Cây

Biên soạn: Nguyễn Viết Hưng

ĐH Sư Phạm Tp Hcm

^o^^o^^o^

^o^^o^

^o^

Cây

1. **ĐN và tính chất**
2. **Cây khung ngắn nhất**
3. **Cây có hướng**
4. **Phép duyệt cây**

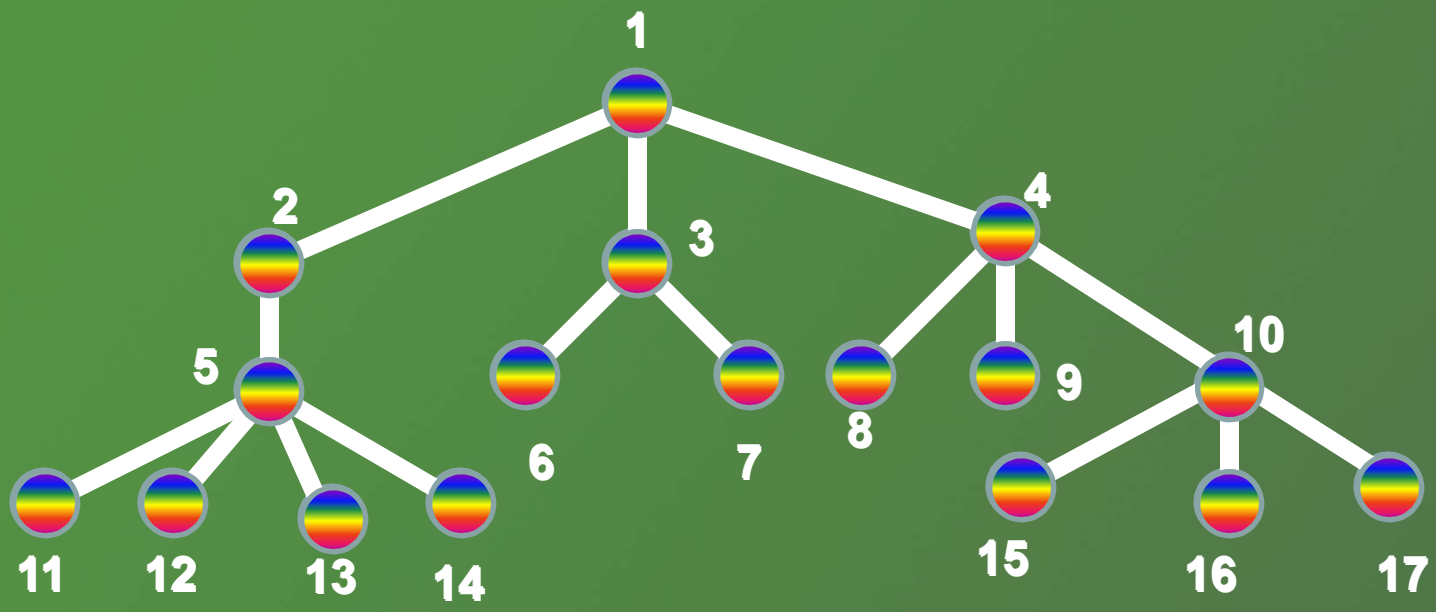
Định nghĩa và tính chất

Định nghĩa cây.

a) Cho G là đồ thị vô hướng. G được gọi là *một cây* nếu G liên thông và không có chu trình đơn.

b) *Rừng* là đồ thị mà mỗi thành phần liên thông của nó là một cây.

Định nghĩa và tính chất



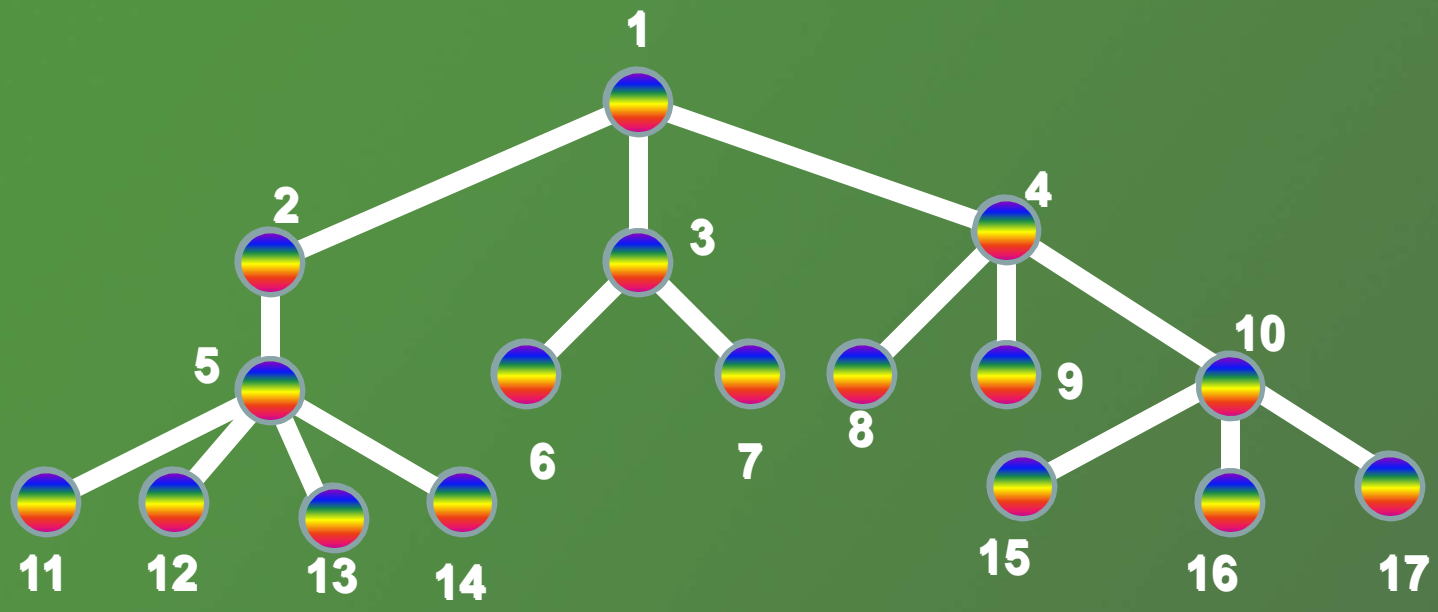
Định nghĩa và tính chất

Điều kiện cần và đủ(cây).

Cho T là đồ thị vô hướng có n đỉnh. Các phát biểu sau đây là tương đương:

i. **T là cây.**
ii. **T liên thông và có n-1 cạnh.**
iii. **T không có chu trình và có n-1 cạnh**
iv. **T liên thông và mỗi cạnh là một cầu.**
v. **Giữa hai đỉnh bất kỳ có đúng một đường đi nối chúng với nhau.**
vi. **T không có chu trình và nếu thêm vào một cạnh giữa hai đỉnh không kề nhau thì có một chu trình duy nhất.**

Định nghĩa và tính chất



Định nghĩa và tính chất

Định nghĩa cây khung.

Cho G = (V,E) là đồ thị vô hướng.

- T là đồ thị con khung của G.
- Nếu T là một cây thì T được gọi là *cây khung*(hay *cây tối đại*, hay *cây bao trùm*) của đồ thị G.

Cây khung ngắn nhất

Thuật toán tìm cây khung ngắn nhất

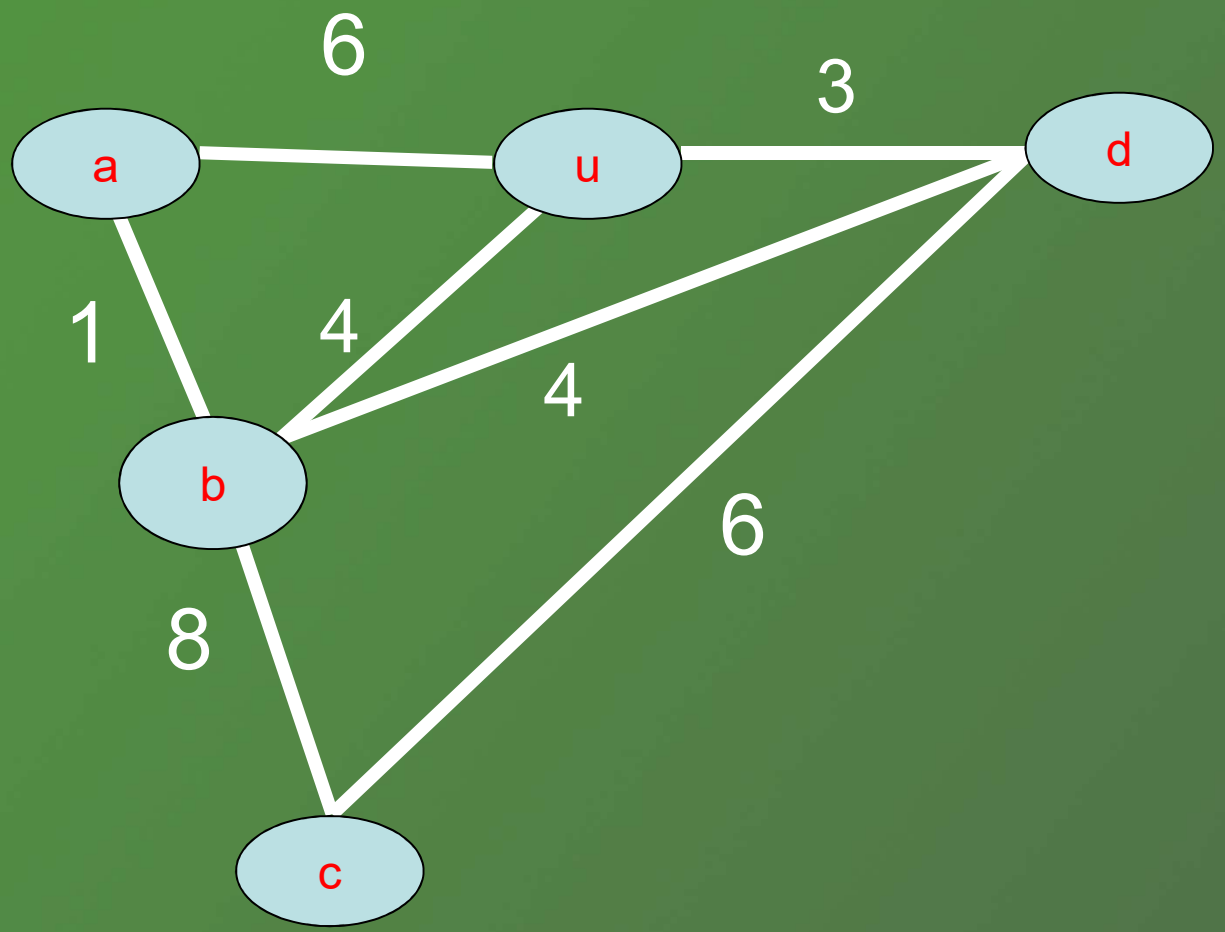
Thuật toán Kruscal: Cho G là đồ thị liên thông, có trọng số, n đỉnh.

Bước 1. Trước hết chọn cạnh ngắn nhất e_1 trong các cạnh của G.

Bước 2. Khi đã chọn k cạnh $e_1,e_2,\ldots e_k$ thì chọn tiếp cạnh e_{k+1} ngắn nhất trong các cạnh còn lại của G sao cho không tạo thành chu trình với các cạnh đã chọn trước.

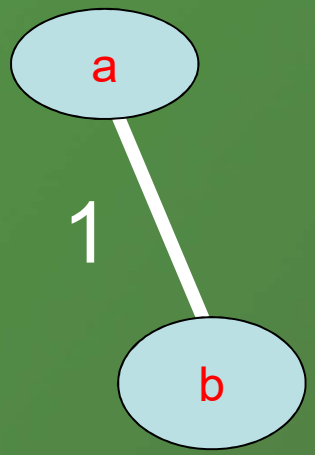
Bước 3. Chọn đủ n-1 cạnh thì dừng.

