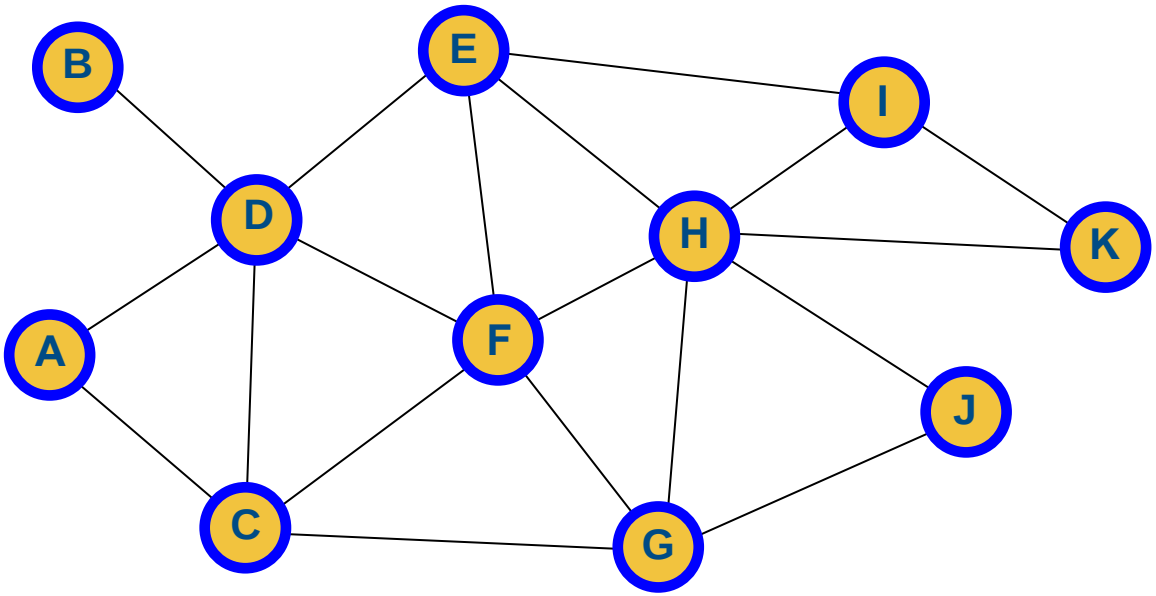


Cuối cùng thêm *l* và *m* là con của *g* và *k* tương ứng

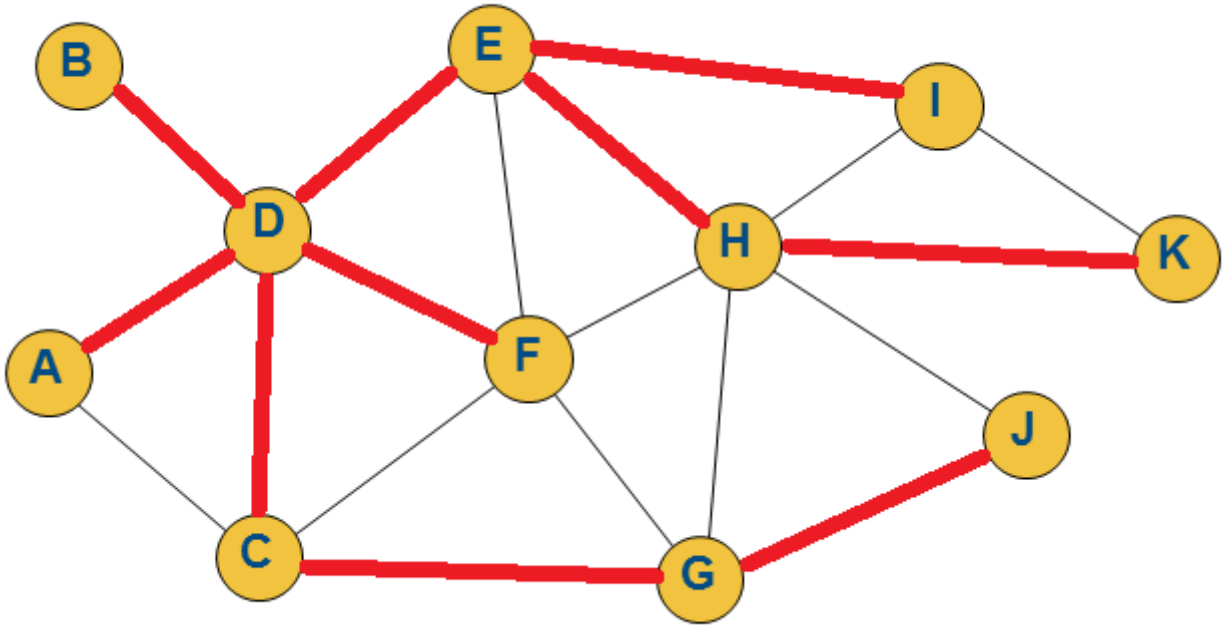
Các đỉnh mức 3 là: *l*, *m*

Ta có được cây khung cần tìm

Ví dụ. Tìm cây khung của đồ thị bằng thuật toán BFS với D là đỉnh bắt đầu



Đáp án.



Tìm kiếm theo chiều sâu (DFS)

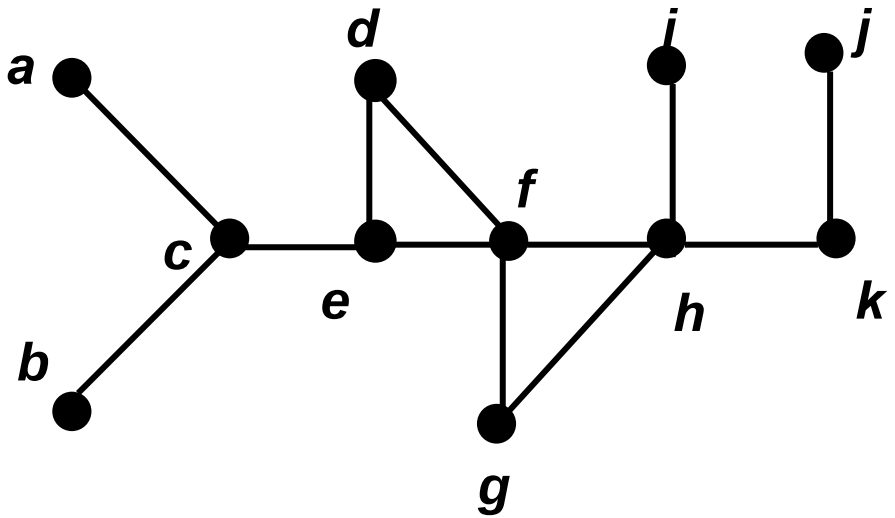
Cho G là đồ thị liên thông với tập đỉnh $\{v_1, v_2, \ldots, v_n\}$

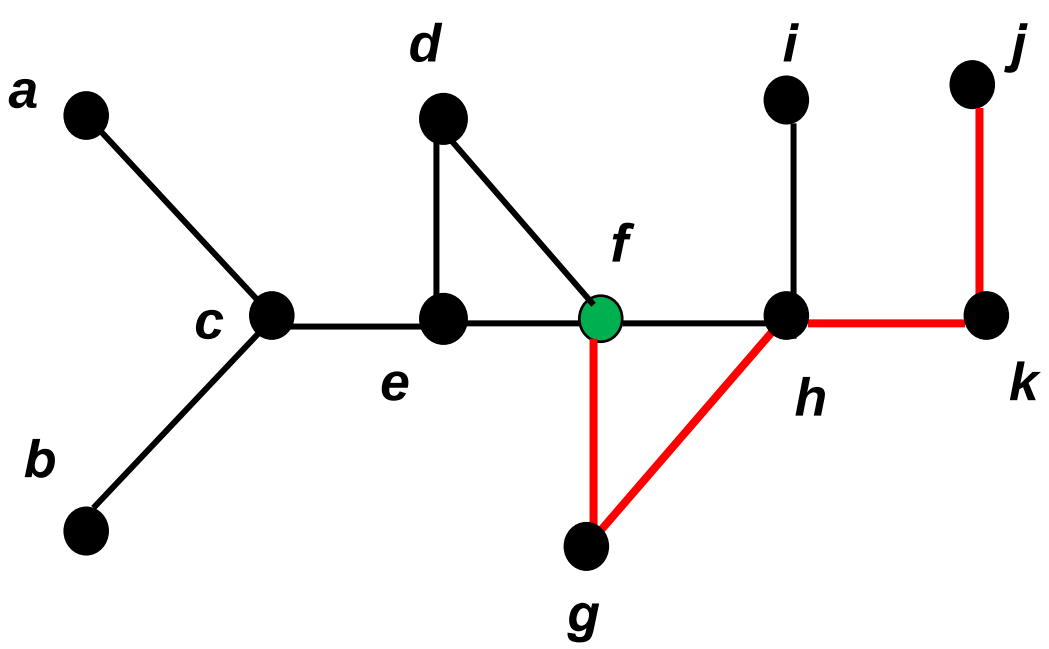
- Chọn một đỉnh tùy ý của đồ thị làm gốc.
- Xây dựng đường đi từ đỉnh này bằng cách lần lượt ghép các cạnh sao cho mỗi cạnh mới ghép sẽ nối đỉnh cuối cùng trên đường đi với một đỉnh còn chưa thuộc đường đi. Tiếp tục ghép thêm cạnh vào đường đi chừng nào không thể thêm được nữa.
- Nếu đường đi qua tất cả các đỉnh của đồ thị thì cây do đường đi này tạo nên là cây khung.

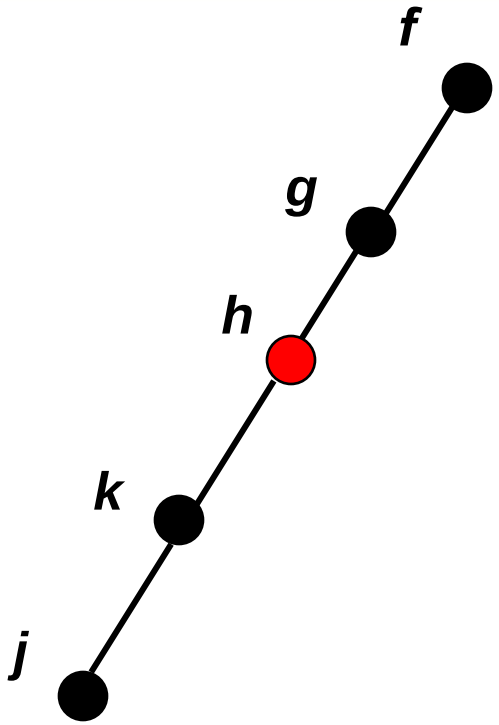
- Nếu chưa thì lùi lại đỉnh trước đỉnh cuối cùng của đường đi và xây dựng đường đi mới xuất phát từ đỉnh này đi qua các đỉnh còn chưa thuộc đường đi.

  Nếu điều đó không thể làm được thì lùi thêm một đỉnh nữa trên đường đi và thử xây dựng đường đi mới. Tiếp tục quá trình như vậy cho đến khi tất cả các đỉnh của đồ thị được ghép vào cây. Cây *T* có được là cây khung của đồ thị.