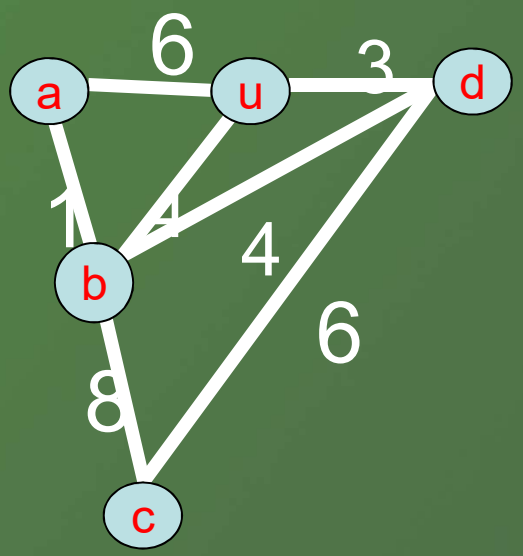
Cây khung ngắn nhất



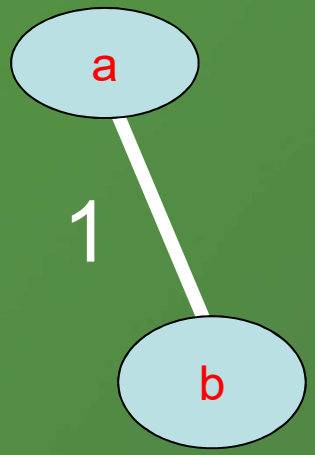
Cây khung ngắn nhất

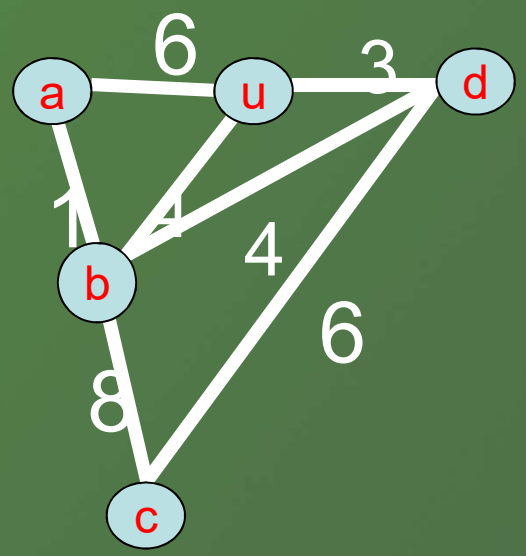
S_1





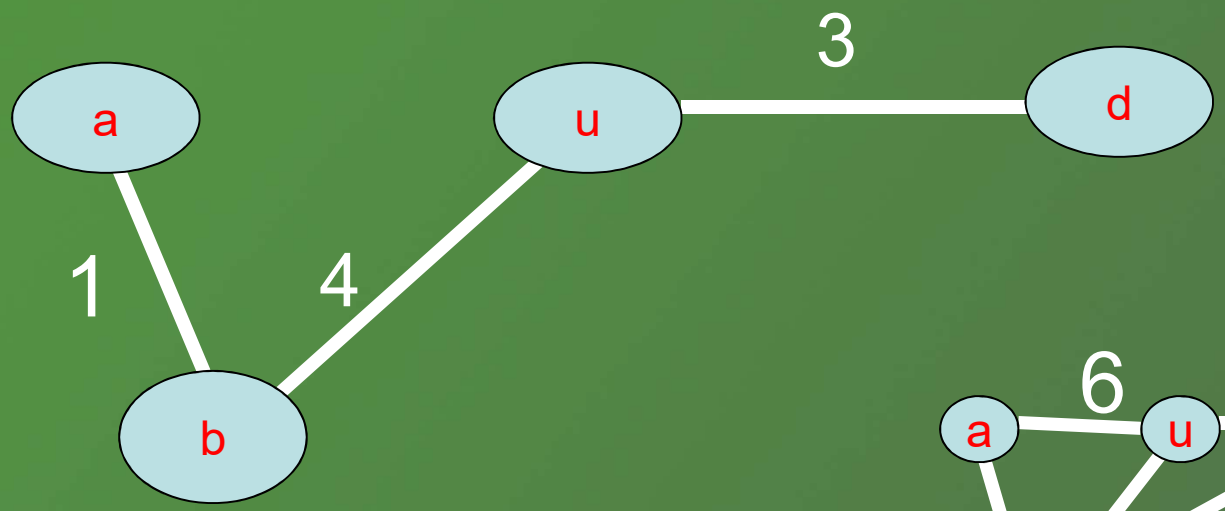
Cây khung ngắn nhất

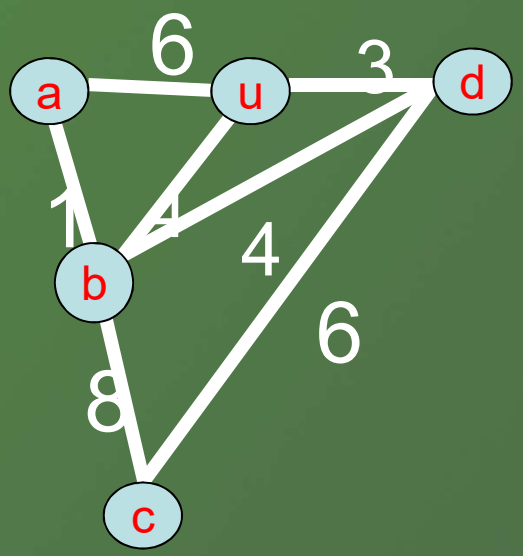




S_2

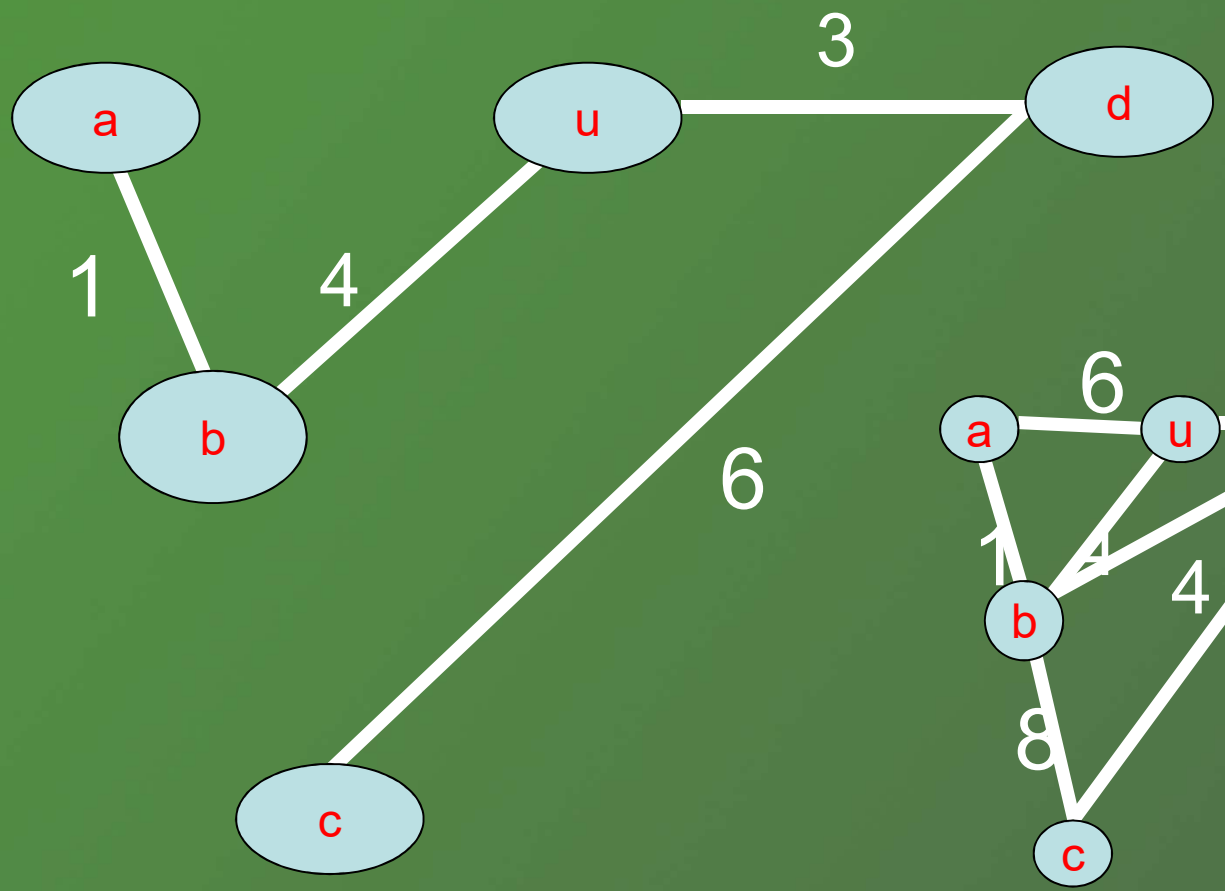
Cây khung ngắn nhất





S_3

Cây khung ngắn nhất



S_4

Cây khung ngắn nhất

Thuật toán tìm cây khung ngắn nhất

Thuật toán Prim.

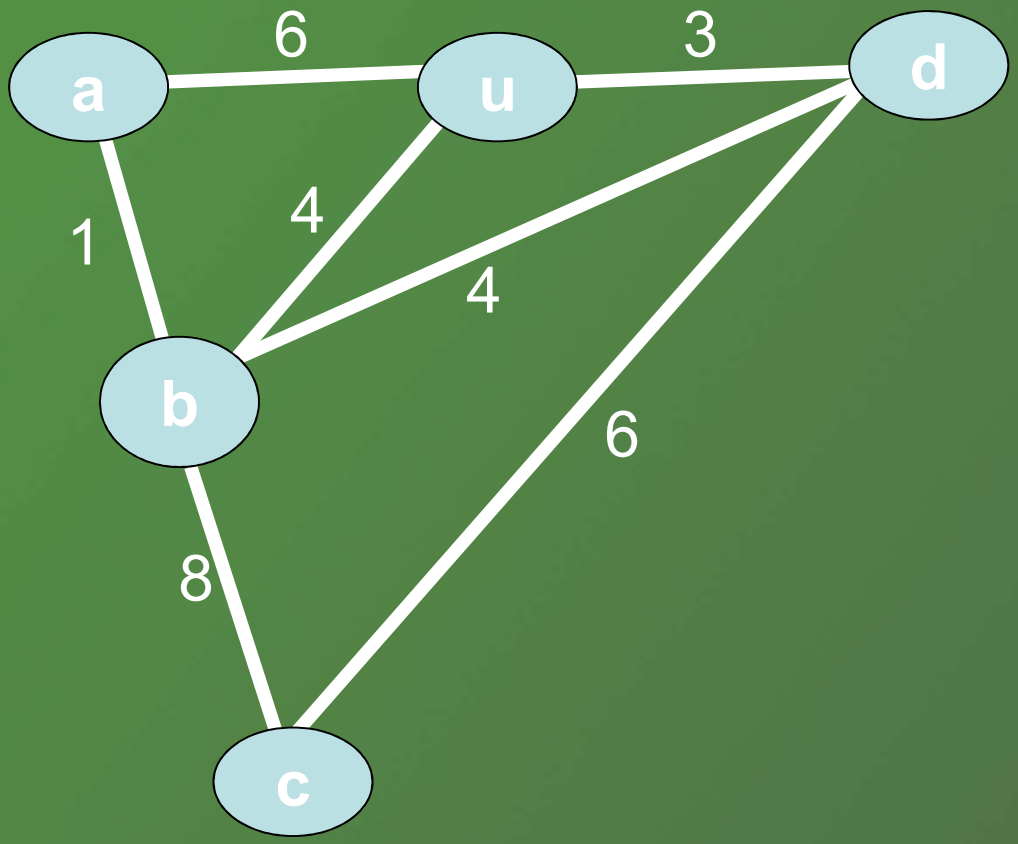
Bước 1. Chọn 1 đỉnh bất kỳ v_1 để có cây T_1 chỉ gồm 1 đỉnh.

Bước 2. Khi đã chọn cây T_k thì chọn tiếp cây $T_{k+1} = T_k \cup e_{k+1}$. Trong đó e_{k+1} là cạnh ngắn nhất trong các cạnh có một đầu mút thuộc T_k và đầu mút kia không thuộc T_k

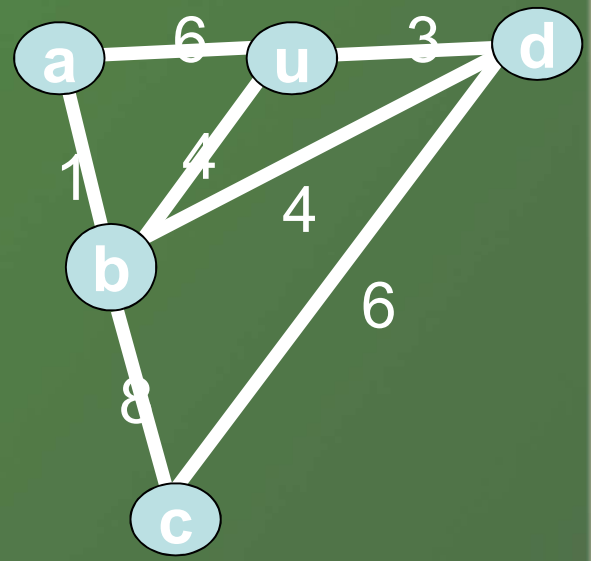
Bước 3. Chọn được cây T_n thì dừng.

Cây khung ngắn nhất

Thuật toán tìm cây khung ngắn nhất



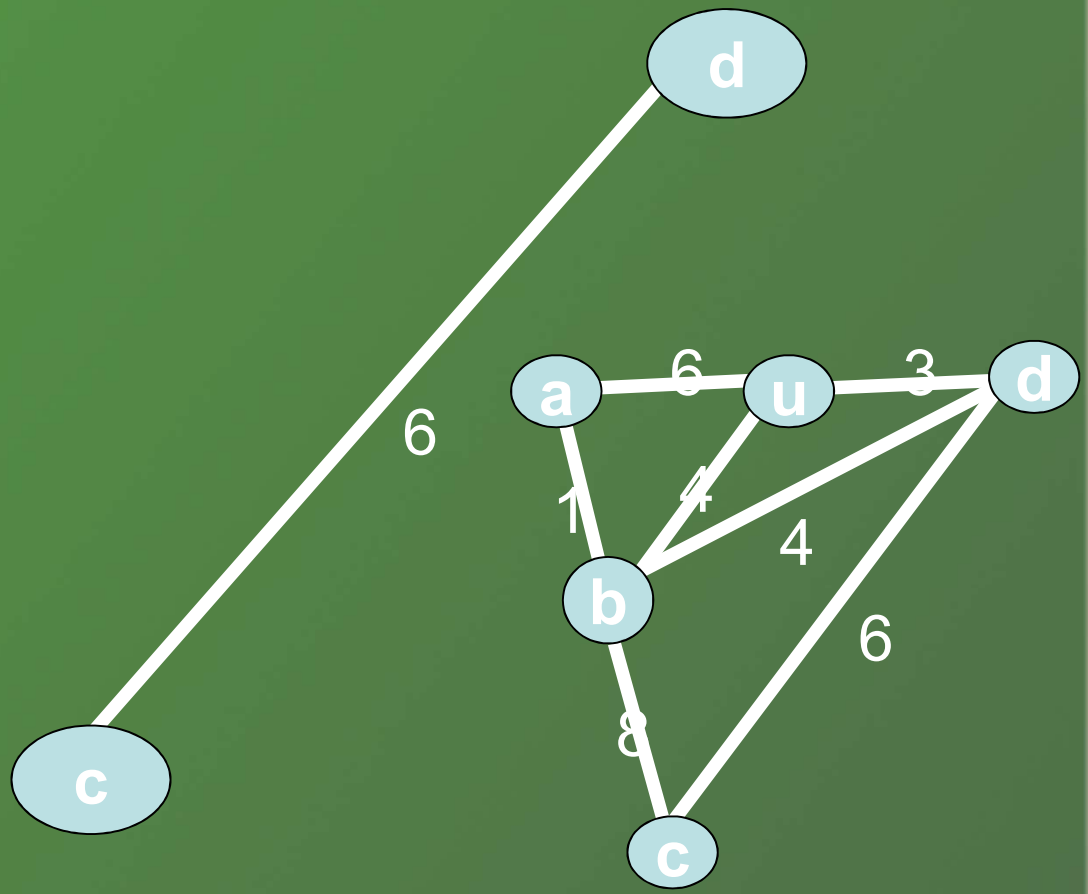
Cây khung ngắn nhất



T_1

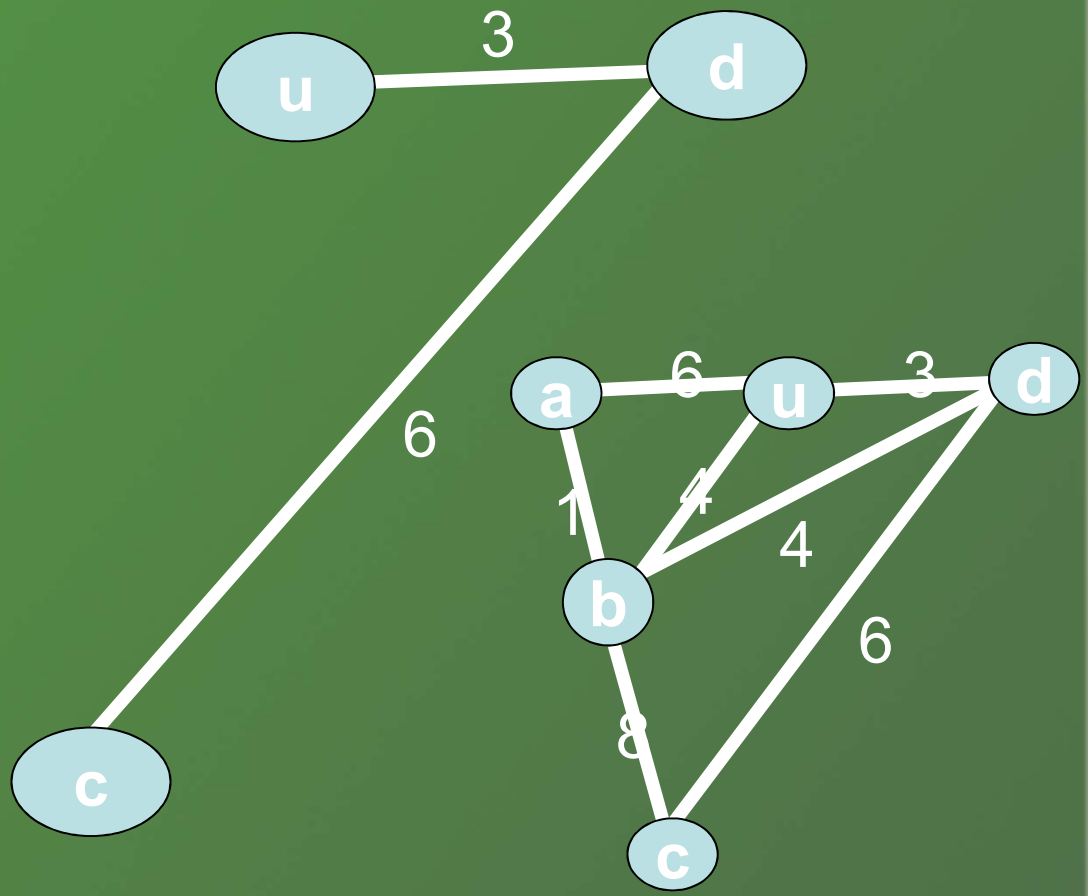
c

Cây khung ngắn nhất



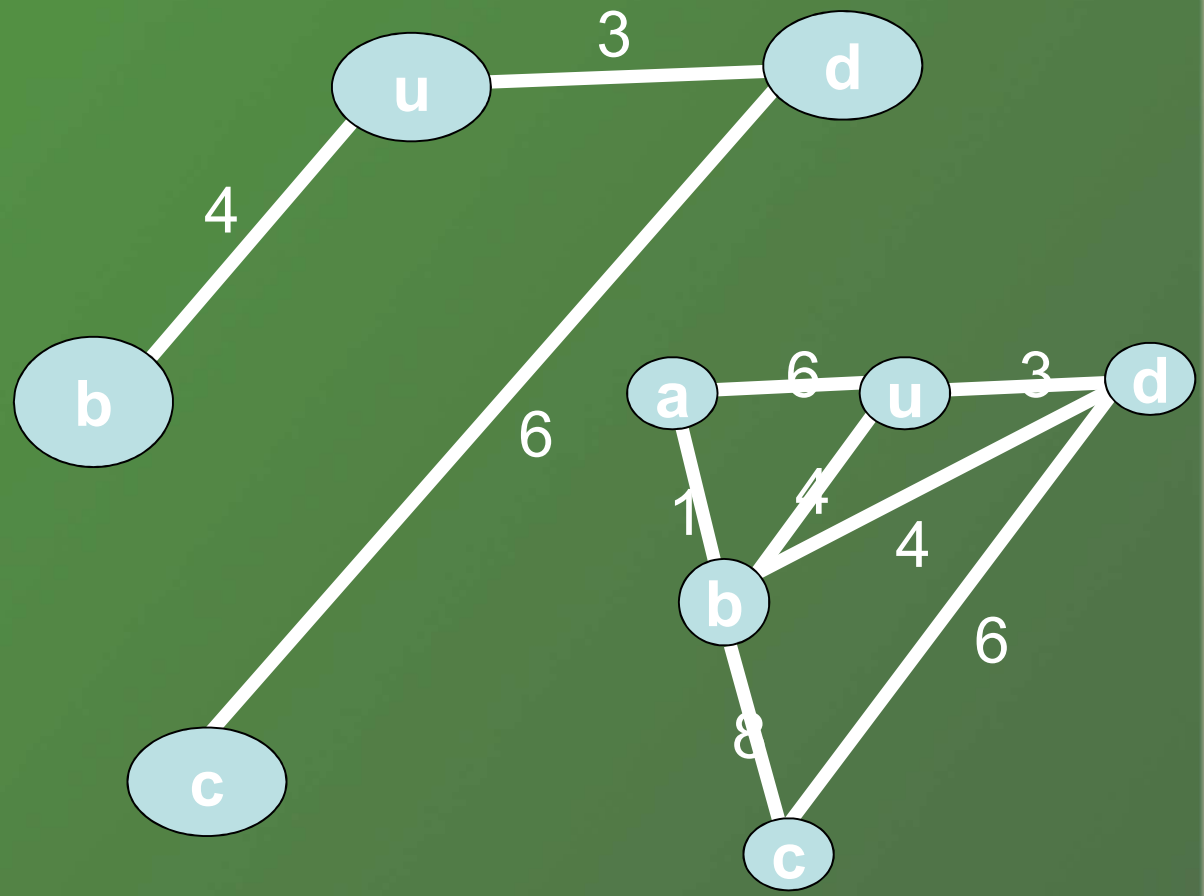
T_2

Cây khung ngắn nhất



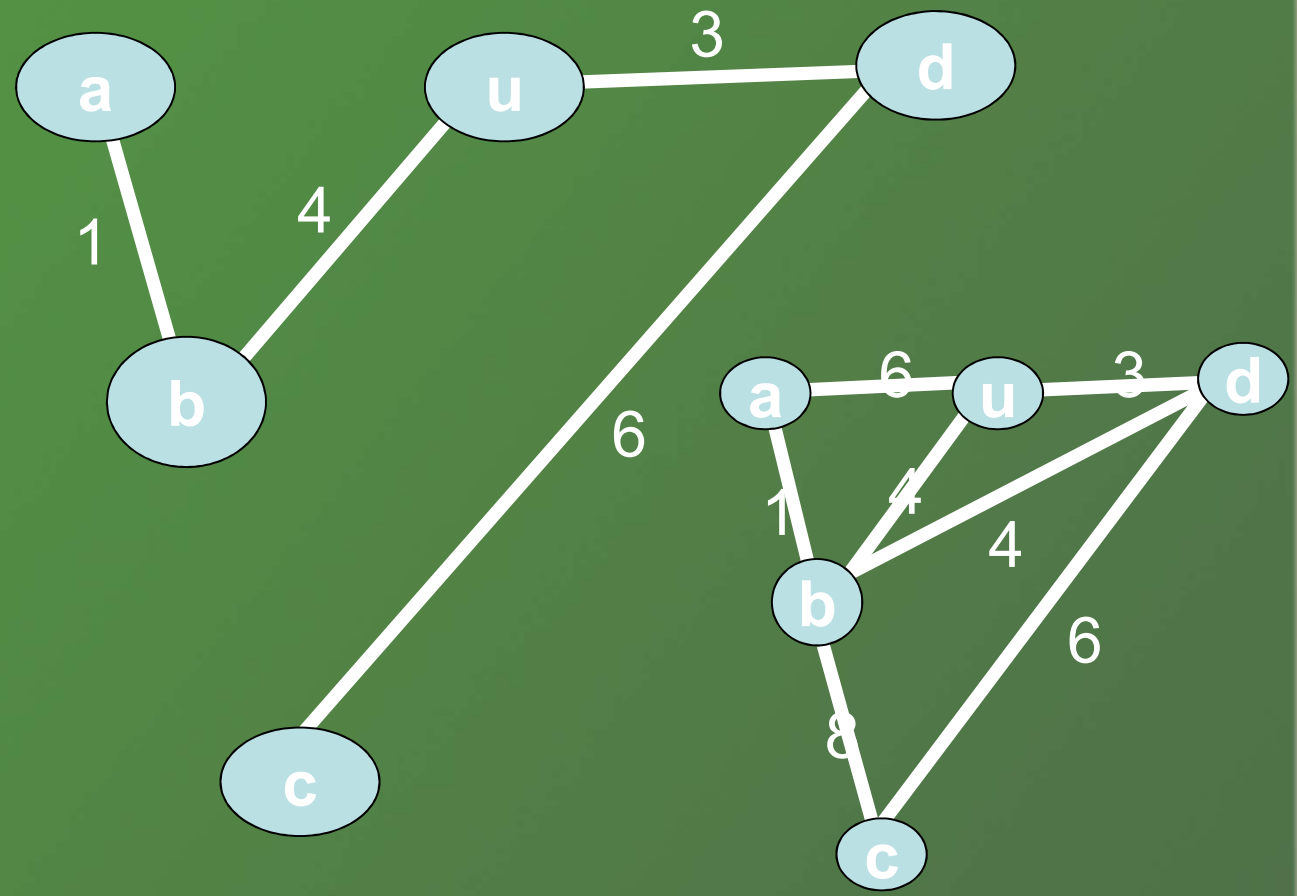
T_3

Cây khung ngắn nhất



T_4

Cây khung ngắn nhất



T_5

Cây có hướng

Định nghĩa.

a. Cây có hướng là đồ thị có hướng mà đồ thị đối xứng của nó là một cây.

b. Cây có gốc là một cây có hướng, trong đó có một đỉnh đặc biệt gọi là gốc, từ gốc có đường đi đến mọi đỉnh khác của cây.

Nhận xét

Trong một cây có gốc r thì deg-(r) = 0, deg-(v) = 1 với mọi đỉnh không phải là gốc.

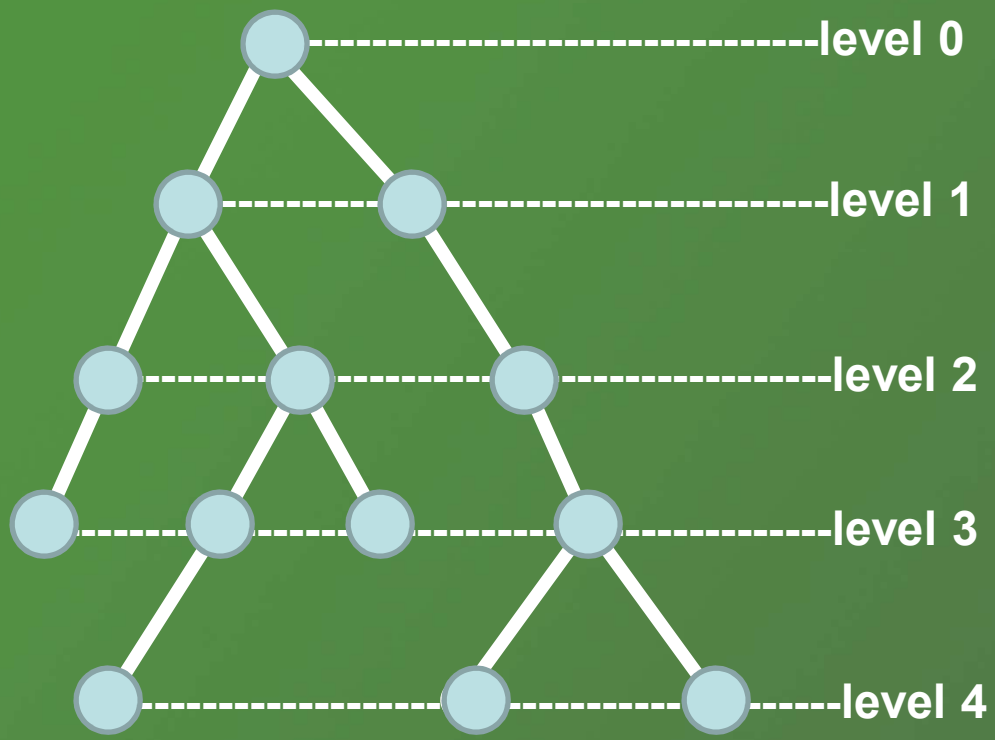
Cây có hướng

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*(level1).
- Đỉnh sau của đỉnh mức 1(xếp phía dưới đỉnh mức)gọi là *đỉnh mức 2*.
- ……
- Level (v) = k $\Leftrightarrow$ đường đi từ gốc r đến v qua k cung.

Cây có hướng



Cây có hướng

Định nghĩa

Cho cây có gốc r

a) Nếu uv là một cung của T thì u được gọi là *cha của v*,v gọi là *con của u*.

b) Đỉnh không có con gọi là *lá*(hay *đỉnh ngoài*). Đỉnh không phải là lá gọi là *đỉnh trong*.

c) Hai đỉnh có cùng cha gọi là *anh em*.

Cây có hướng

Định nghĩa

Cho cây có gốc r

d) 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}.

e) Cây con tại đỉnh v là cây có gốc là v và tất cả các đỉnh khác là mọi hậu duệ của v trong cây T đã cho.