Ví dụ. Tìm một cây khung của đồ thị với f là đỉnh gốc

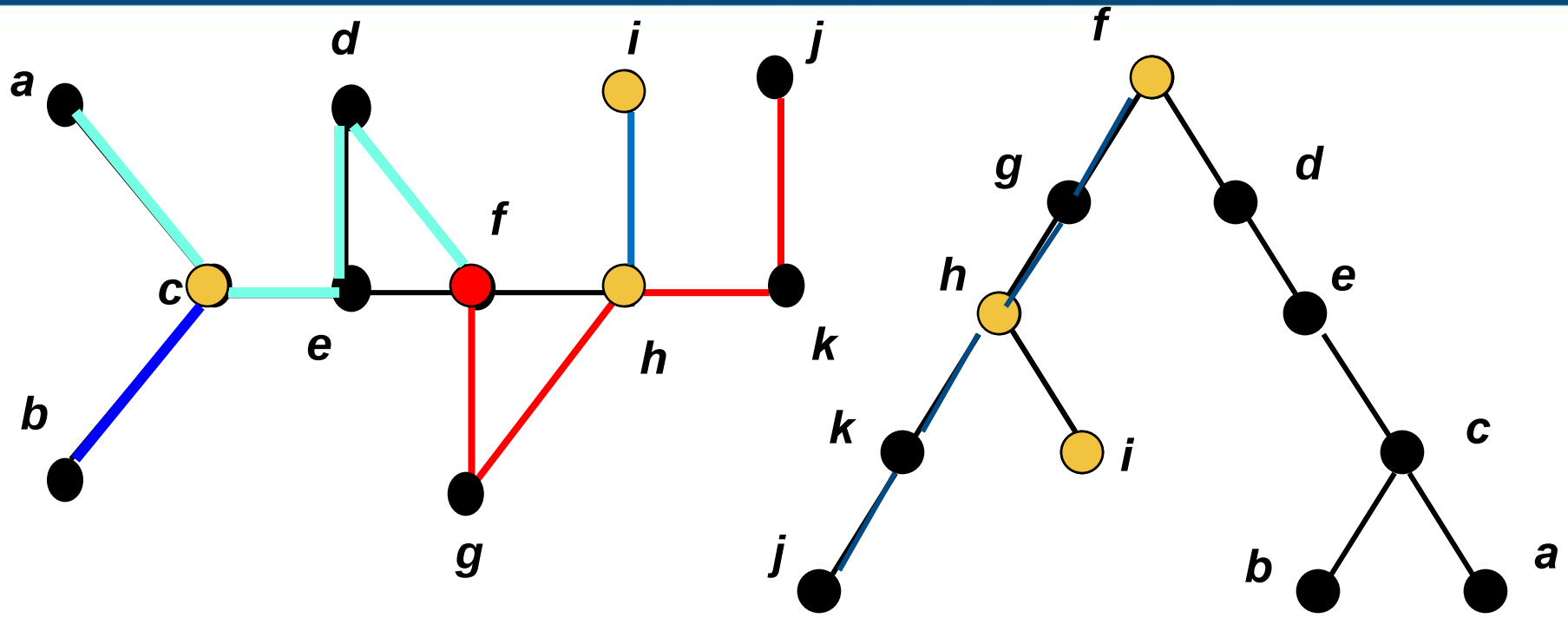






Thêm các hậu duệ của *f* : *g*, *h*, *k*, *j*

Lùi về *k* không thêm được cạnh nào, tiếp tục lùi về *h*



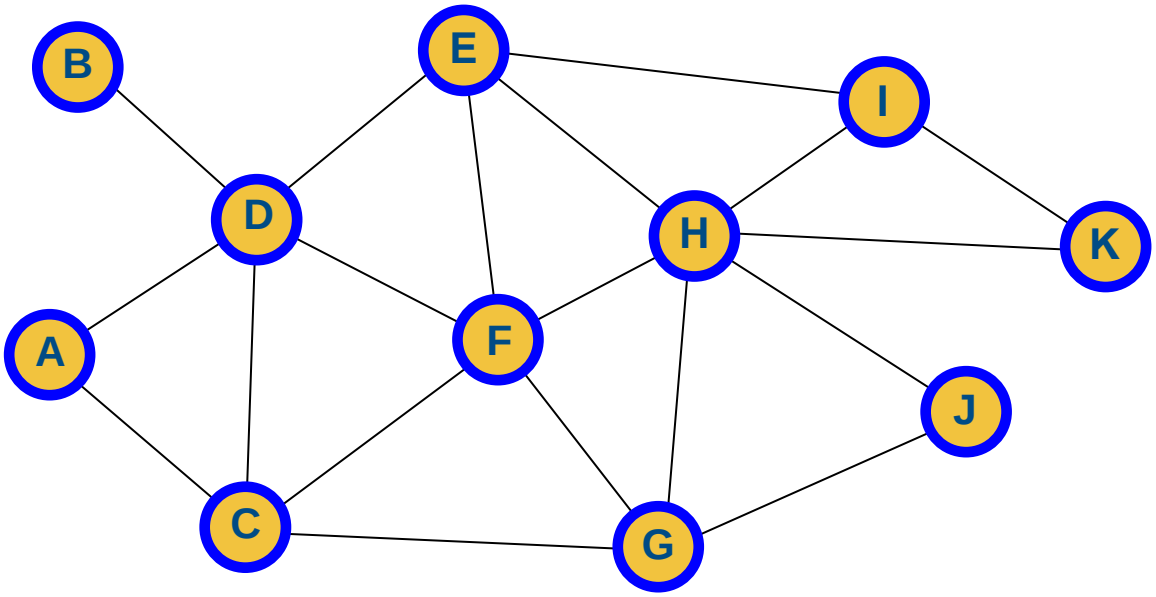
Thêm *i* làm con thứ hai của *h* và lùi về *f*.

Lại thêm các hậu duệ của *f* : *d*, *e*, *c*, *a*

Lùi về *c* và thêm *b* làm con thứ hai của nó .

Cây thu được là cây khung của đồ thị đã cho

Ví dụ. Tìm một cây khung của đồ thị bằng thuật toán DFS với A là đỉnh bắt đầu



Đồ thị có trọng số

Định nghĩa. Đồ thị G = (V,E) gọi là đồ thị có ***trọng số*** (hay chiều dài, trọng lượng) nếu mỗi cạnh e được gán với một số thực w(e). Ta gọi **w(e)** là ***trọng lượng*** của e.

- ***Độ dài*** của đường đi từ u đến v bằng tổng trọng lượng các cạnh mà đường đi qua.
- **Trọng lượng** của một cây T của G bằng với tổng trọng lượng các cạnh trong cây
- **Cây khung ngắn nhất** là cây khung có trọng lượng nhỏ nhất của G.

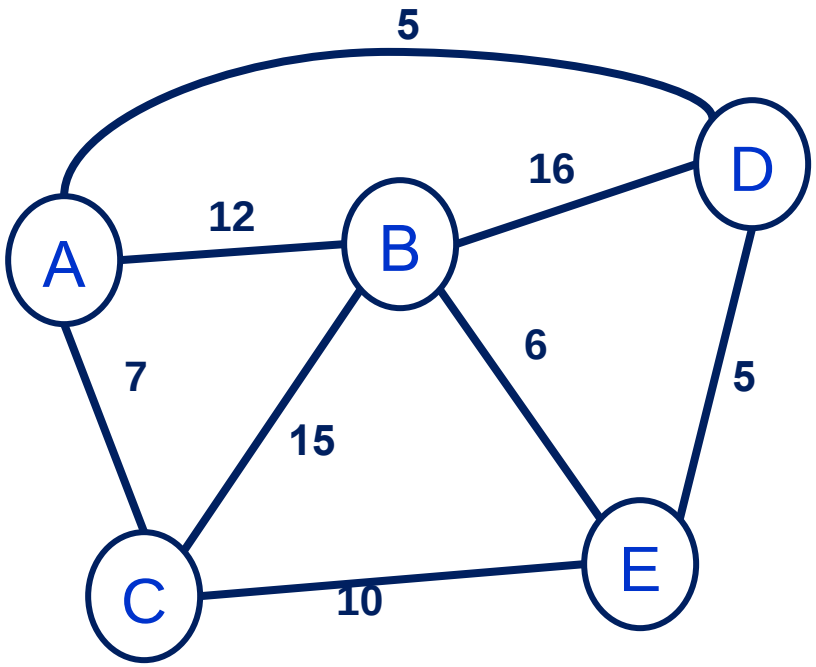
Ma trận khoảng cách (trọng số)

Định nghĩa. Cho G = (V, E), V = $\{v_1, v_2, \ldots, v_n\}$ là đơn đồ thị có trọng số. **Ma trận khoảng cách** của G là ma trận D= (d_{ij}) được xác định như sau:

$$d_{ij} = \begin{cases} 0 & khi \quad i = j \\ w(v_i v_j) & khi \quad v_i v_j \in E \\ \infty & khi \quad v_i v_j \notin E \end{cases}$$

Ma trận khoảng cách

Ví dụ. Tìm ma trận khoảng cách của đồ thị sau



$$\begin{bmatrix} 0 & 12 & 7 & 5 & \infty \\ 12 & 0 & 15 & 16 & 6 \\ 7 & 15 & 0 & \infty & 10 \\ 5 & 16 & \infty & 0 & 5 \\ \infty & 6 & 10 & 5 & 0 \end{bmatrix}$$

Thuật toán tìm cây khung ngắn nhất

Có nhiều thuật toán xây dựng cây khung ngắn nhất:

- Thuật toán Boruvka
- Thuật toán Kruskal
- Thuật toán Jarnik – Prim
- Phương pháp Dijkstra
- Thuật toán Cheriton – Tarjan
- Thuật toán Chazelle
- …

Thuật toán Kruskal

Input: Đồ thị G=(X, E) liên thông, X gồm n đỉnh

Output: Cây khung ngắn nhất T=(V, U) của G

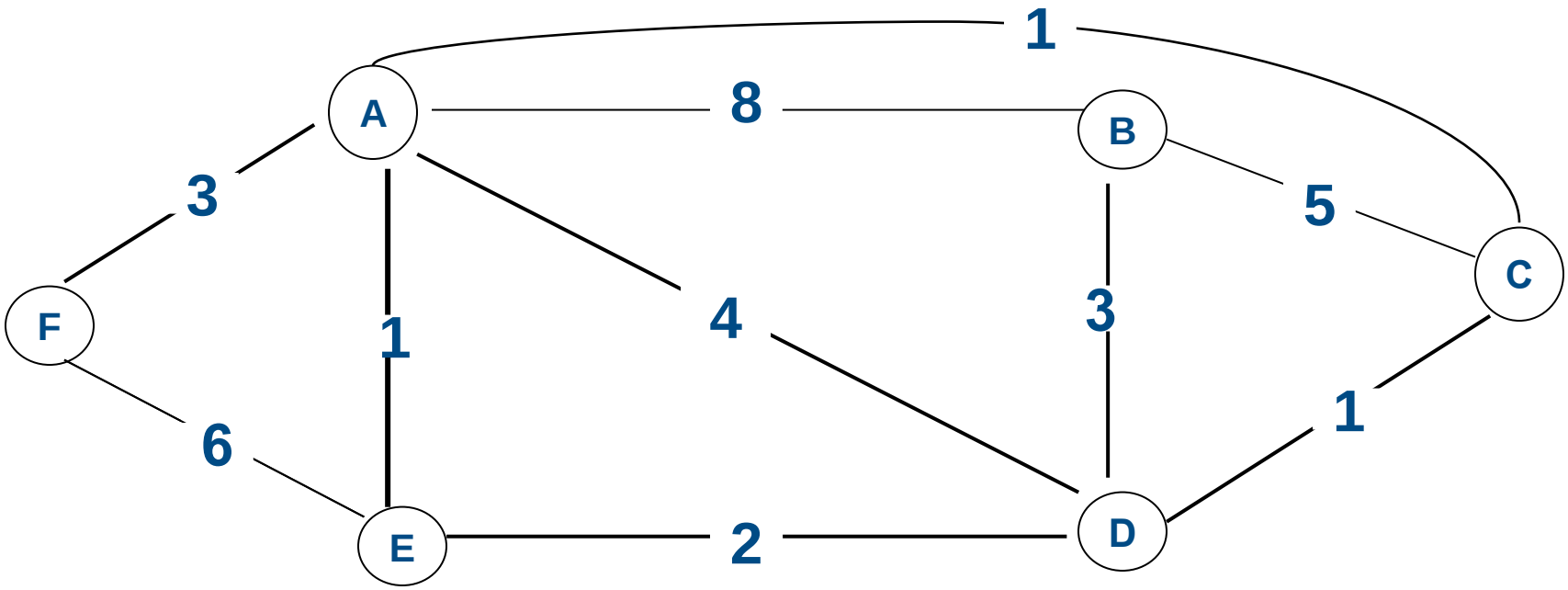
Bước 1. Sắp xếp các cạnh trong G tăng dần theo trọng lượng; khởi tạo T := ∅.