Cây có hướng

Định nghĩa

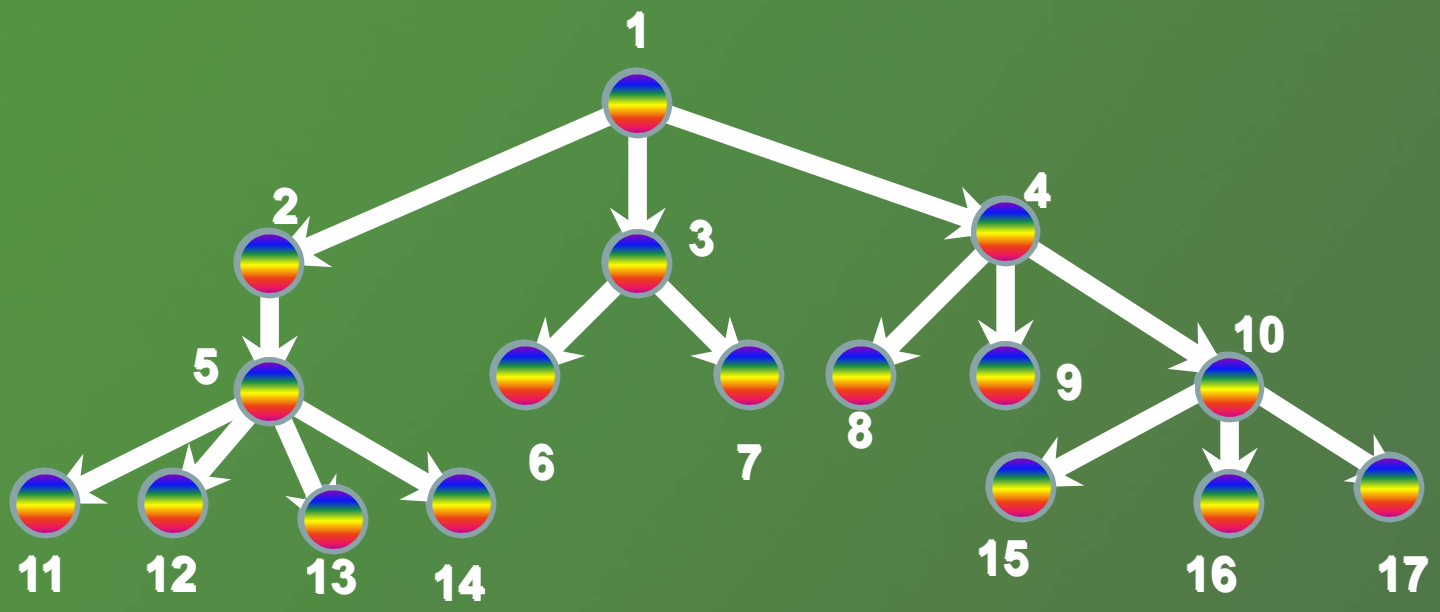
Cho T là cây có gốc.

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.

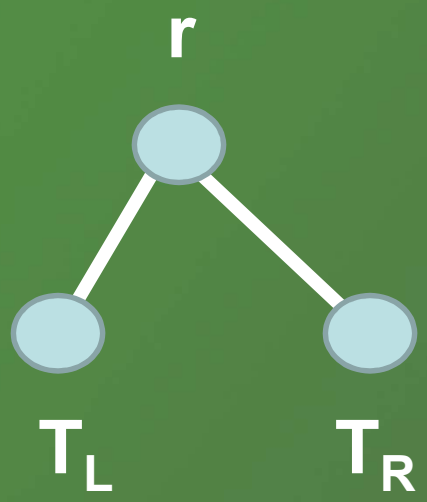
Cây có hướng



Cây có hướng

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. T_L và T_R lần lượt được gọi là cây con bên trái và cây con bên phải của T.



Cây có hướng

Định nghĩa

Độ dài đường đi trong và độ dài đường đi ngoài

Cho T là cây nhị phân đủ.

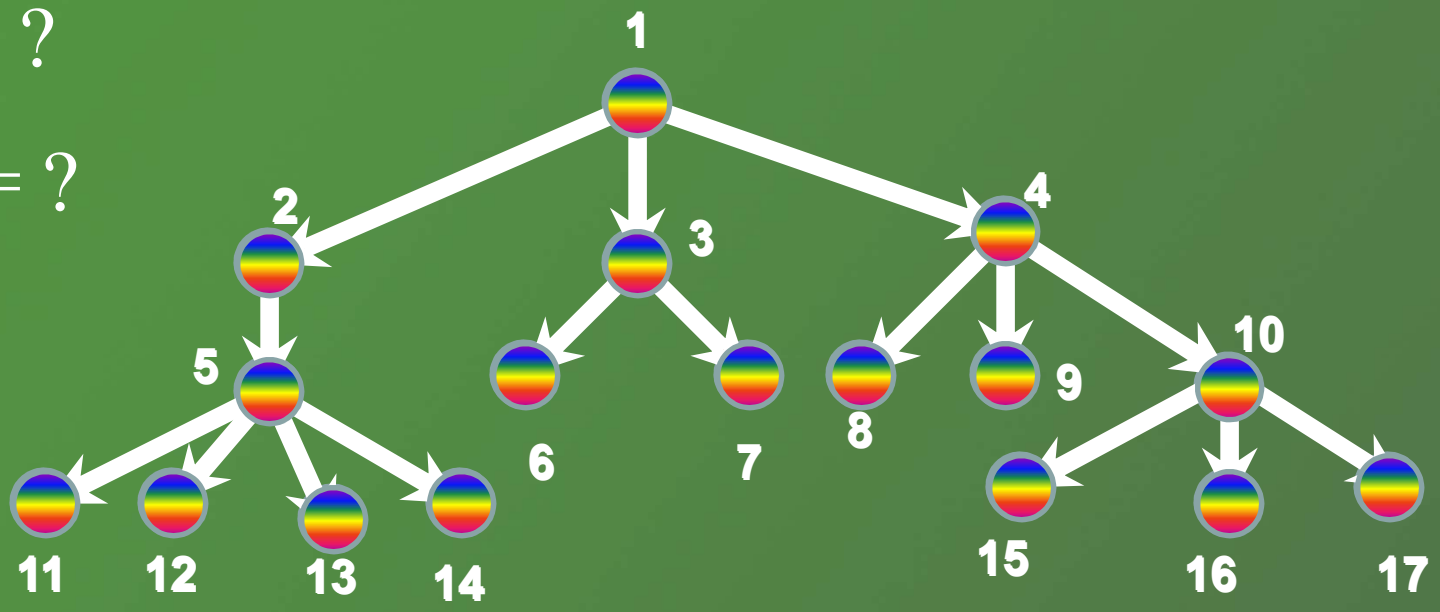
a) Độ dài đường đi trong là tổng tất cả các mức của các đỉnh trong, ký hiệu IP(T).

b) Độ dài đường đi ngoài là tổng tất cả các mức của các lá, ký hiệu EP(T).

Cây có hướng

IP(T) = ?

EP(T) = ?



Cây có hướng

Định lí

Cho T là cây nhị phân đủ với k đỉnh trong và s lá.

Ta có:

$$s = k+1 \text{ và } EP=IP+2k$$

Cây có hướng

Định nghĩa

Cho T là cây nhị phân không đủ. Lập T’ là cây có được bằng cách sau:

i. Thêm vào mỗi lá của T hai con.

ii. Thêm vào v một con nếu v là đỉnh trong của T mà chỉ có một con. Ta đặt:

IP(T) :=IP(T’)& EP(T):=EP(T’)

Phép duyệt cây(Tree travesal)

Định nghĩa

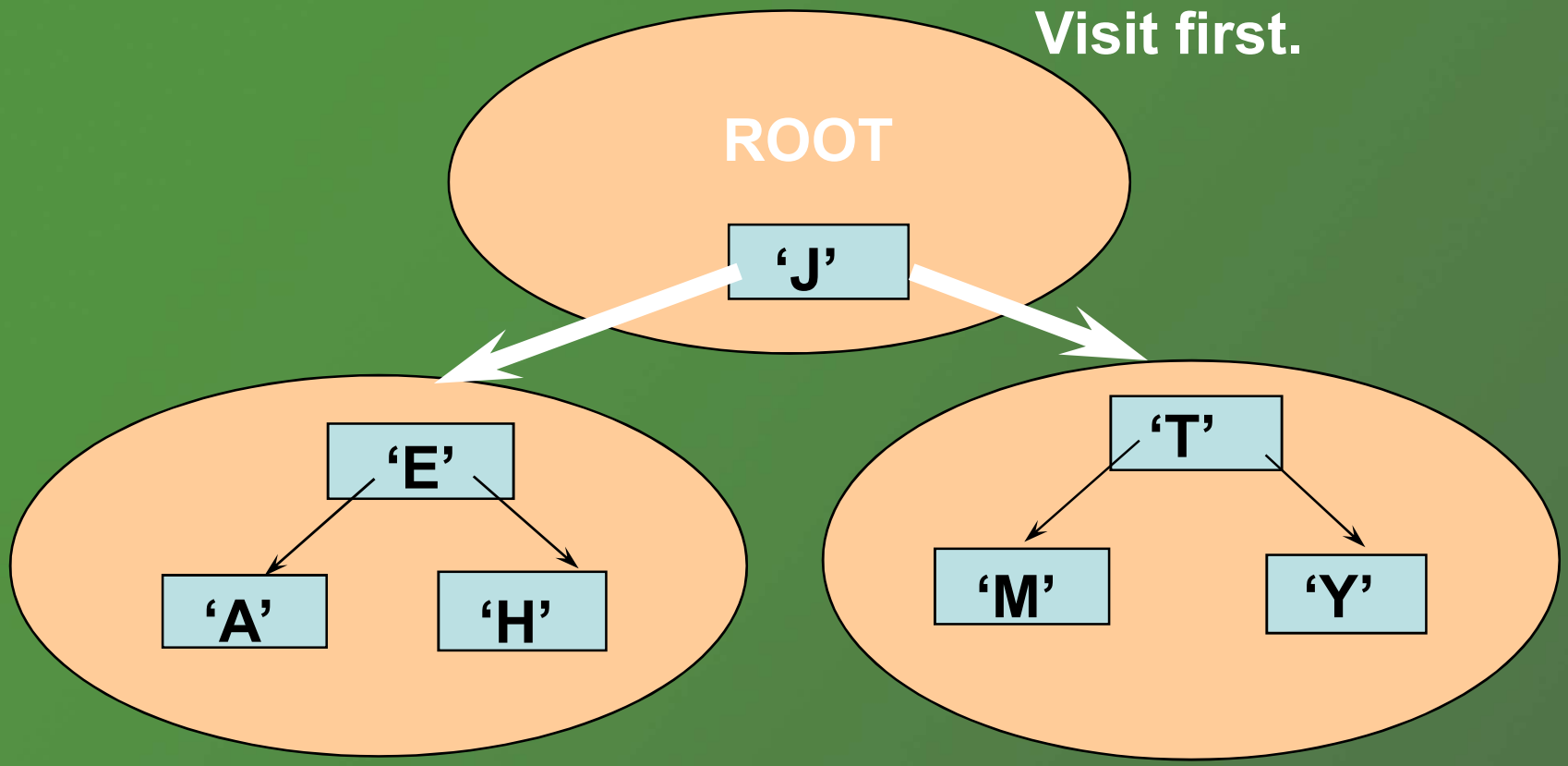
Duyệt cây là liệt kê tất các đỉnh của cây theo một thứ tự nào đó thành một dãy, mỗi đỉnh chỉ xuất hiện một lần .

Phép duyệt cây

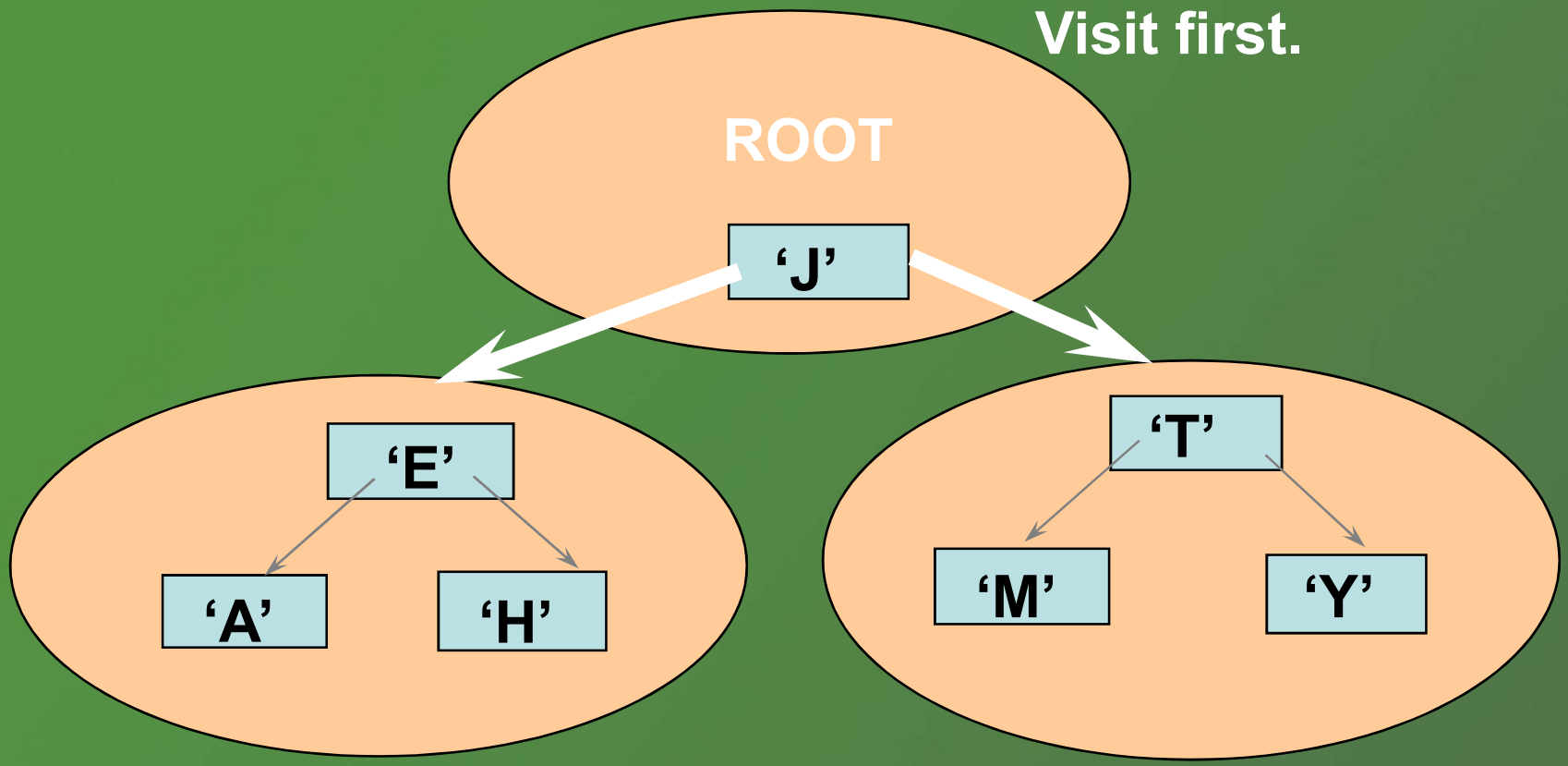
Phép duyệt tiền thứ tự (Preoder traversal)

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

Preorder Traversal: J E A H T M Y



Preorder Traversal: J E A H T M Y

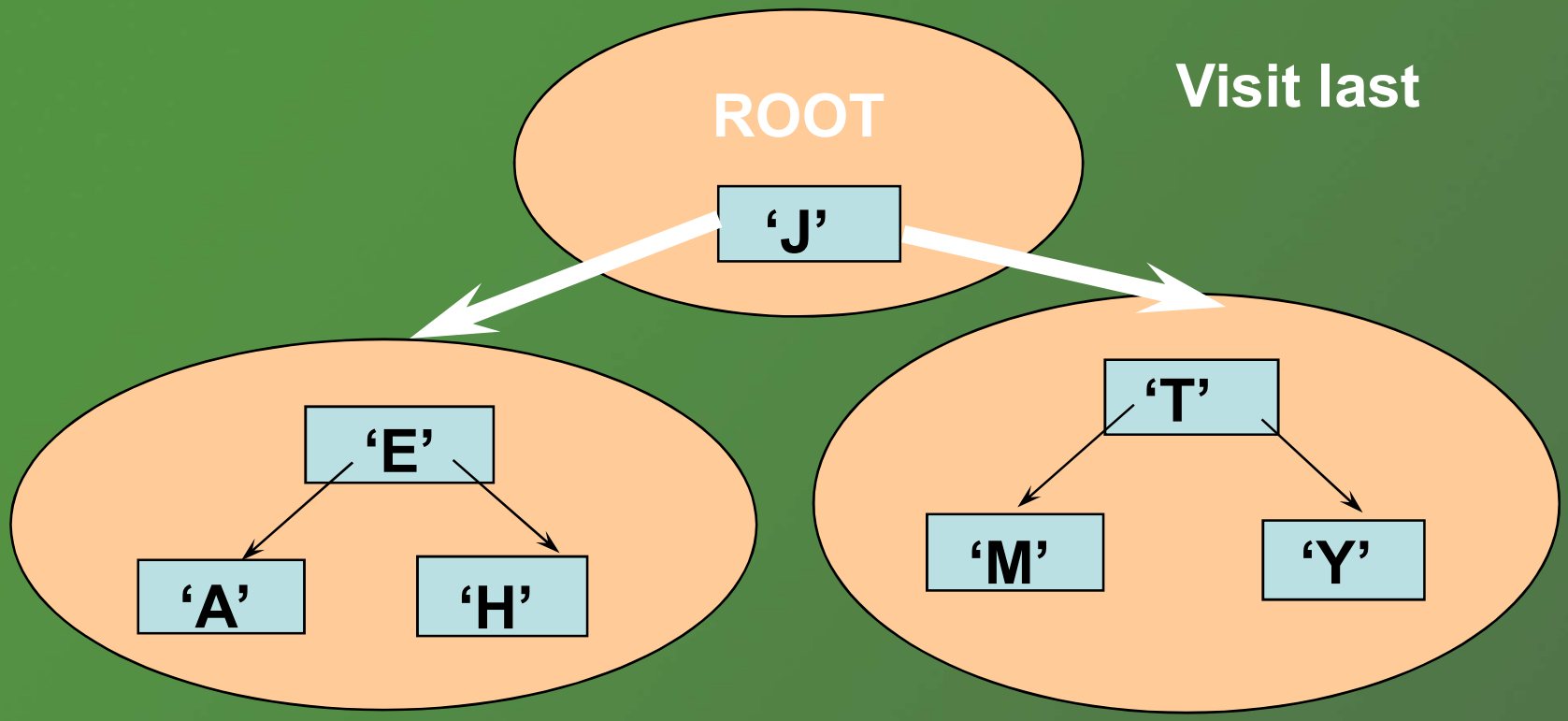


Phép duyệt cây

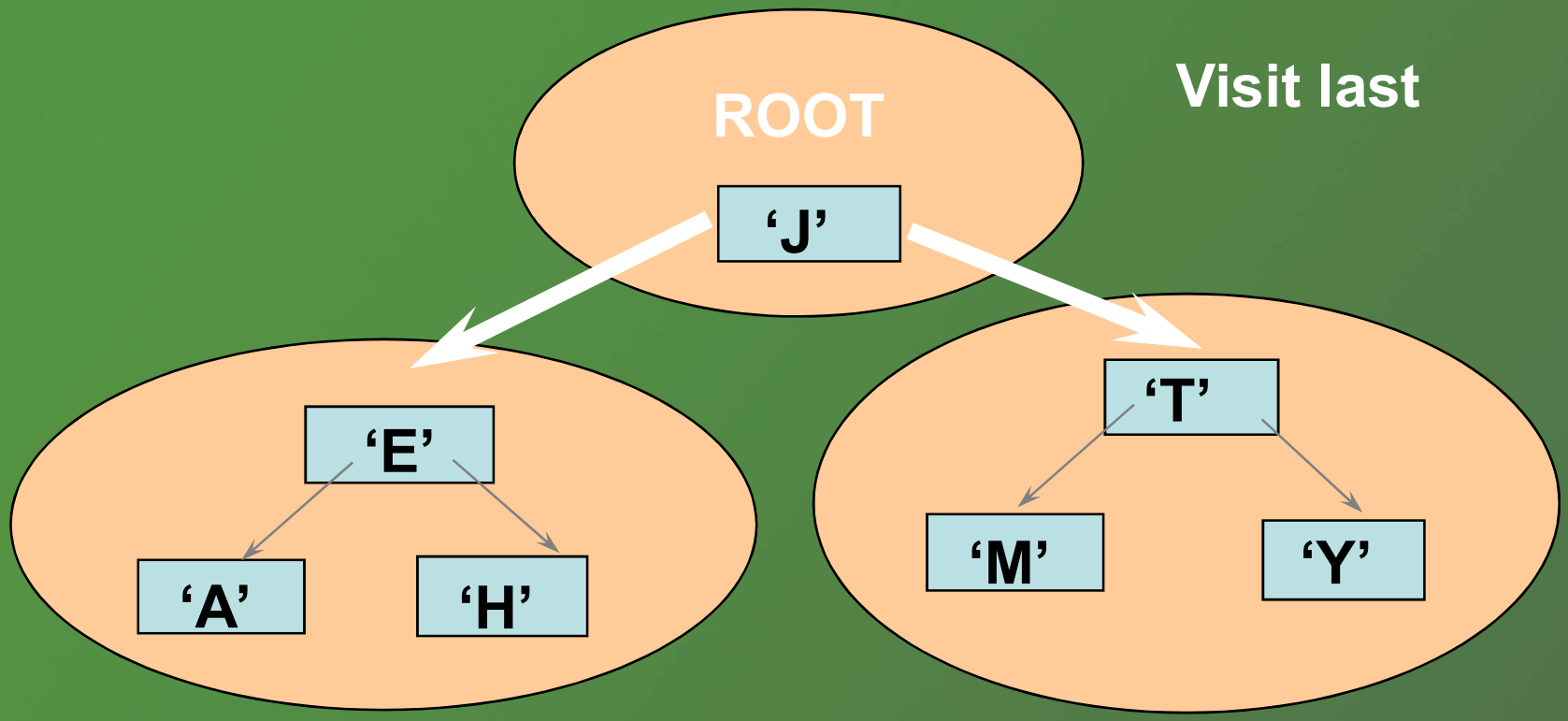
Phép duyệt hậu thứ tự (Posoder traversal).

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

Postorder Traversal: A H E M Y T J



Postorder Traversal: A H E M Y T J



Visit last

Visit left subtree in Postorder

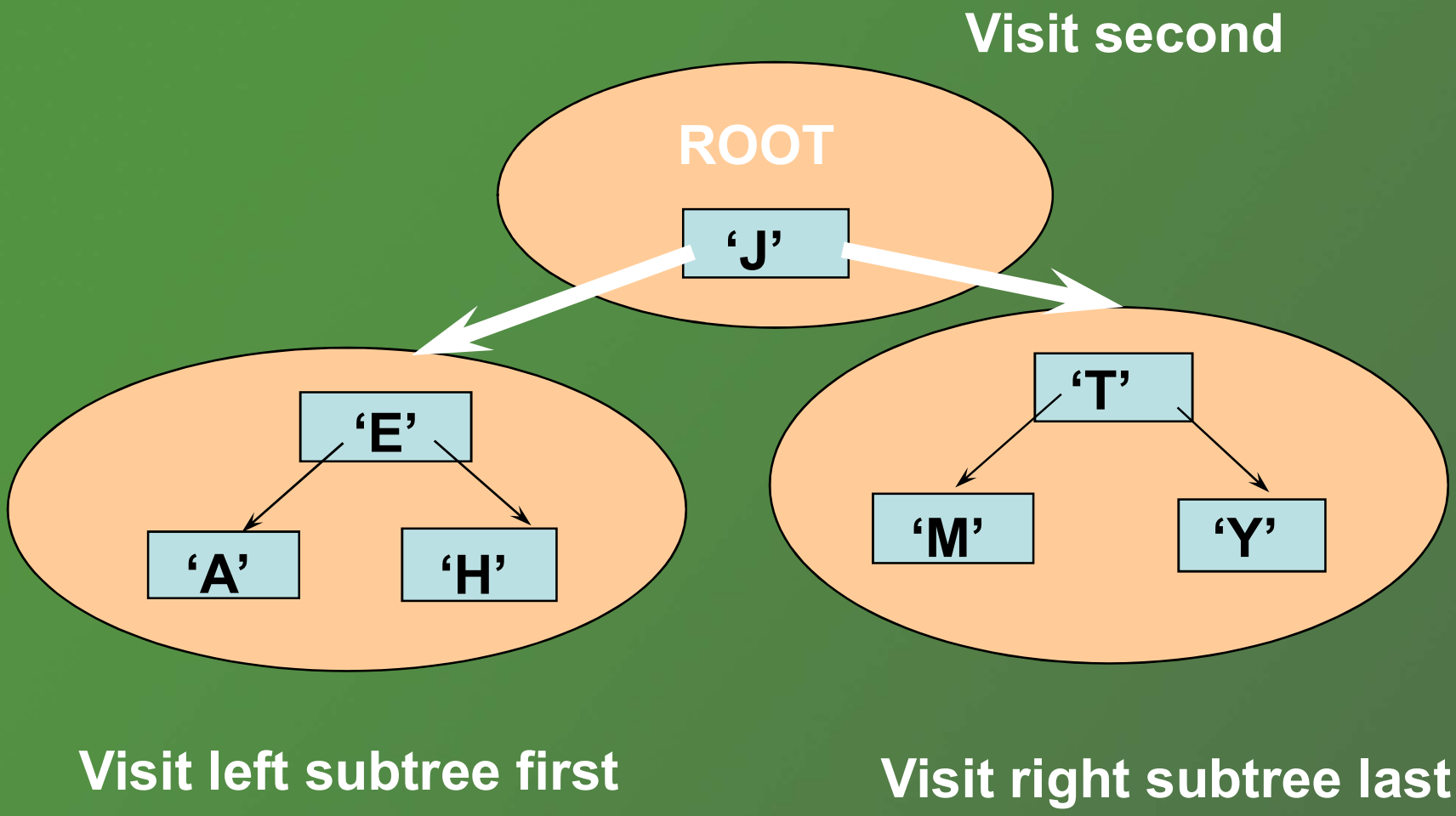
Visit right subtree in Postorder

Phép duyệt cây

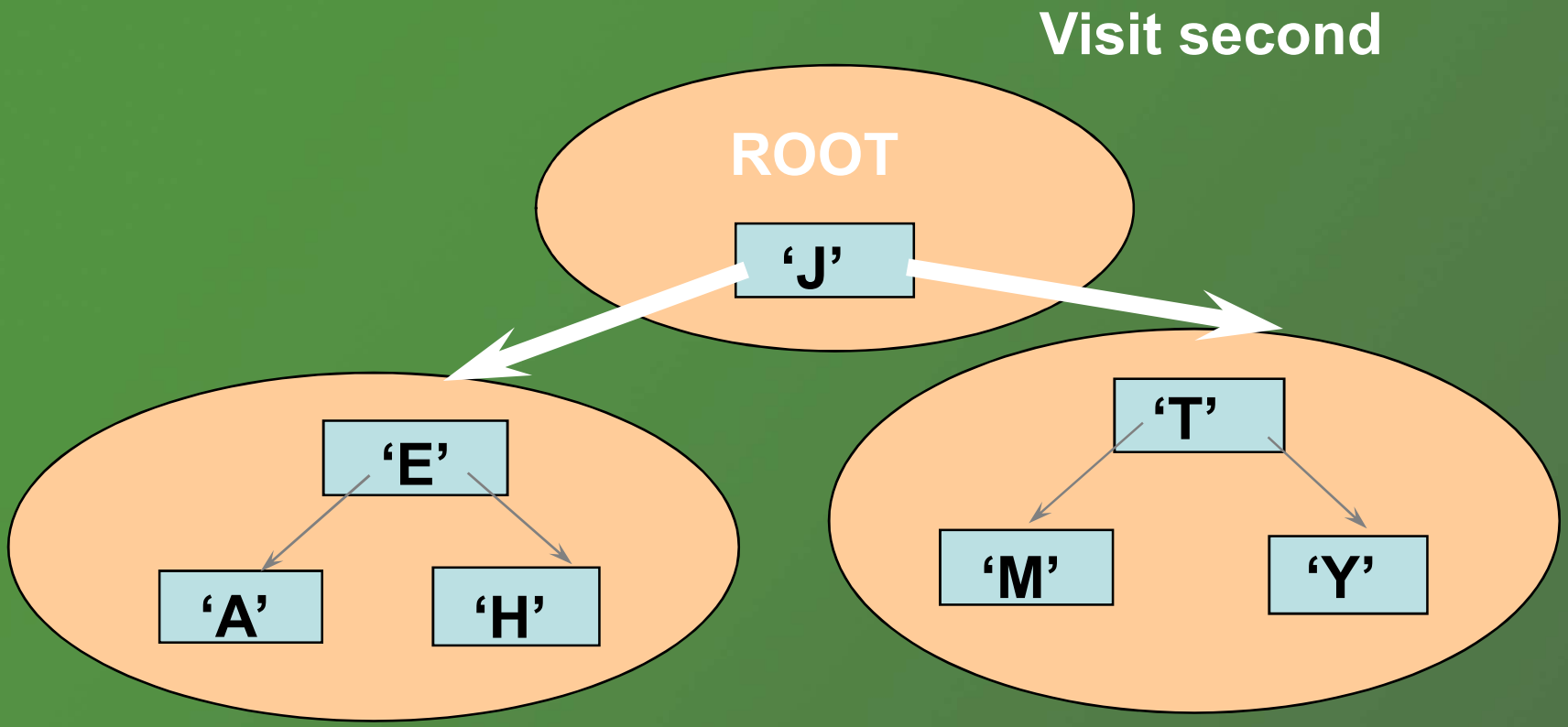
Phép duyệt trung thứ tự cho *cây nhị phân* (Inorder traversal)

1. Duyệt cây con bên trái T_L theo trung thứ tự.
2. Đến gốc r.
3. Duyệt cây con bên phải theo trung thứ tự.

Inorder Traversal: A E H J M T Y



Inorder Traversal: A E H J M T Y



Cây khung có hướng

Định nghĩa

Cho G(V,E) là đồ thị có hướng và T = (V,F) là đồ thị con khung của G. Nếu T là cây có hướng thì T gọi là cây khung có hướng(hay cây có hướng tối đại) của G.