Bước 2. Lần lượt lấy từng cạnh e thuộc danh sách đã sắp xếp. Nếu T+{e} không chứa chu trình thì thêm e vào T: T := T+{e}.

Bước 3. Nếu T đủ n-1 cạnh thì dừng; ngược lại, lặp bước 2.

Thuật toán Kruskal

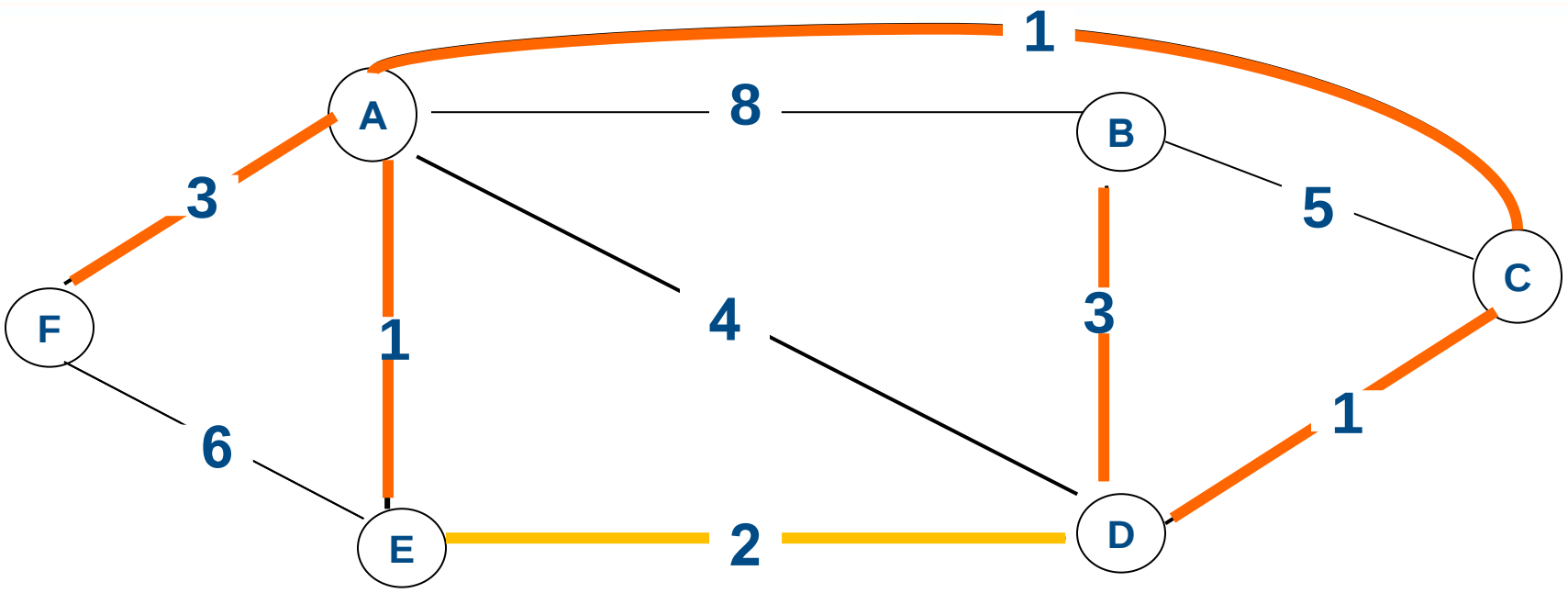
Ví dụ. Tìm cây khung ngắn nhất của đồ thị sau



Giải. Sắp xếp các cạnh

AC	AE	CD	DE	AF	BD	AD	BC	EF	AB
1	1	1	2	3	3	4	5	6	8

Thuật toán Kruskal

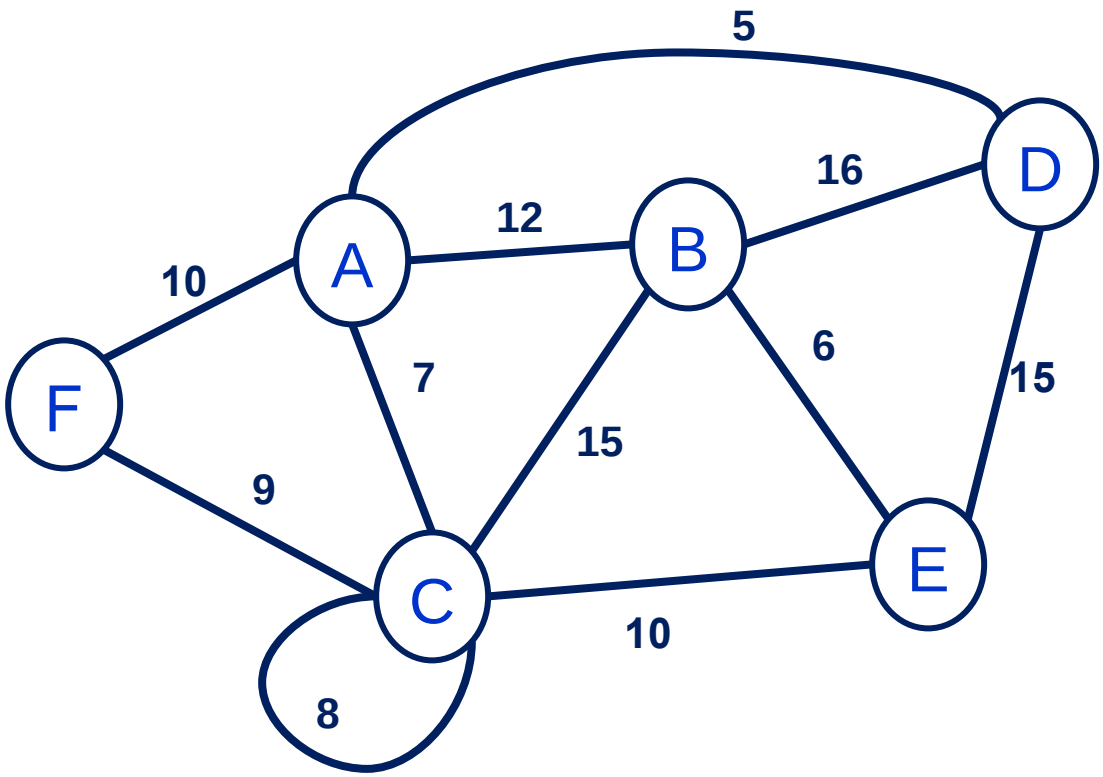


AC	AE	CD	DE	AF	BD	AD	BC	EF	AB
1	1	1	2	3	3	4	5	6	8

Như vậy **T = { AC, AE, CD, AF, BD }** là khung ngắn nhất với trọng lượng: 9

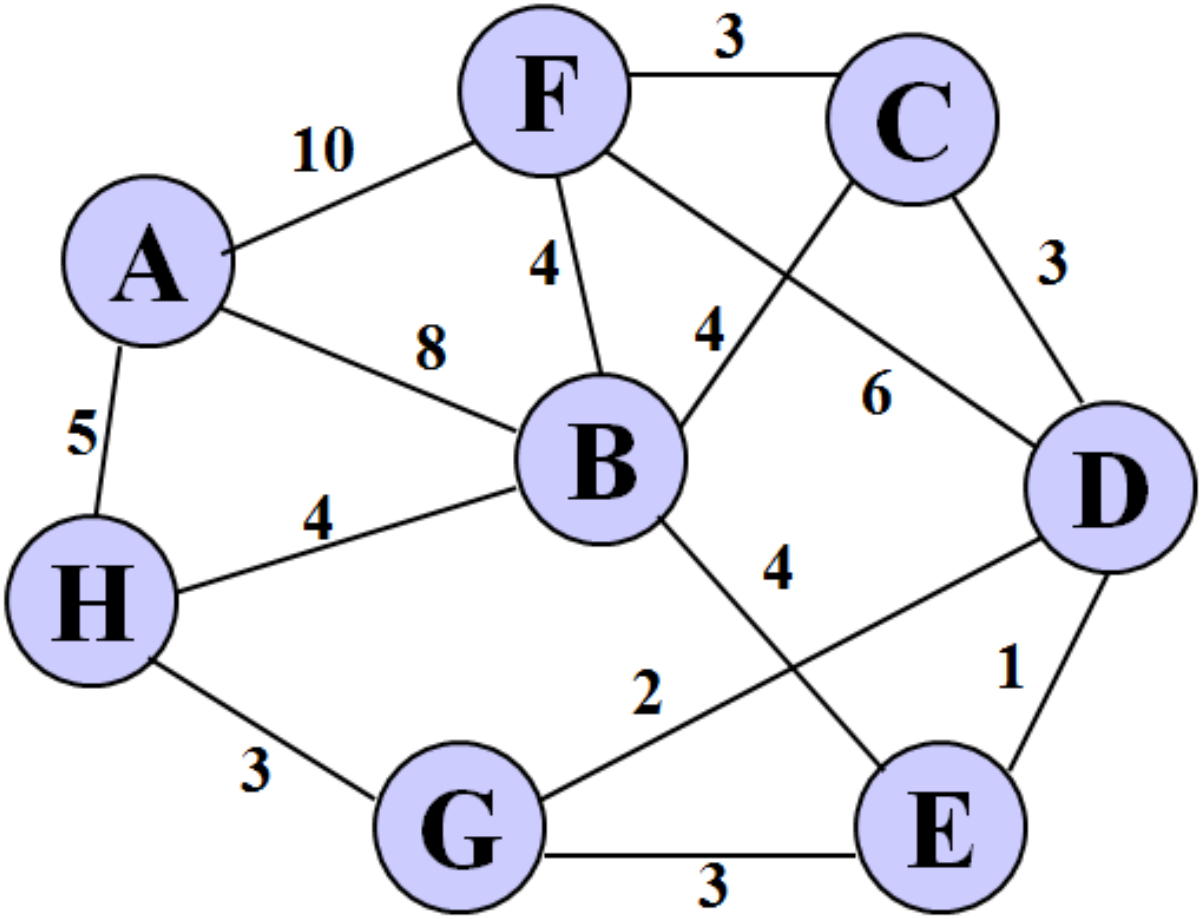
Thuật toán Kruskal

Ví dụ. Tìm cây khung ngắn nhất của đồ thị sau



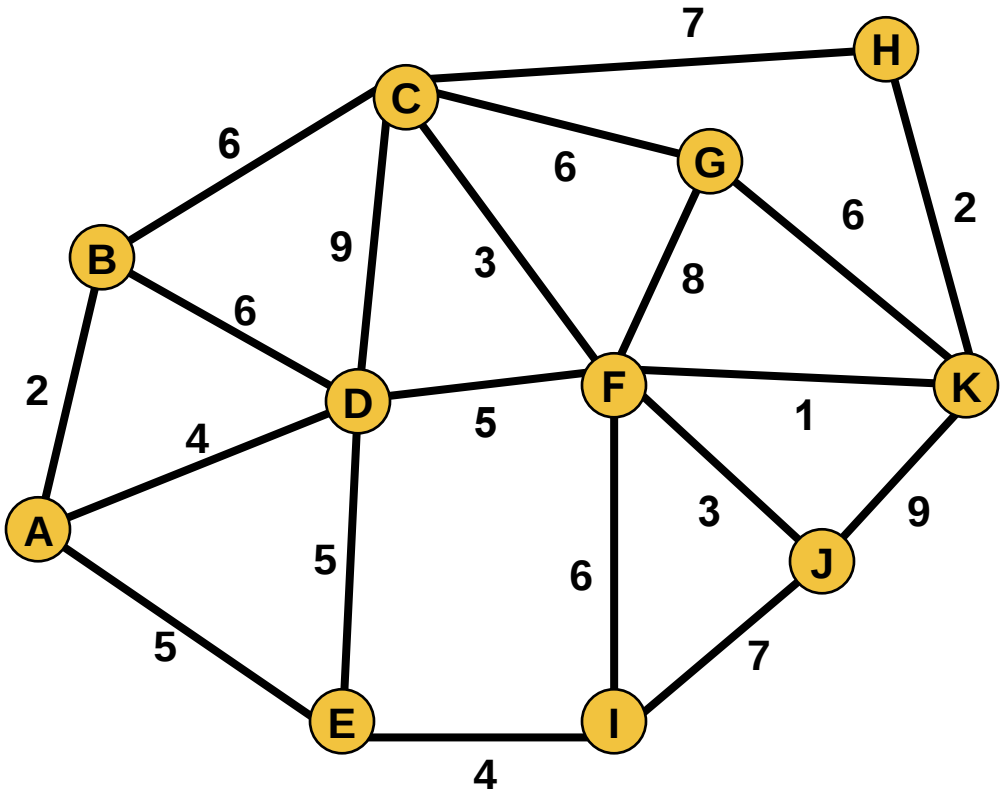
Thuật toán Kruskal

Ví dụ. Tìm cây khung ngắn nhất của đồ thị sau



Thuật toán Kruskal

Ví dụ. Dùng thuật toán Kruskal để tìm cây khung nhỏ nhất của đồ thị sau:



Thuật toán Prim

Input: Đồ thị liên thông G=(X, E), X gồm n đỉnh

Output: Cây khung ngắn nhất T=(V, U) của G

Bước 1. Chọn tùy ý v $\in$ X và khởi tạo **V** := { v }; U := $\varnothing$;

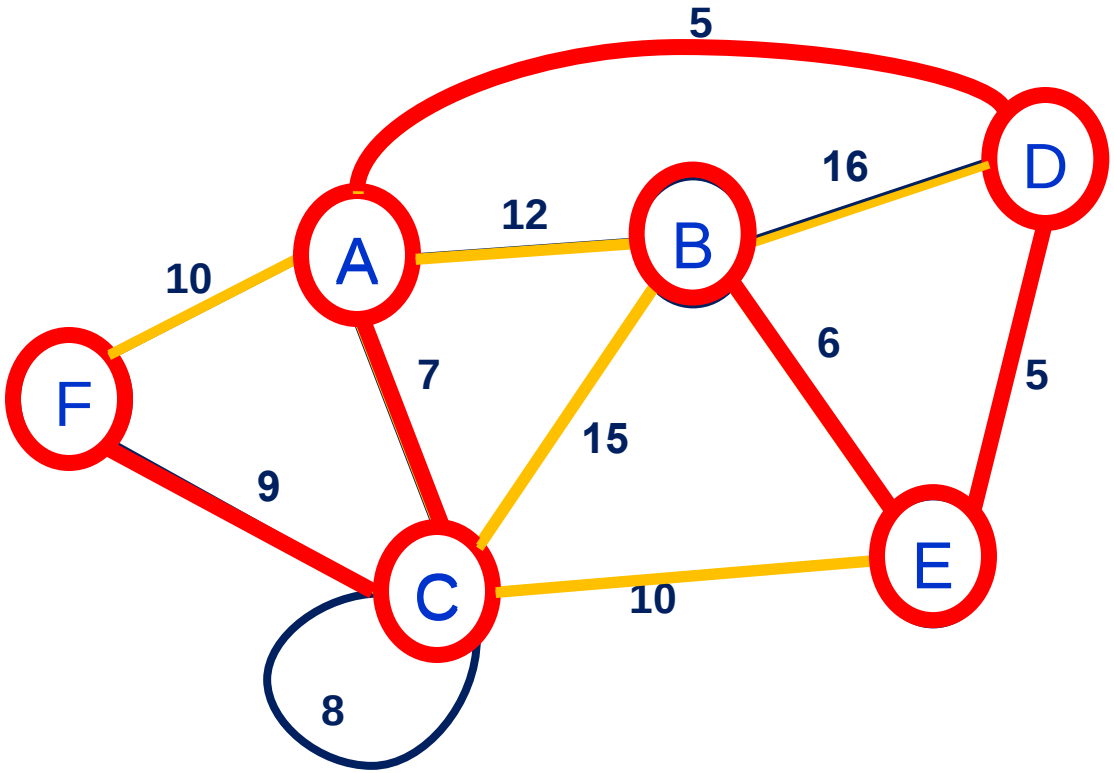
Bước 2. Chọn cạnh e có trọng lượng nhỏ nhất trong các cạnh wv mà w $\in$ X\V và v $\in$ **V** (không sinh ra chu trình)

Bước 3. V := V $\cup$ {w}; U := U $\cup$ {e}

Bước 4. Nếu U đủ n-1 cạnh thì dừng, ngược lại lặp từ bước 2.

Thuật toán Prim

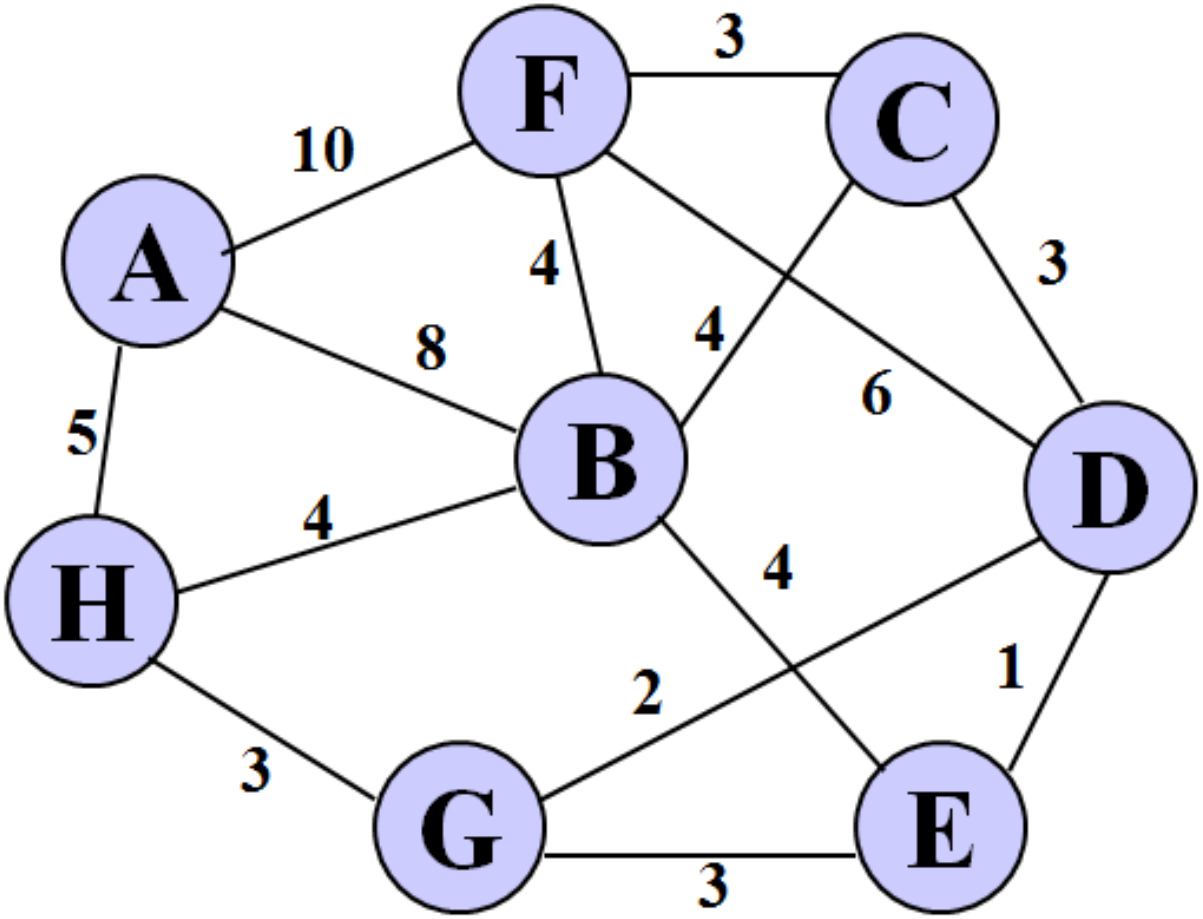
Ví dụ. Tìm cây khung ngắn nhất của đồ thị sau



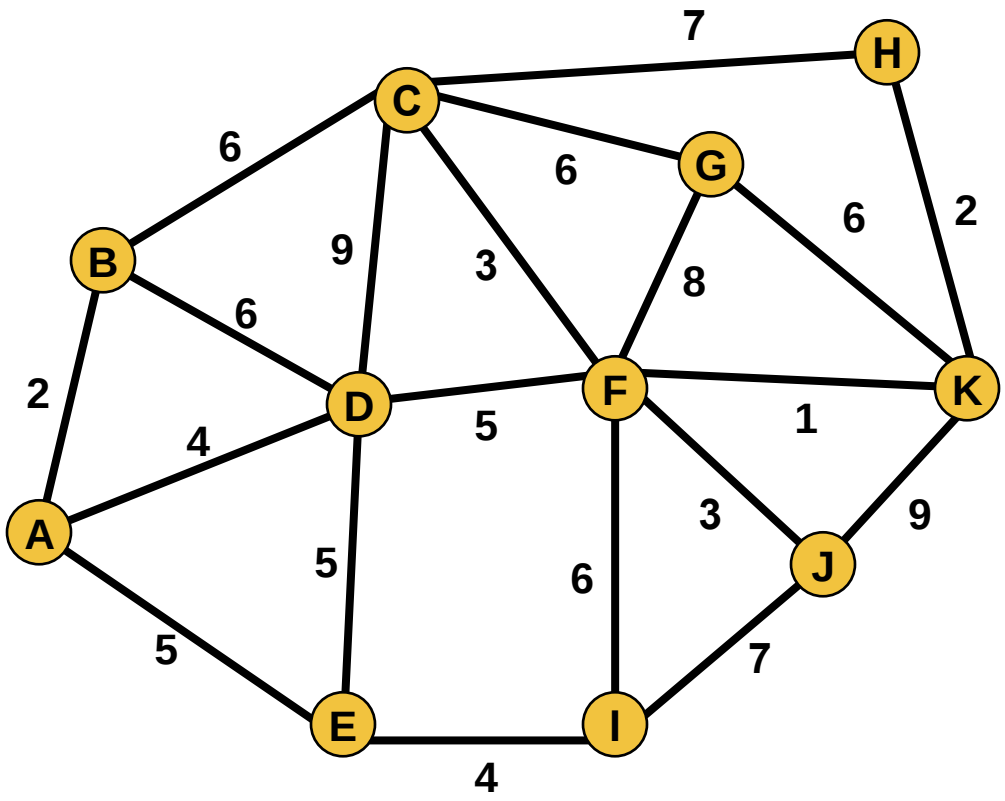
V = {F, C, A, D, E, B} U = {FC, CA, AD, DE, EB}