Cây khung có hướng

Matrận Kirchhoff (G không khuyên)

a) Nếu G là đồ thị có hướng thì K(G) =(k_{ij})

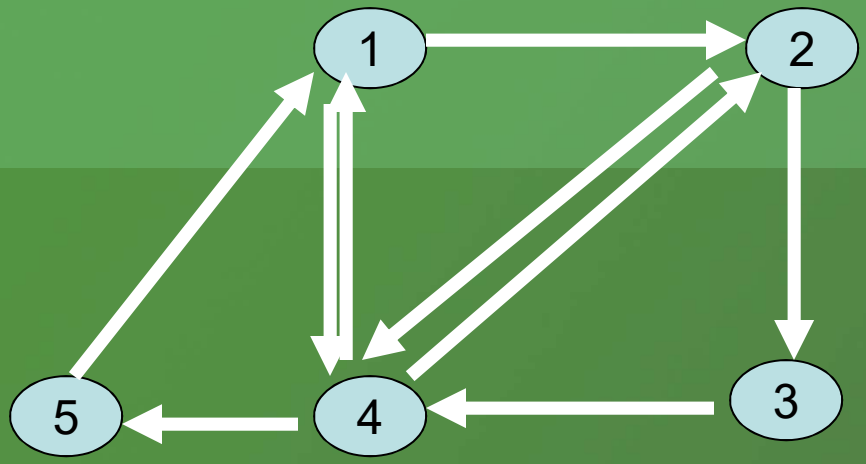
$$k_{ij} = \begin{cases} \deg^-(i) & khi\ i = j \\ -B_{ij} & khi\ i \neq j \end{cases}$$

trong đó B_{ij} là số cung đi từ i đến j

b) Nếu G là đồ thị vô hướng thì K(G) =(k_{ij})

$$k_{ij} = \begin{cases} \deg(i) & khi\ i = j \\ -B_{ij} & khi\ i \neq j \end{cases}$$

trong đó B_{ij} là số cung đi từ i đến j



$$\begin{pmatrix} 2 & -1 & 0 & -1 & 0 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ -1 & -1 & 0 & 3 & -1 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Cây khung có hướng

Định lý

- Cho G là đồ thị không khuyên. Đặt $K_q(G)$ là phần phụ của k_{qq}(Ma trận có được từ K(G) bằng cách xoá dòng q và cột q.
- Số cây khung có hướng trong G có gốc là đỉnh q bằng $detK_q(G)$.

Breadth-first Search Algorithm
Thuật toán ưu tiên chiều rộng

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 làm con của nó và các cạnh nối v_1 với chúng.

Những đỉnh này là đỉnh mức 1 trong cây.

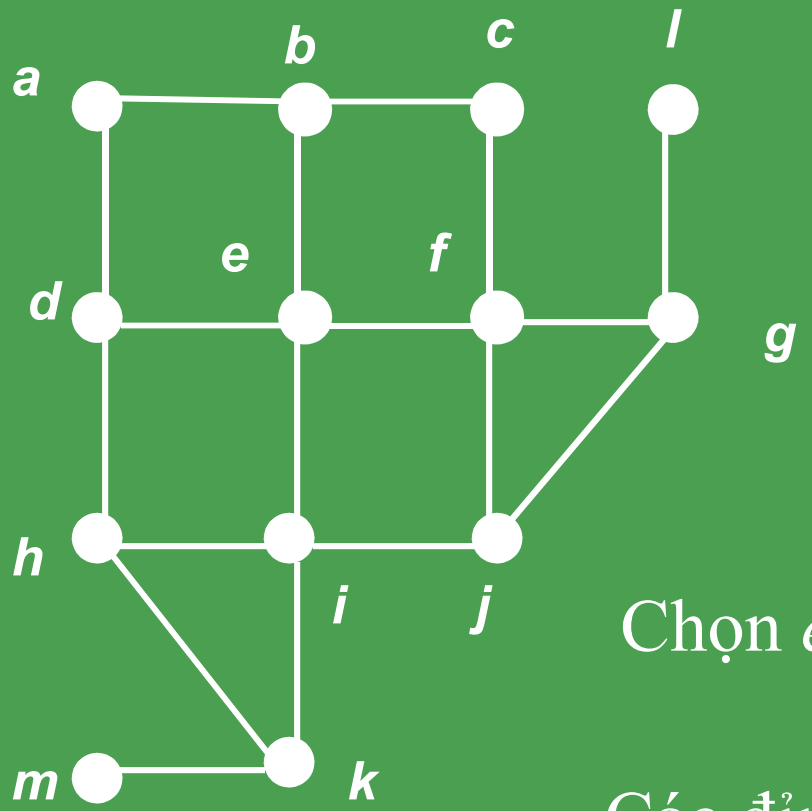
Bước 2: đối với mọi đỉnh v mức 1, thêm vào các cạnh kề với v vào cây sao cho không tạo nên chu trình đơn.

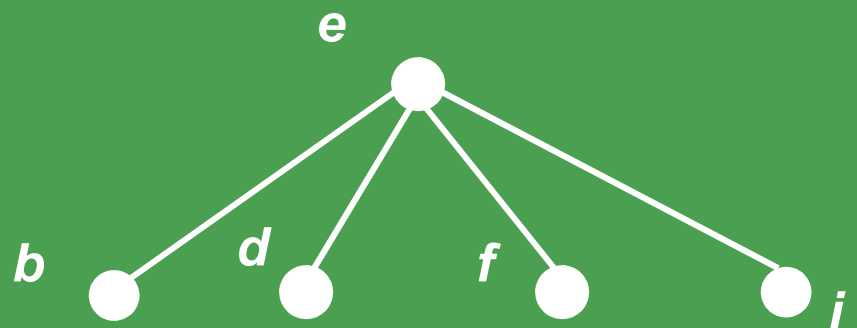
Thu được các đỉnh mức 2.

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

Cây T thu được là cây khung của đồ thị.

Ví dụ. Xét đồ thị liên thông *G*.

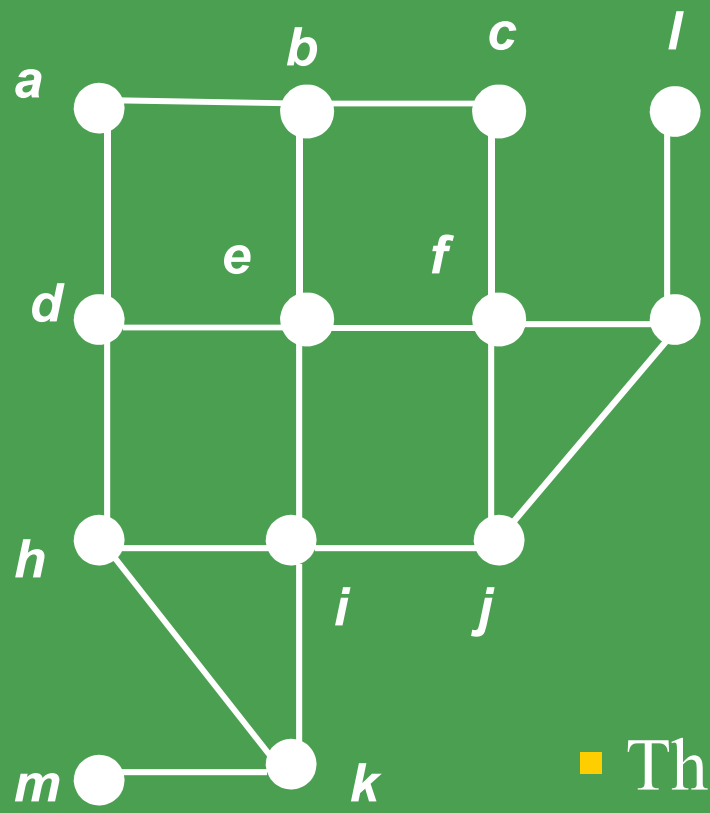


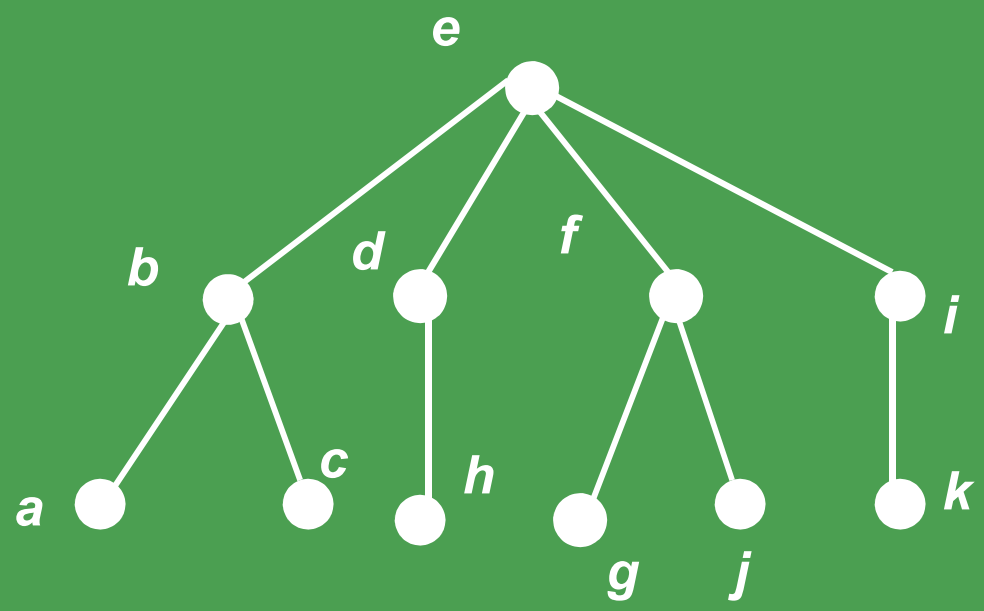


Chọn *e* làm gốc

Các đỉnh kề với *e* là con của nó.