Trọng lượng: 32

Ví dụ. Tìm cây khung ngắn nhất của đồ thị sau

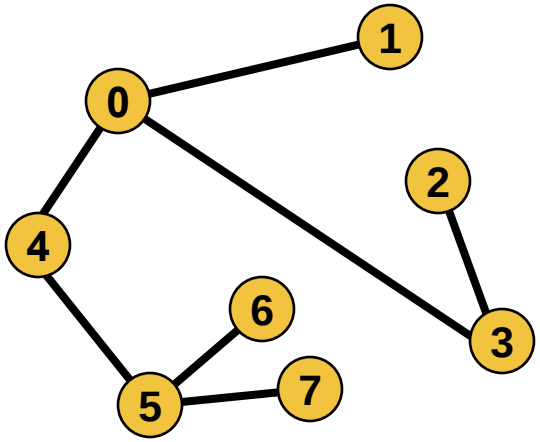


Ví dụ. Dùng thuật toán Prim để tìm cây khung nhỏ nhất của đồ thị sau:

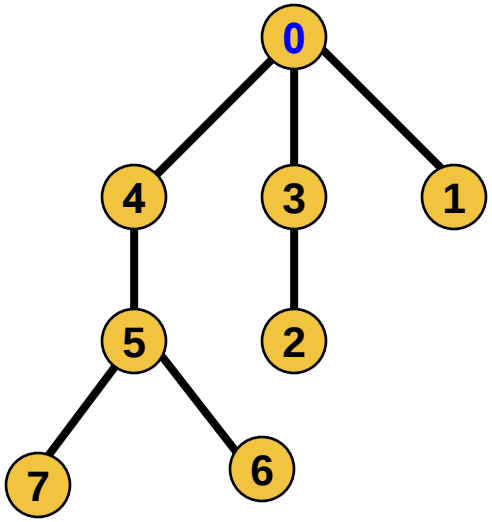


3. Cây có gốc

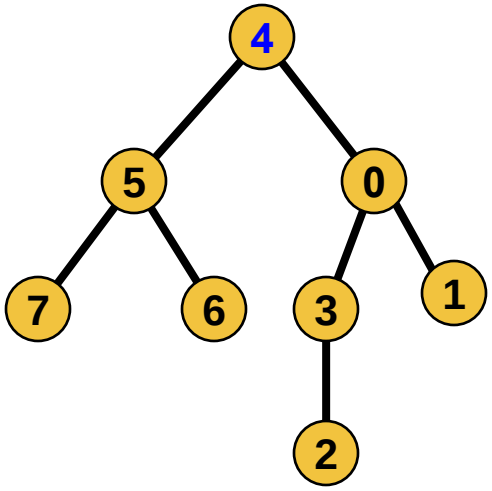
Định nghĩa. Cho *T* là một cây. Chọn một đỉnh *r* của cây gọi là ***gốc***. Vì có đường đi sơ cấp duy nhất từ gốc tới mỗi đỉnh của đồ thị nên ta định hướng mỗi cạnh là hướng từ gốc đi ra. Cây cùng với gốc sinh ra một đồ thị có hướng gọi là ***cây có gốc***



Cây T



Cây gốc 0

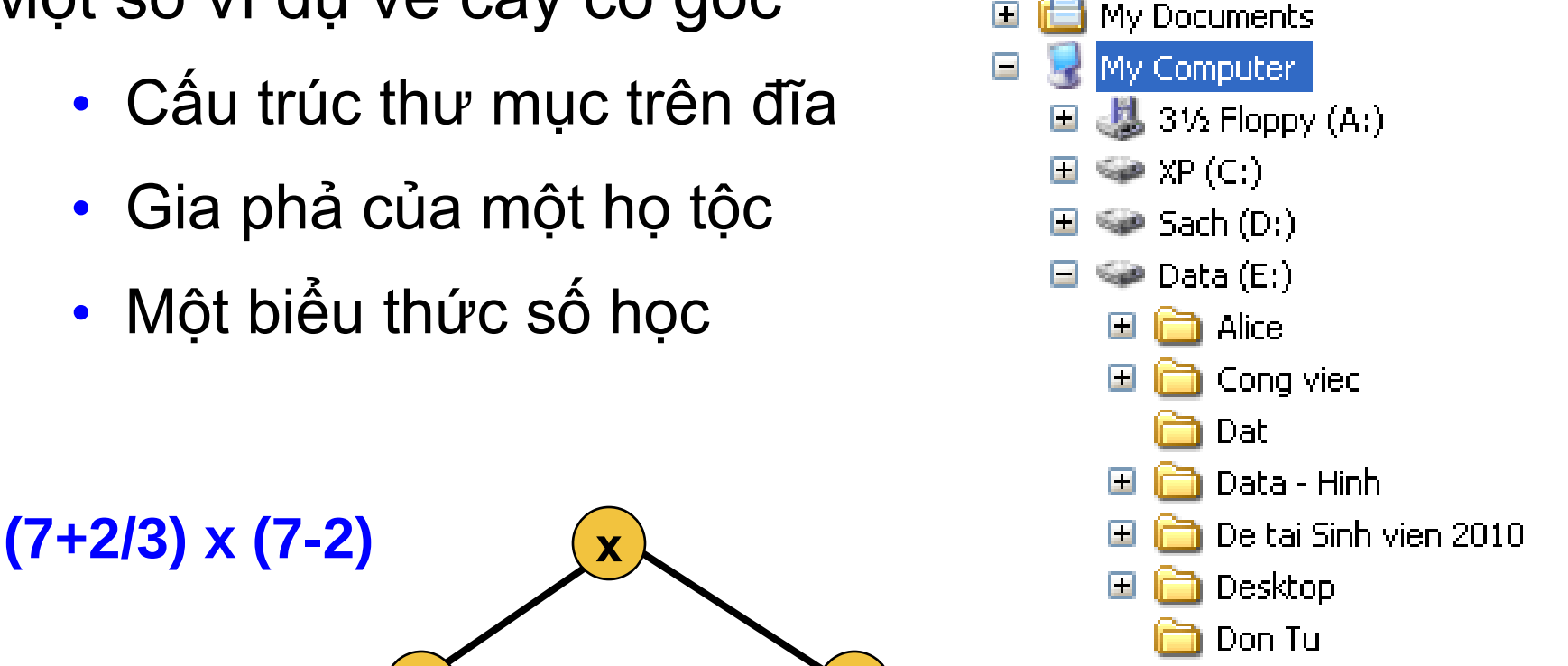


Cây gốc 4

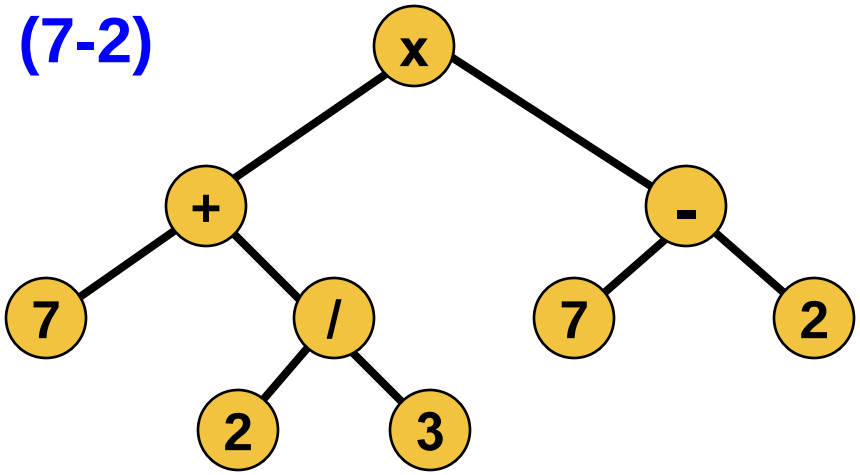
Ví dụ

Một số ví dụ về cây có gốc

- Cấu trúc thư mục trên đĩa
- Gia phả của một họ tộc
- Một biểu thức số học



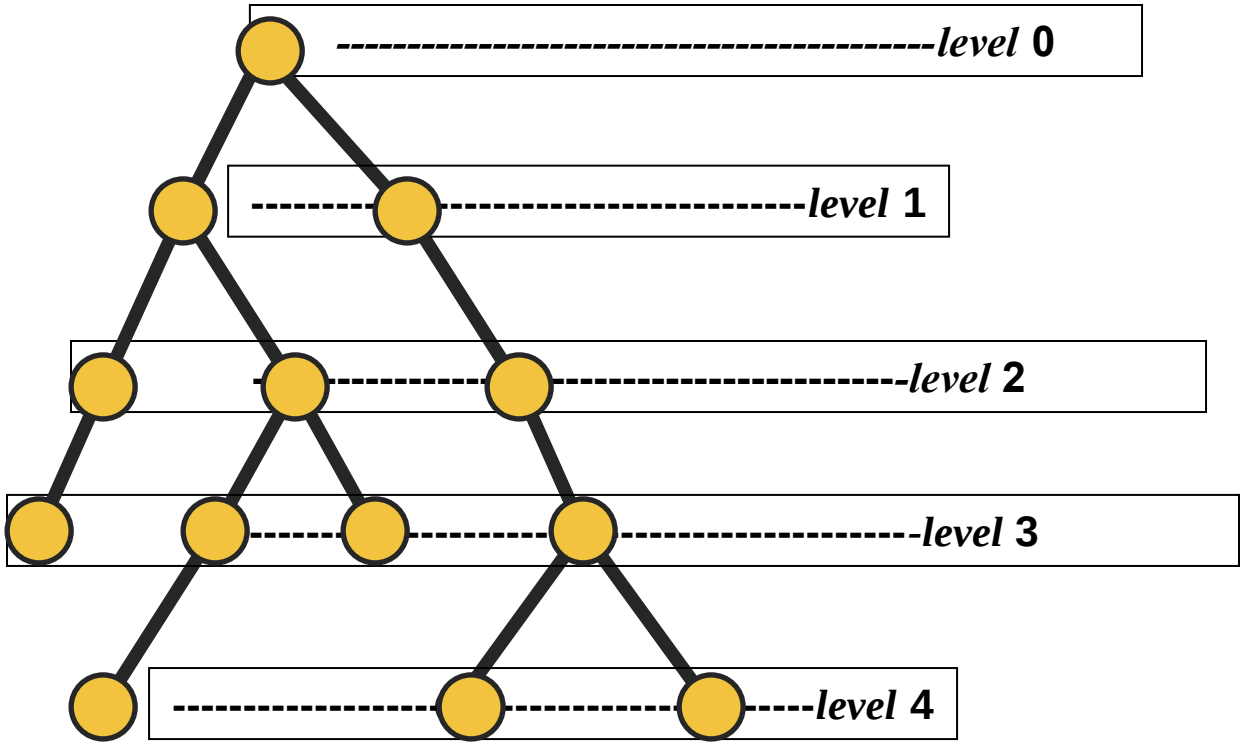
(7+2/3) x (7-2)



Một số khái niệm

Định nghĩa. Cho cây có gốc *r*.

- Gốc *r* được gọi là *đỉnh **mức** 0* (level 0).
- Các đỉnh kề với gốc *r* được xếp ở phía dưới gốc và gọi là *đỉnh mức* 1 (level 1).
- Đỉnh sau của đỉnh mức 1 (xếp phía dưới đỉnh mức1) gọi là *đỉnh mức 2.* ……
- Level(*v*) = *k* $\Leftrightarrow$ đường đi từ gốc *r* đến *v* qua *k* cung.
- ***Độ cao** của cây* là mức cao nhất của các đỉnh.


level 0
level 1
level 2
level 3
level 4

Một số khái niệm

Định nghĩa. Cho cây có gốc *r*

- Nếu *uv* là một cung của *T* thì *u* được gọi là ***cha*** *của v*, còn *v* gọi là ***con*** *của u*.
- Đỉnh không có con gọi là ***lá*** (hay *đỉnh ngoài*). Đỉnh không phải là lá gọi là ***đỉnh trong***.
- Hai đỉnh có cùng cha gọi là ***anh em***.
- Nếu có đường đi $v_1v_2\ldots v_k$ thì v_1, v_2,.., v_{k-1} gọi là ***tổ tiên*** *của* v_k. Còn v_k gọi là ***hậu duệ*** của v_1, v_2,.., v_{k-1}.
- ***Cây con*** tại đỉnh *v* là cây có gốc là *v* và tất cả các đỉnh khác là hậu duệ của *v* trong cây *T* đã cho.

Một số khái niệm

Định nghĩa. Cho *T* là cây có gốc.

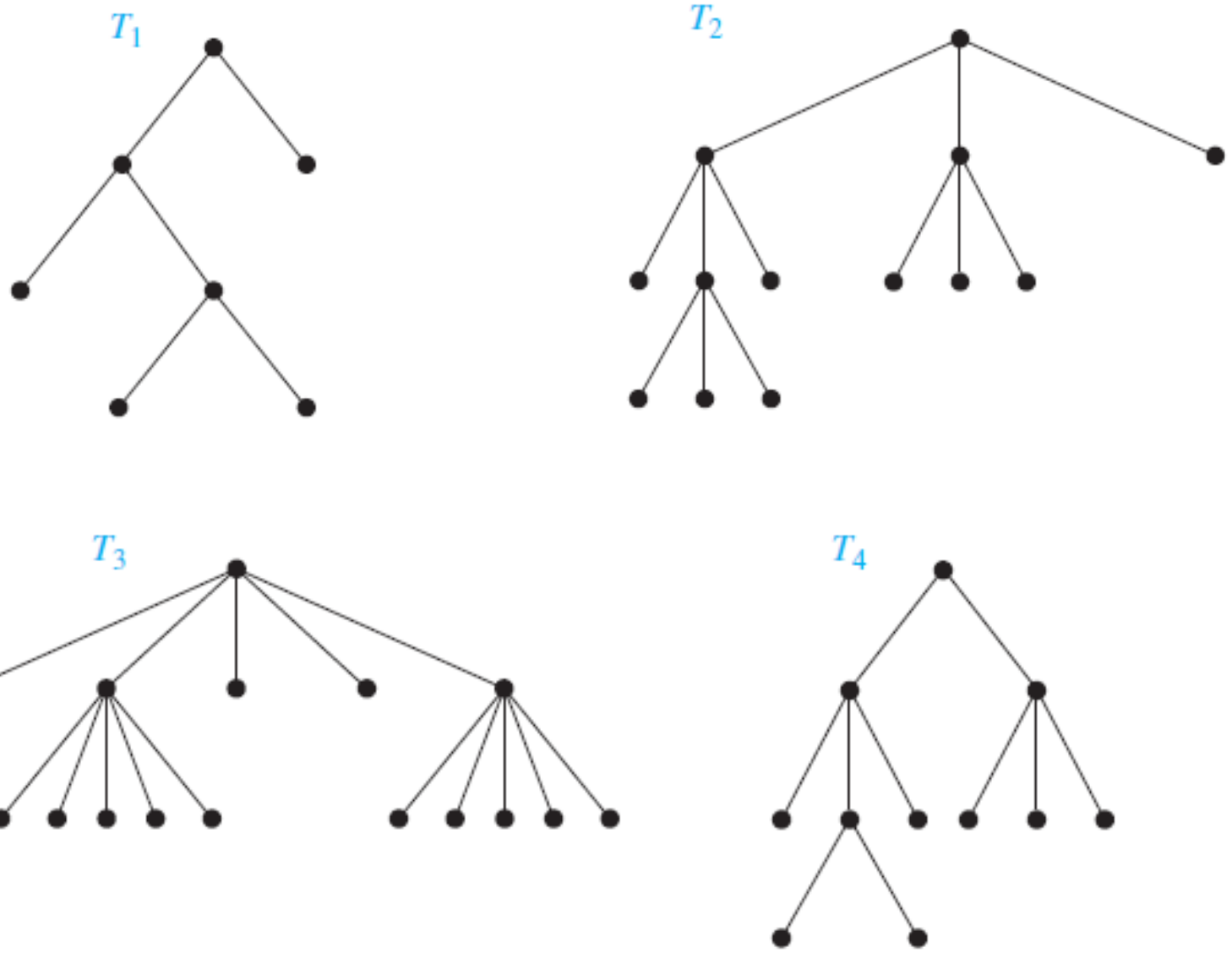
a) *T* được gọi là ***cây k-phân*** nếu mỗi đỉnh của *T* có nhiều nhất là *k* con.

b) Cây 2-phân được gọi là ***cây nhị phân***.

c) *Cây* ***k-phân đủ*** là cây mà mọi đỉnh trong có đúng *k* con.

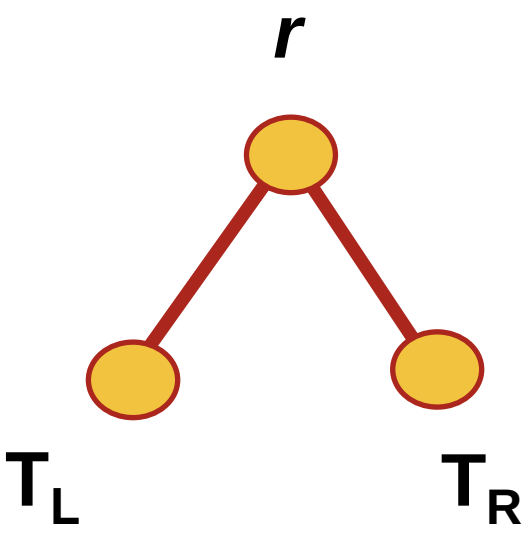
d) Cây *k*-phân với độ cao *h* được gọi là ***cân đối*** nếu các lá đều ở mức *h* hoặc *h* – 1.

Một số khái niệm



Một số khái niệm

Định nghĩa. Cho *T* là cây nhị phân có gốc là *r*. Ta có thể biểu diễn *T* như hình vẽ dưới với hai cây con tại *r* là T_L và T_R ,chúng lần lượt được gọi là ***cây con bên trái*** *và* ***cây con bên phải*** của *T*.



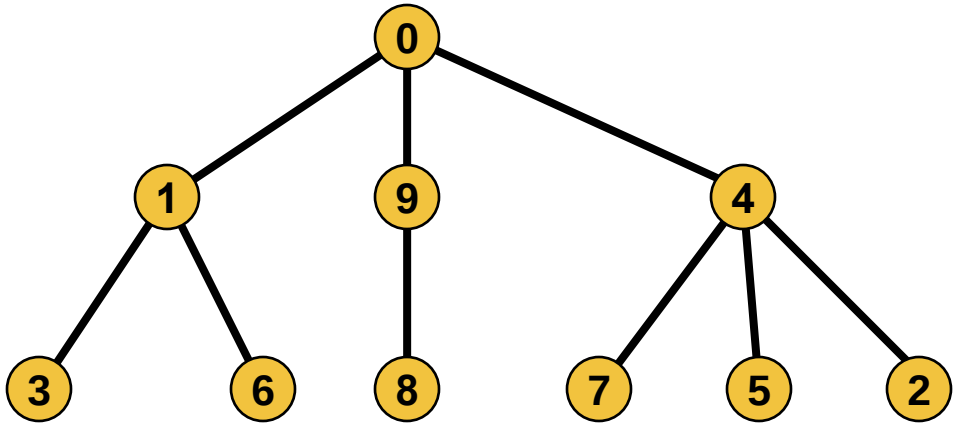
Biểu diễn cây

- Chúng ta có thể biểu diễn cây như 1 đồ thị
  - Ma trận
  - Danh sách

Nhận xét: Vì số cạnh của cây rất thưa (n-1 cạnh) nên dùng ma trận để biểu diễn cây là không hiệu quả

Biểu diễn cây bằng danh sách kề

Ví dụ. Cho cây sau



Đỉnh	Đỉnh kề		
0	1	9	4
1	3	6	
2	Ø		
3	Ø		
4	7	5	2
5	Ø		
6	Ø		
7	Ø		
8	Ø		
9	8		

Một số bài toán liên quan tới cây

- **Bài toán 1:** Kiểm tra xem đồ thị G có phải là 1 cây không
- **Bài toán 2:** Tìm gốc của cây
- **Bài toán 3:** Tính độ cao của cây với gốc là đỉnh r

4. Phép duyệt cây

Định nghĩa. ***Duyệt cây*** là liệt kê tất các đỉnh của cây theo một thứ tự nào đó thành một dãy, mỗi đỉnh chỉ xuất hiện một lần.