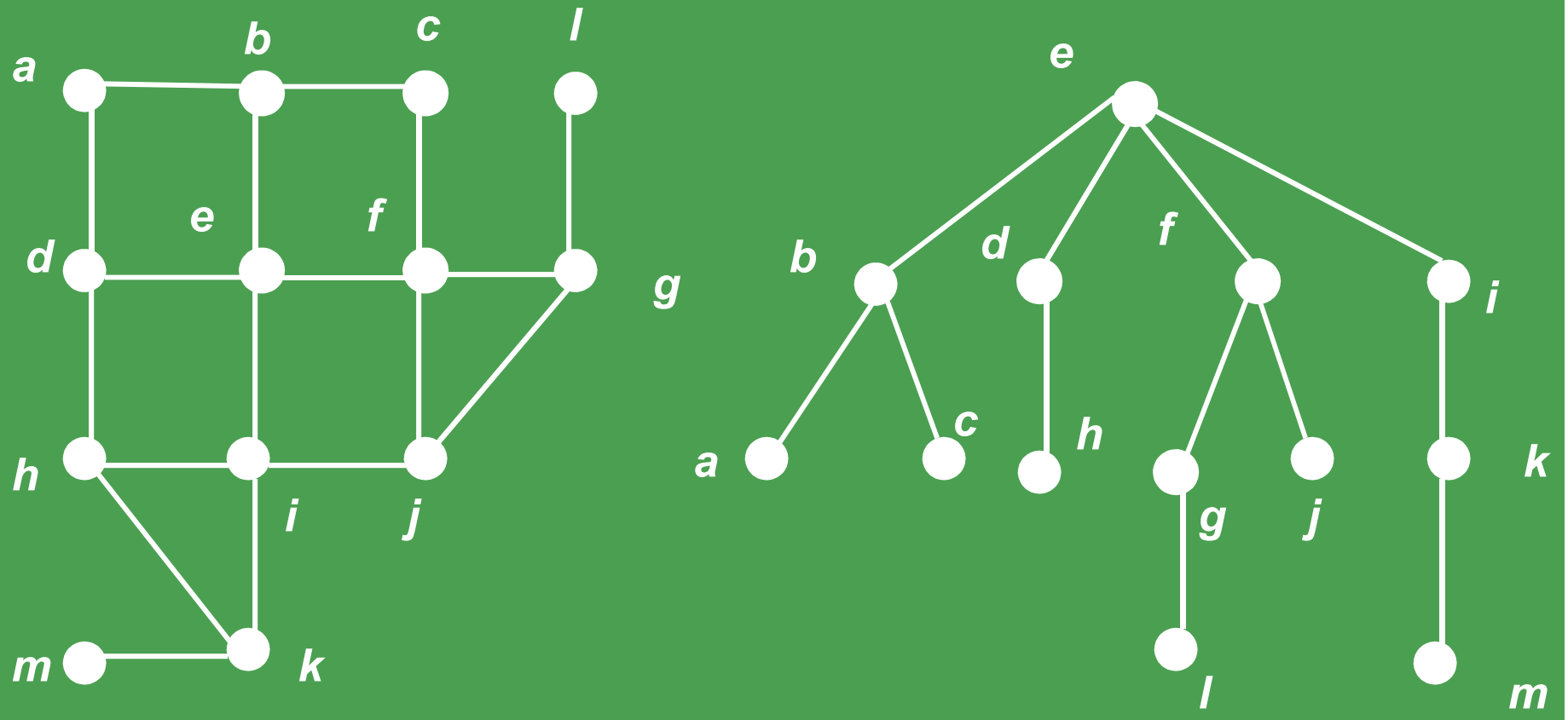
Các đỉnh mức 1 là: *b, d, f, i*





- 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*

Depth-first Search Algorithm
Thuật toán ưu tiên chiều sâu

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

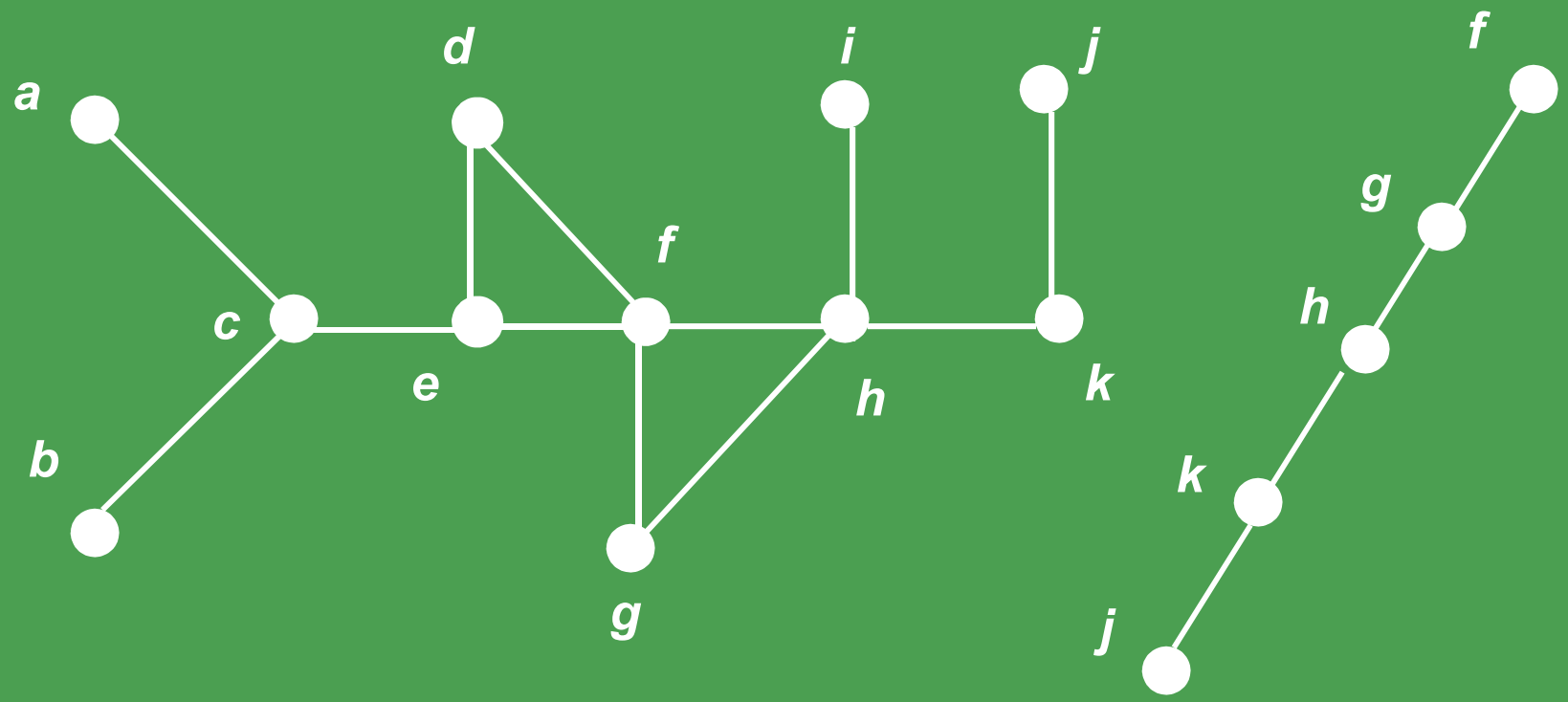
1. Chọn một đỉnh tùy ý của đồ thị làm gốc. Xây dựng đường đi từ đỉnh này bằng cách lượt ghép các cạnh sao cho mỗi cạnh mới ghép sẽ nối đỉnh cuối cùng trên đường đi với một đỉnh còn chưa thuộc đường đi.
2. Tiếp tục ghép thêm cạnh vào đường đi chừng nào không thể thêm được nữa.

i. Nếu đường đi qua tất cả các đỉnh của đồ thị thì cây do đường đi này tạo nên là cây khung.

ii. 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.

iii. Nếu điều đó không thể làm được thì lùi thêm một đỉnh nữa trên đường đi, tức là lùi hai đỉnh trên đường đi và thử xây dựng đường đi mới.

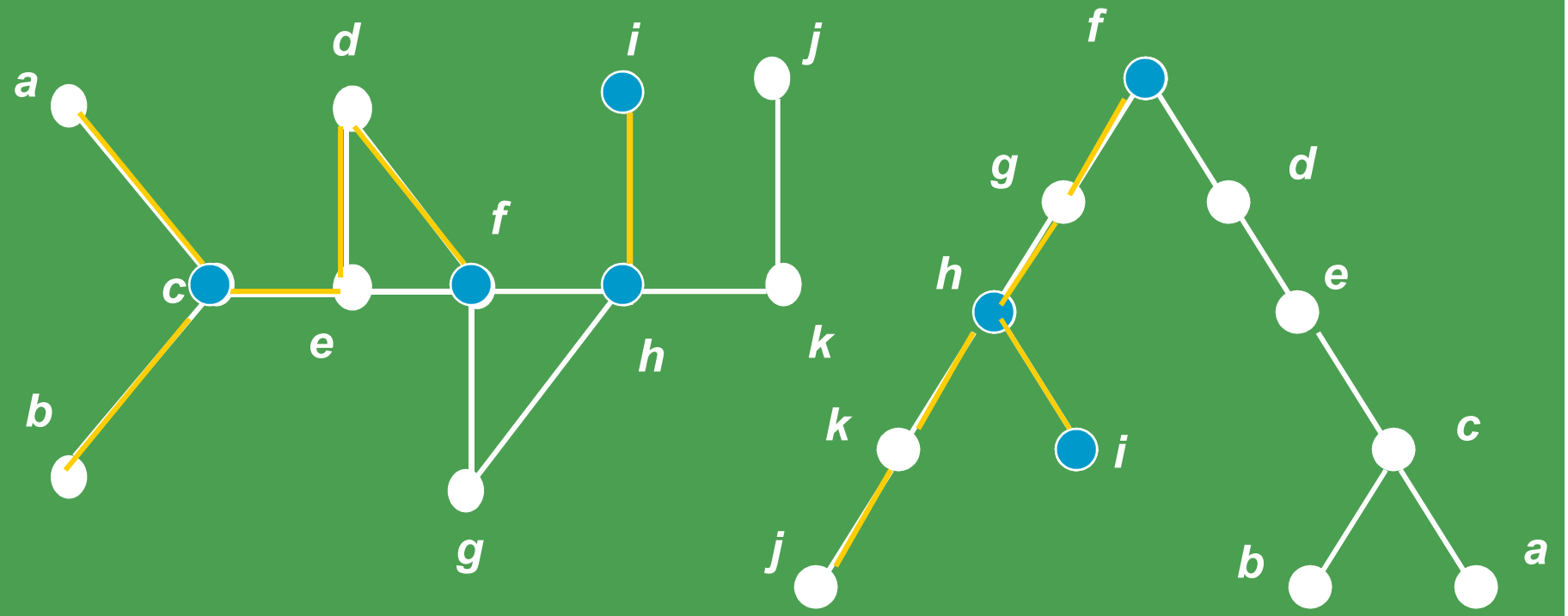
Ví dụ. Tìm cây bao trùm của đồ thị *G*.



Giải. Bắt đầu chọn đỉnh *f* làm gốc và

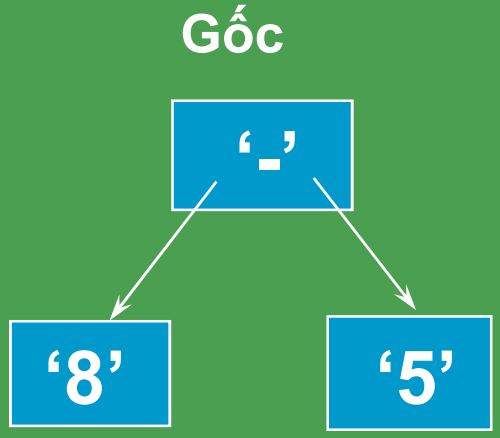
Thêm các hậu duệ của *f* : *g, h, k, j*

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

Cây nhị phân của biểu thức



INORDER TRAVERSAL: 8 - 5 có giá trị 3

PREORDER TRAVERSAL: - 8 5

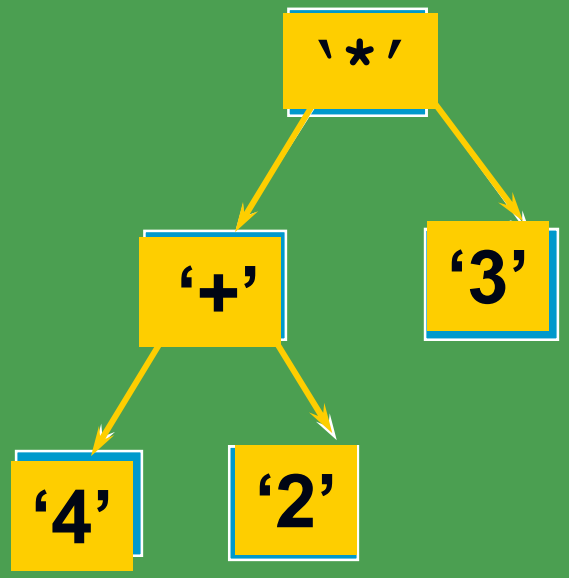
POSTORDER TRAVERSAL: 8 5 -

Định nghĩa

Cây nhị phân của biểu thức là cây nhị phân mà

1. Mỗi biến số được biểu diễn bởi một lá.
2. Mỗi đỉnh trong biểu diễn một phép toán với các thành tố là cây con tại đỉnh ấy.
3. 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.

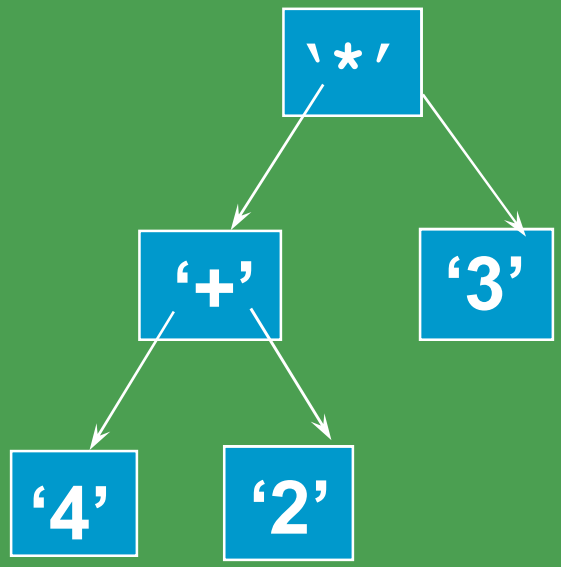
Cây nhị phân của biểu thức



Kết quả?

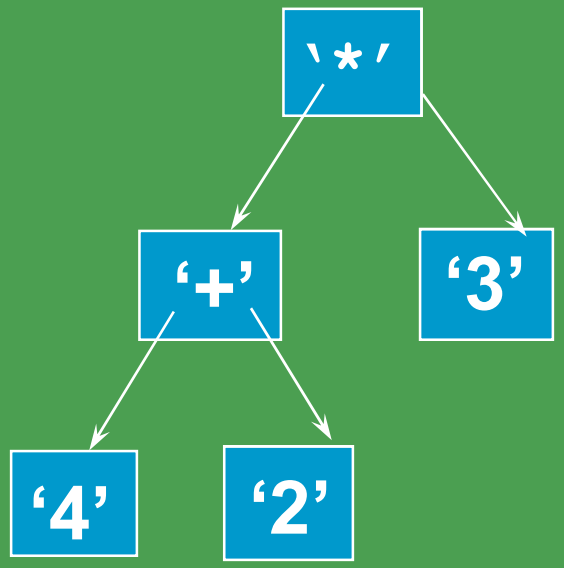
(4 + 2) * 3 = 18

Cây nhị phân của biểu thức



Dạng trung tố, tiền tố, hậu tố?

Cây nhị phân của biểu thức



Infix: ((4 + 2) * 3)

Prefix: * + 4 2 3 Ký pháp Ba lan : *từ phải sang trái*

Postfix: 4 2 + 3 * Ký pháp BL đảo : *từ trái sang phải*

Giải thích

Để có biểu thức theo ký pháp Ba lan, ta duyệt cây nhị phân của biểu thức bằng phép duyệt tiền thứ tự.

Thực hiện biểu thức 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.

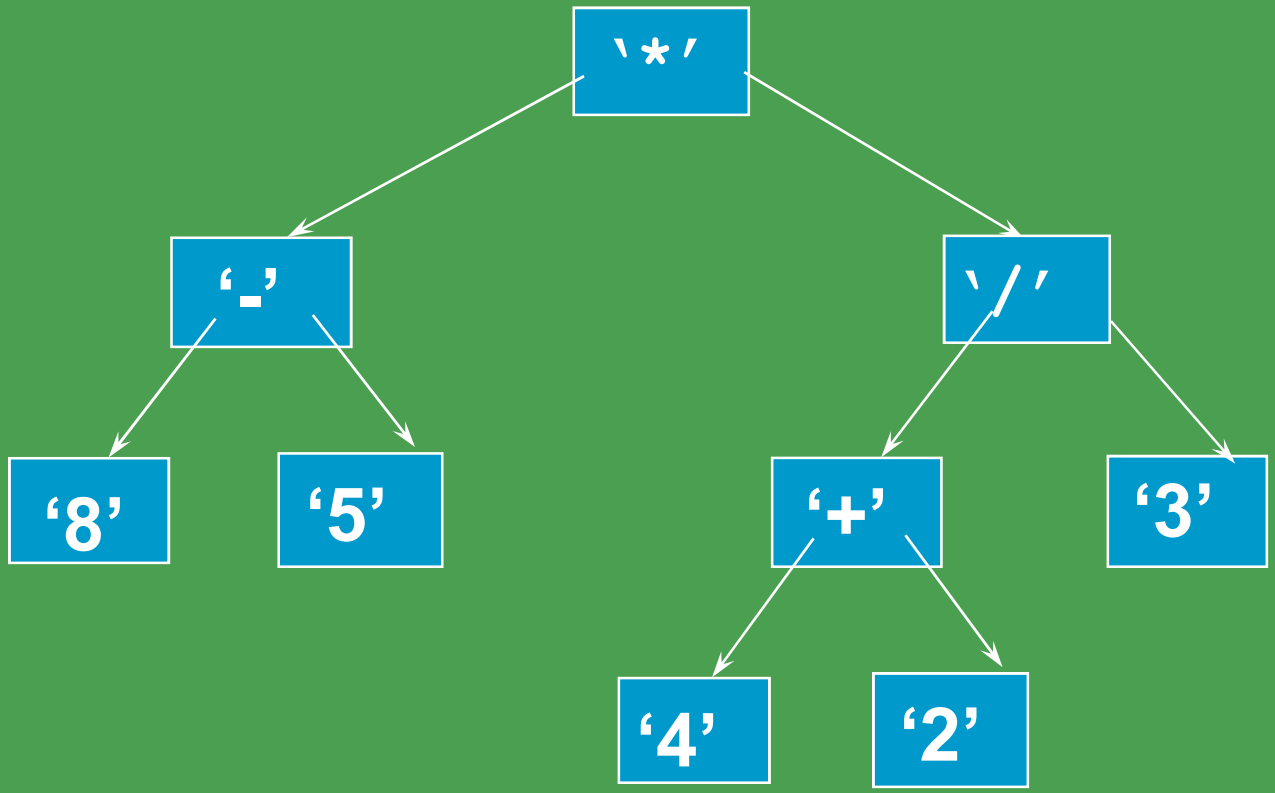
Giải thích

Để có biểu thức theo ký pháp Ba lan ngược, ta duyệt cây nhị phân của biểu thức bằng phép duyệt hậu thứ tự.