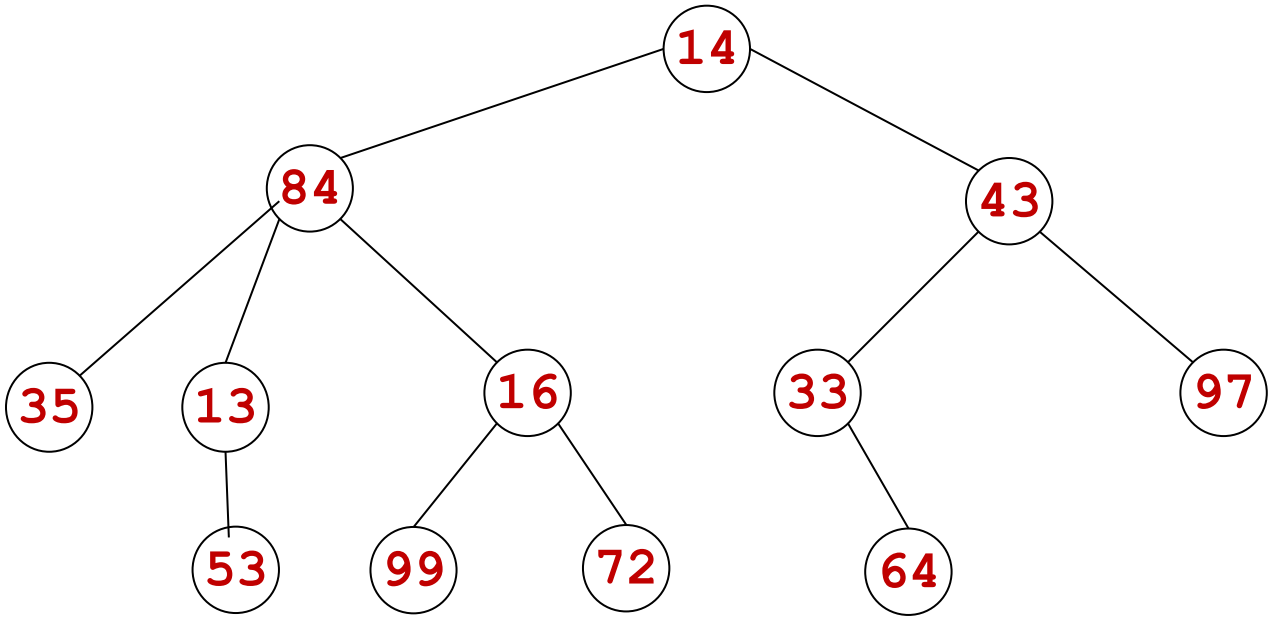
Có 2 phép duyệt cây

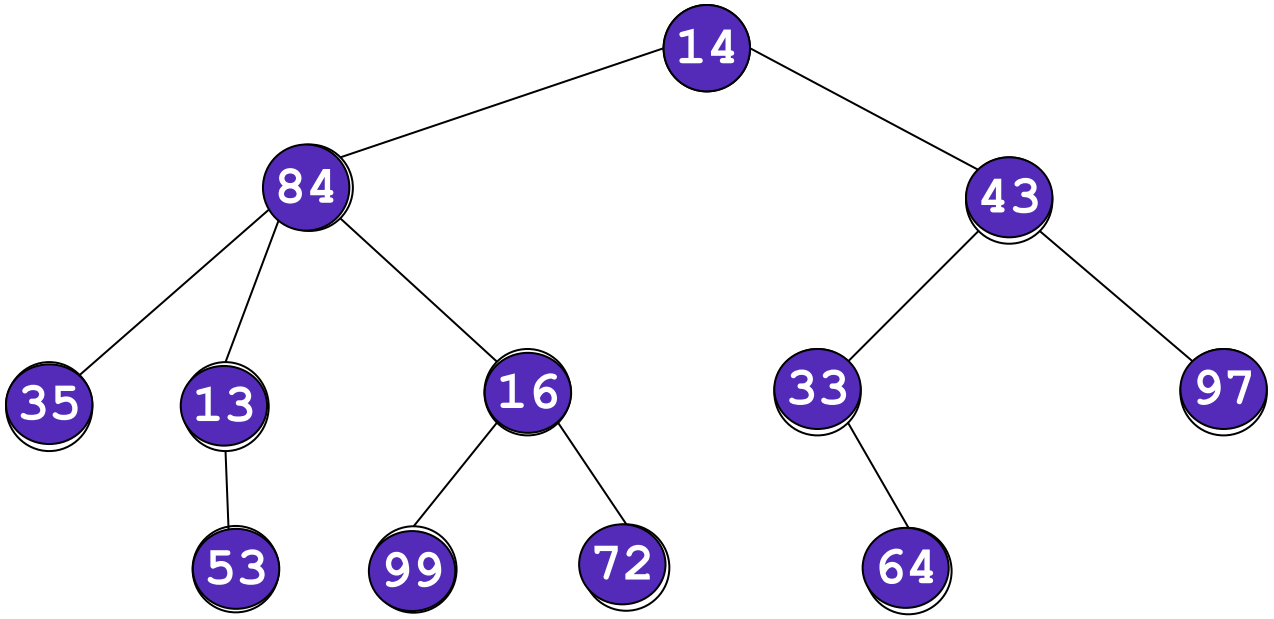
- **Phép duyệt tiền thứ tự (Preorder traversal)**
- **Phép duyệt hậu thứ tự (Posorder traversal).**

Phép duyệt tiền thứ tự

- Đến gốc *r*.
- Dùng phép duyệt tiền thứ tự để duyệt các cây con T_1 rồi cây con T_2 … từ trái sang phải.

Ví dụ. Duyệt cây sau





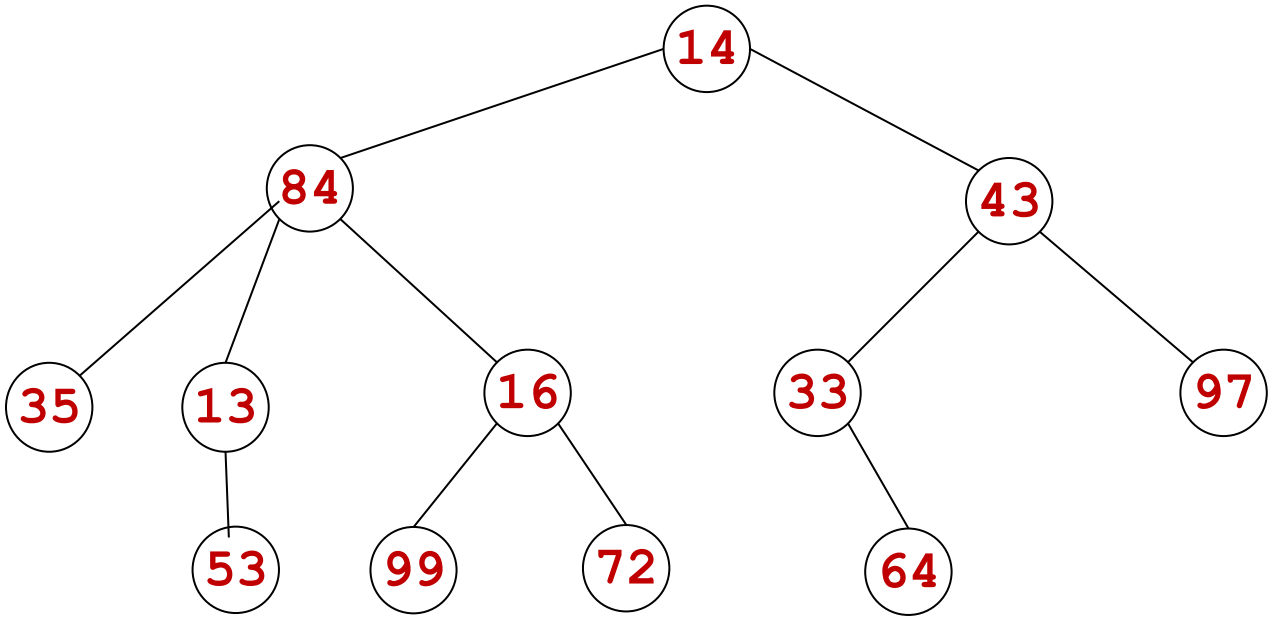
Do đó các đỉnh lần lượt được duyệt là:

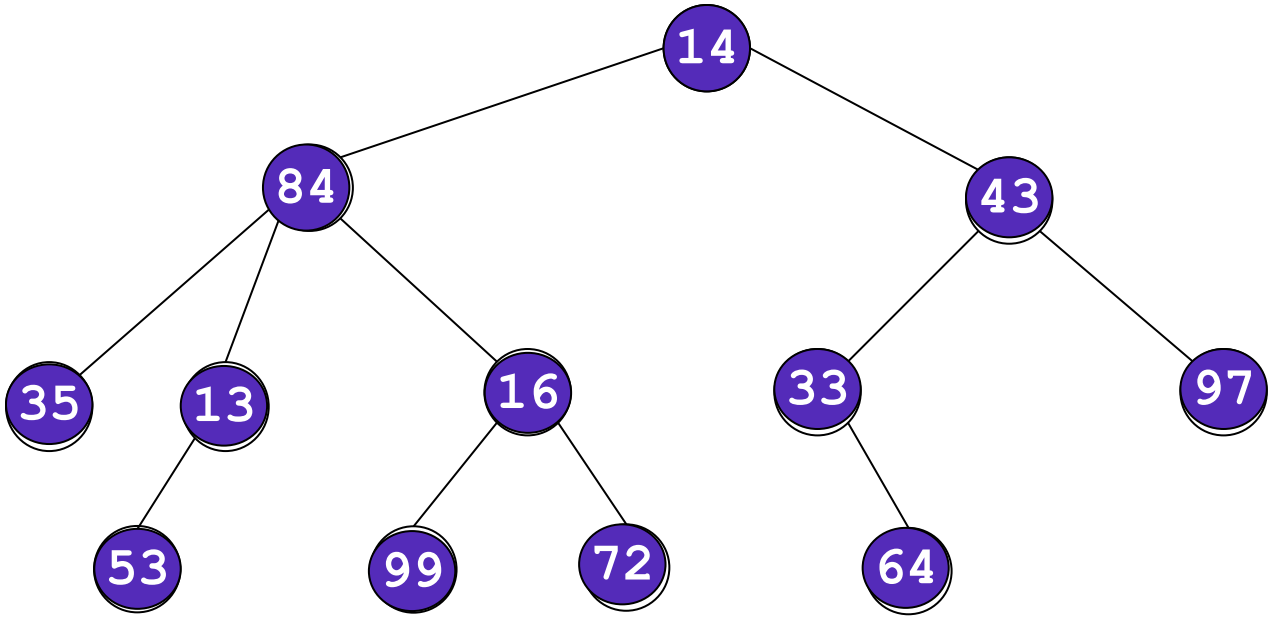
14, 84, 35, 13, 53, 16, 99, 72, 43, 33, 64, 97

Phép duyệt hậu thứ tự

- Dùng phép duyệt hậu thứ tự để lần lượt duyệt cây con T_1, T_2,…. từ trái sang phải.
- Đến gốc *r*.

Ví dụ. Duyệt cây sau





Do đó các đỉnh lần lượt được duyệt là:

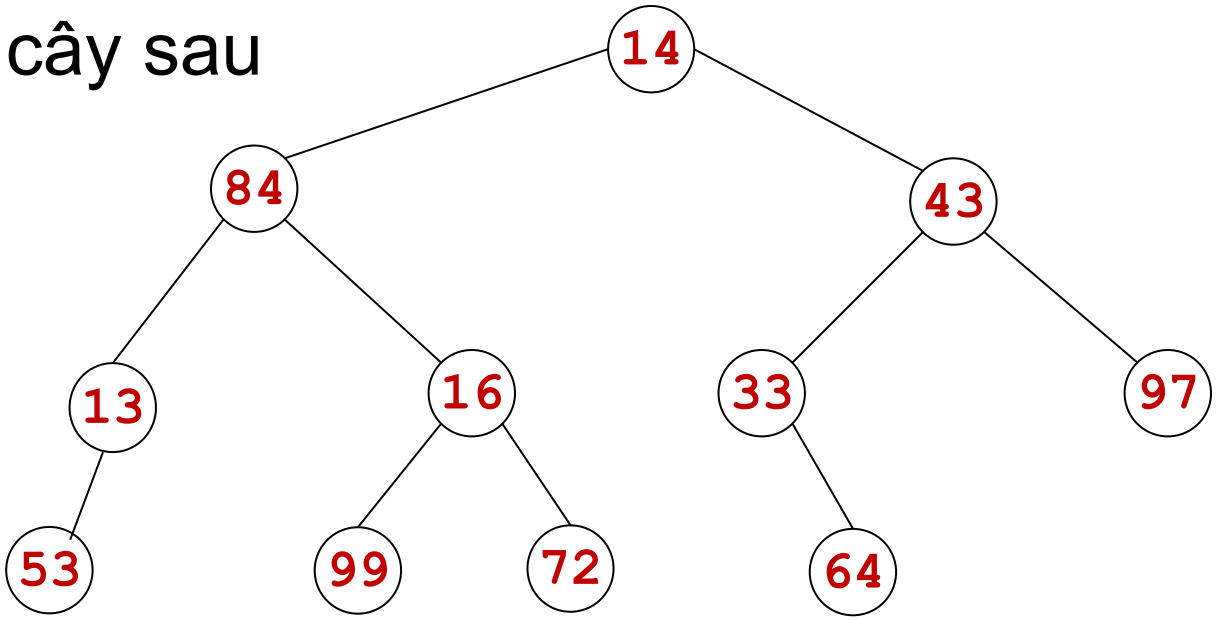
35, 53, 13, 99, 72, 16, 84, 64, 33, 97, 43, 14

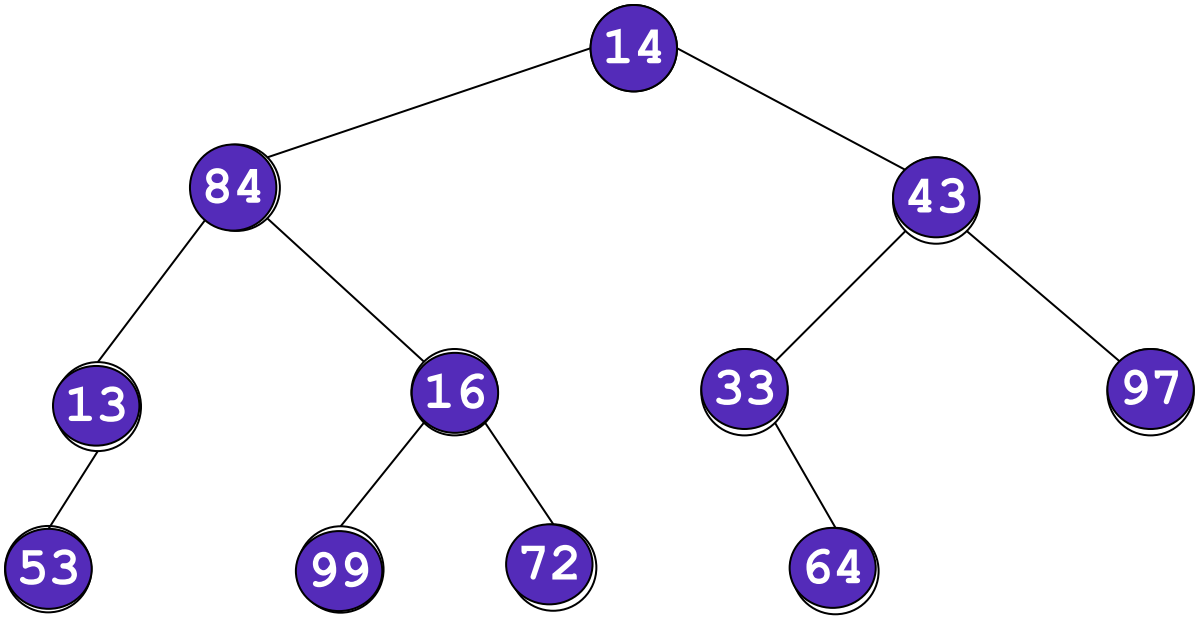
Duyệt cây nhị phân

Đối với cây nhị phân, ta có thêm phép duyệt **trung thứ tự cho *cây nhị phân* (Inorder traversal)**

- Duyệt cây con bên trái T_L theo trung thứ tự.
- Đến gốc *r*.
- Duyệt cây con bên phải theo trung thứ tự.

Ví dụ. Duyệt cây sau



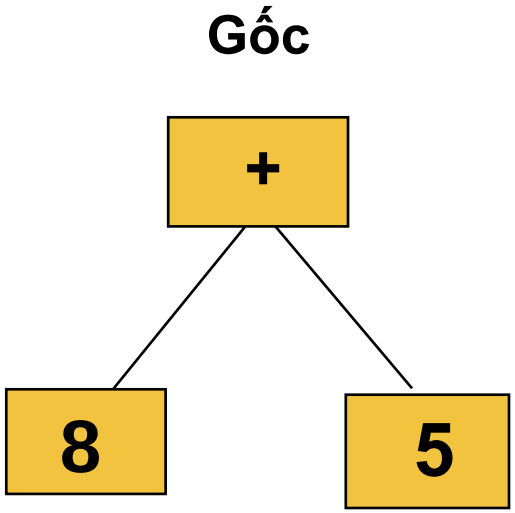


Do đó các đỉnh lần lượt được duyệt là:

53, 13, 84, 99, 16, 72, 14, 33, 64, 43, 97

Cây nhị phân biểu thức

Xét cây như sau



Khi đó, theo phép duyệt

- Tiền thứ tự: + 8 5
- Hậu thứ tự: 8 5 +
- Trung thứ tự: 8 + 5

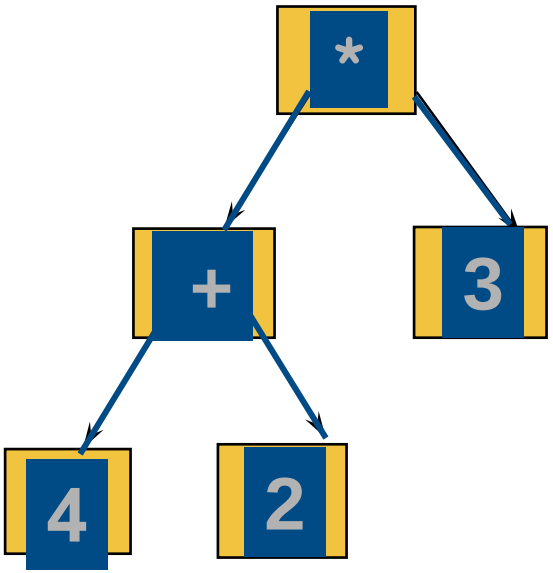
Cây nhị phân biểu thức

Định nghĩa. ***Cây nhị phân của biểu thức*** là cây nhị phân đầy đủ mà

- Mỗi biến số được biểu diễn bởi một lá.
- Mỗi đỉnh trong biểu diễn một phép toán với các thành tố là cây con tại đỉnh ấy.

Cây con bên trái và bên phải của một đỉnh trong biểu diễn cho biểu thức con, giá trị của chúng là thành tố mà ta áp dụng cho phép toán tại gốc của cây con.

Tính giá trị của biểu thức được biểu diễn bằng đồ thị sau



Kết quả?

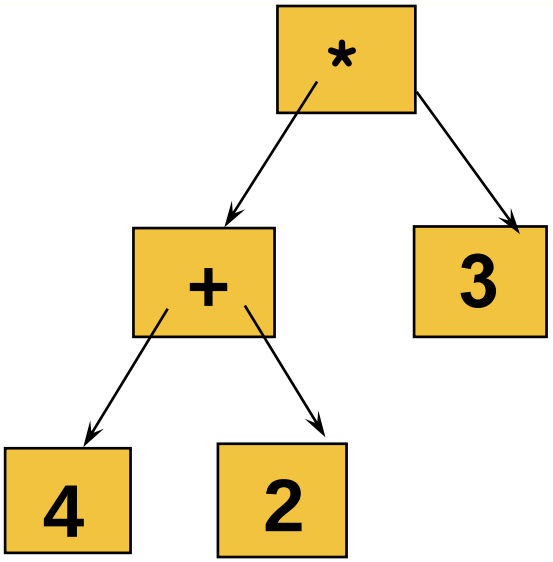
(4 + 2) * 3 = 18

Định nghĩa. Ta gọi kết quả có được khi duyệt cây nhị phân của biểu thức theo phép duyệt

- Trung thứ tự là **trung tố**

- Tiền thứ tự là **tiền tố** và được gọi là **ký pháp Ba Lan**

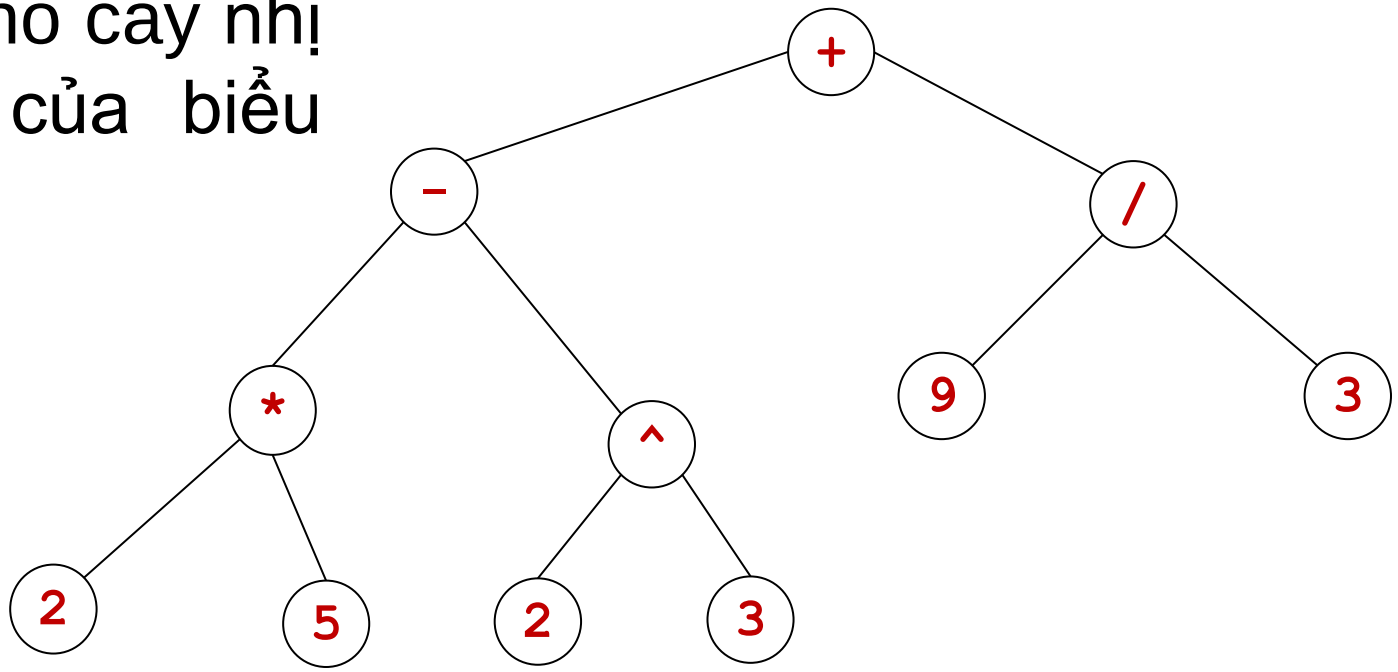
- Hậu thứ tự là **hậu tố** và được gọi là **ký pháp Ba Lan ngược**



Khi đó

- Trung tố: **4 + 2 * 3**
- Tiền tố: *** + 4 2 3** Ký pháp Ba lan
- Hậu tố: **4 2 + 3 *** Ký pháp Ba lan ngược

Ví dụ. Cho cây nhị phân T của biểu thức



Hãy tìm tiền tố, trung tố, hậu tố của G?

Ký pháp Ba Lan

Nhận xét. Để tính biểu thức khi có ký **pháp Ba Lan** ta tính từ phải sang trái: Bắt đầu từ bên phải, khi gặp một phép toán thì phép toán này được thực hiện cho 2 thành tố ngay bên phải nó, kết quả này là thành tố cho phép toán tiếp theo.

Ví dụ. Tính giá trị của ký pháp Ba Lan sau:

a) − ∗ 2 / 8 4 3

b) ^ − ∗ 3 3 ∗ 4 2 5

c) + − ^ 3 2 ^ 2 3 / 6 − 4 2

d) ∗ + 3 + 3 ^ 3 + 3 3 3

Giải. a) − ∗ 2 / 8 4 3

≡ − ∗ 2 2 3

≡ − 4 3

≡ 1

b) ^ − ∗ 3 3 ∗ 4 2 5

≡ ^ − ∗ 3 3 8 5

≡ ^ − 9 8 5

≡ ^ 1 5

≡ 1

c) + − ^ 3 2 ^ 2 3 / 6 − 4 2 ≡?

d) ∗ + 3 + 3 ^ 3 + 3 3 3 ≡?

Ký pháp Ba Lan ngược

Nhận xét. Để tính biểu thức khi có **ký pháp Ba Lan ngược**, ta tính từ bên trái, khi gặp một phép toán thì phép toán này được thực hiện cho 2 thành tố ngay bên trái nó, kết quả này là thành tố cho phép toán tiếp theo.

Ví dụ. Tính giá trị của ký pháp Ba Lan ngược sau:

a) 5 2 1−−3 1 4 ++ ∗

b) 9 3 / 5 + 7 2 − ∗

c) 3 2 ∗ 2 ^ 5 3 − 8 4 / ∗ −