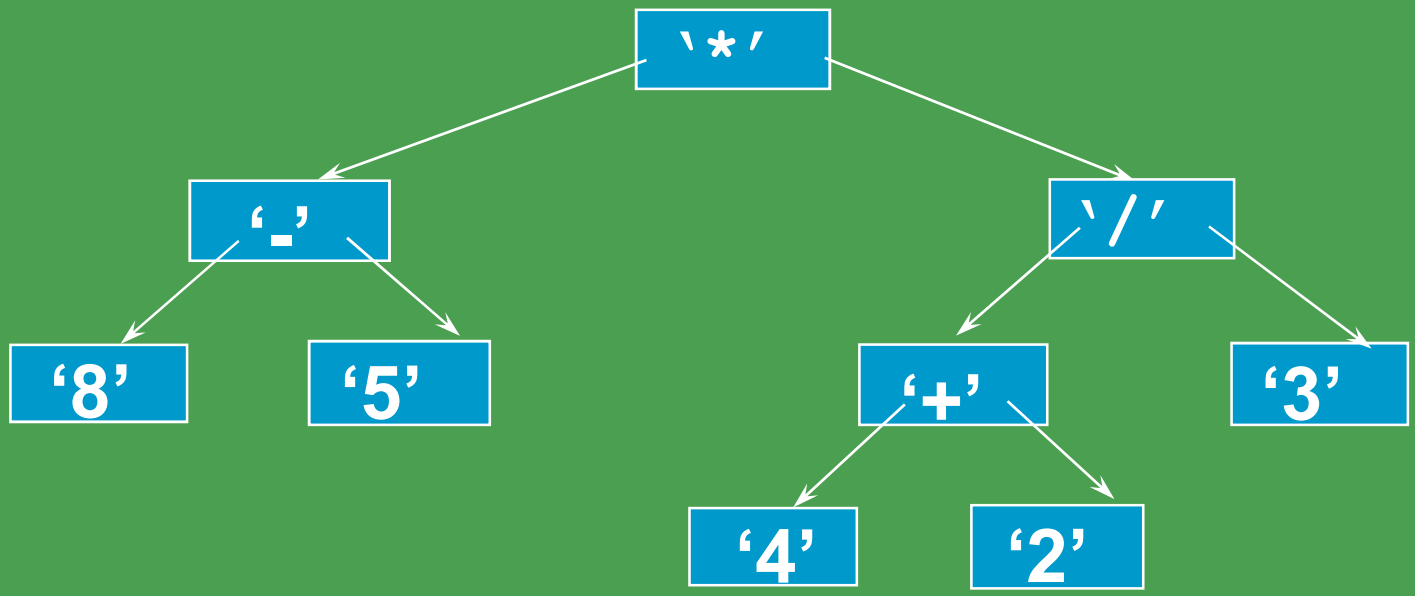
Thực hiện biểu thức từ trái sang phải:

Bắt đầu từ bên trái, khi gặp một phép toán thì phép toán này được thự hiện cho 2 thành tố ngay bên trái nó, kết quả này là thành tố cho phép toán tiếp theo.

Ví dụ



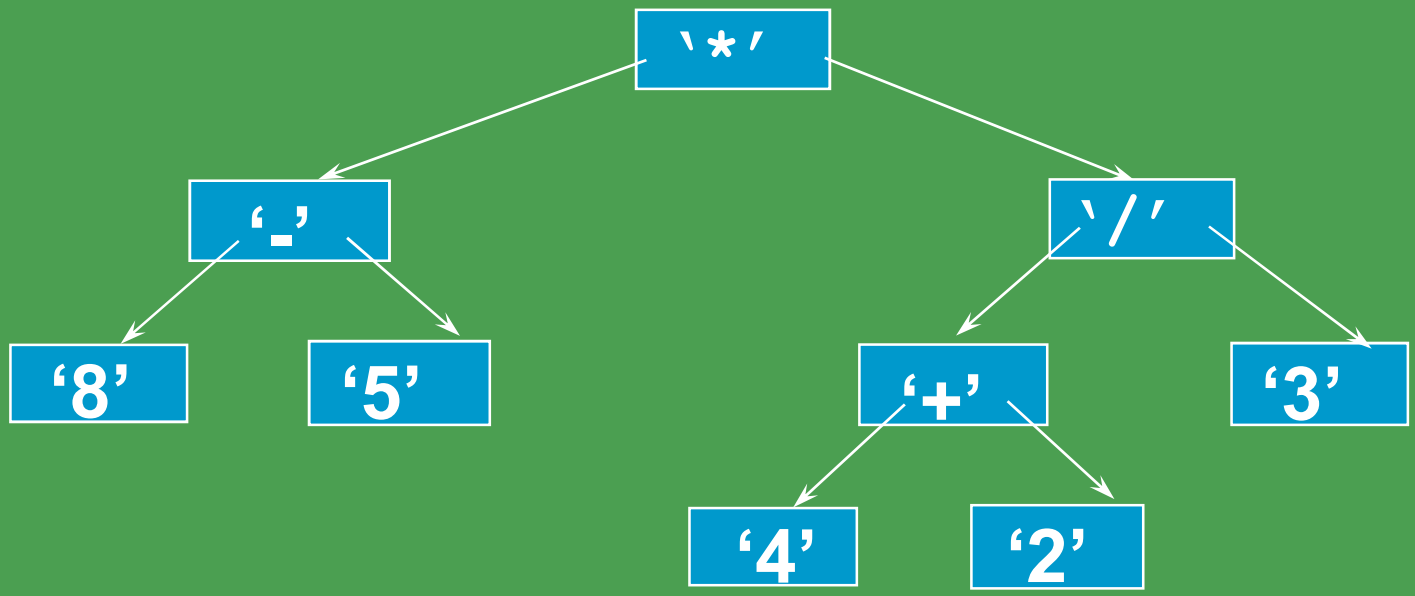
Kết quả của infix, prefix, postfix?



Infix: ((8 - 5) * ((4 + 2) / 3))

Prefix: * - 8 5 / + 4 2 3

Postfix: 8 5 - 4 2 + 3 / *

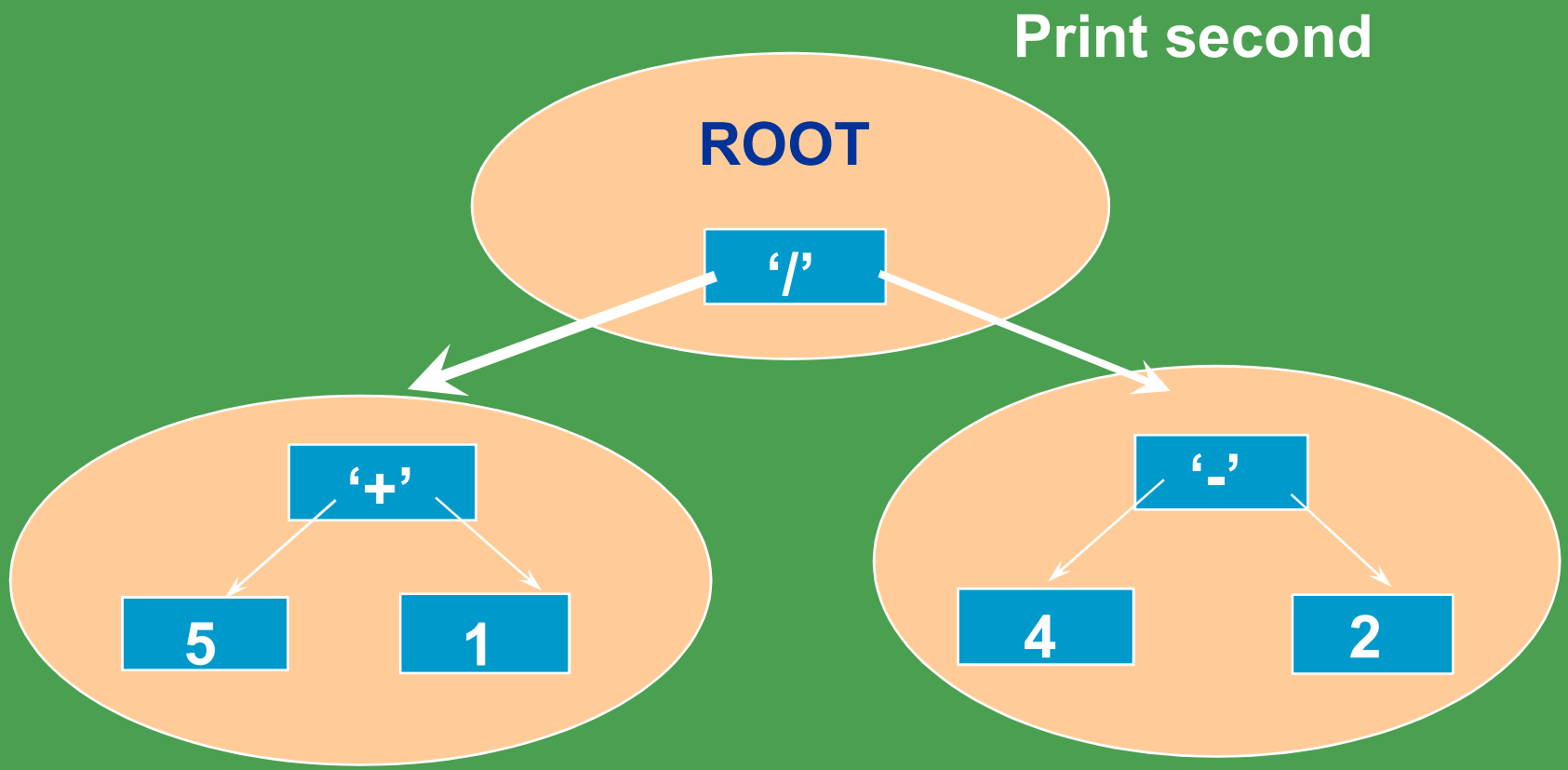


Infix: ((8 - 5) * ((4 + 2) / 3))

Prefix: * - 8 5 / + 4 2 3 *Thực hiện từ phải sang*

Postfix: 8 5 - 4 2 + 3 / * *Thực hiện từ trái sang*

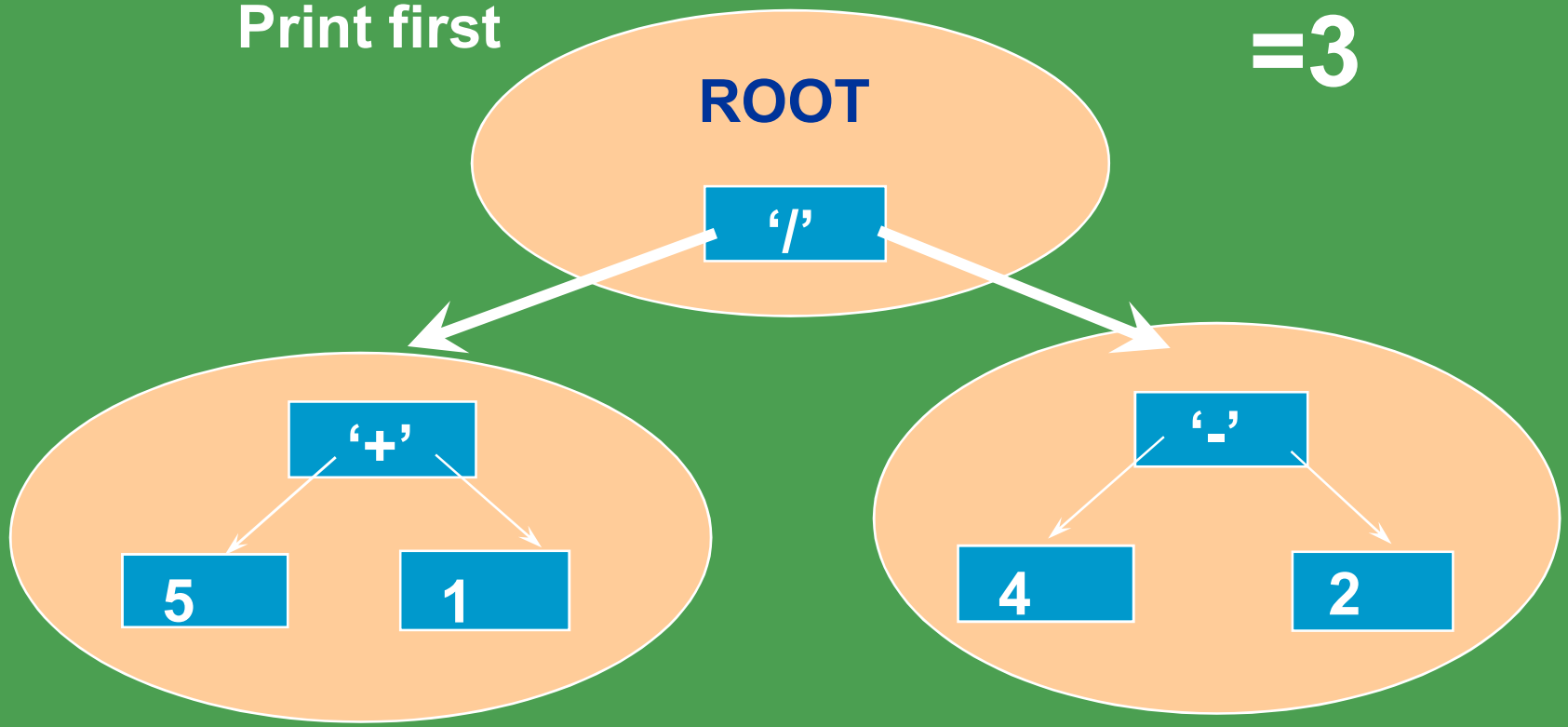
Inorder Traversal: (5 + 1) / (4 - 2) = 3



Preorder Traversal: / + 5 1 – 4 2

=/+51 2 = /62

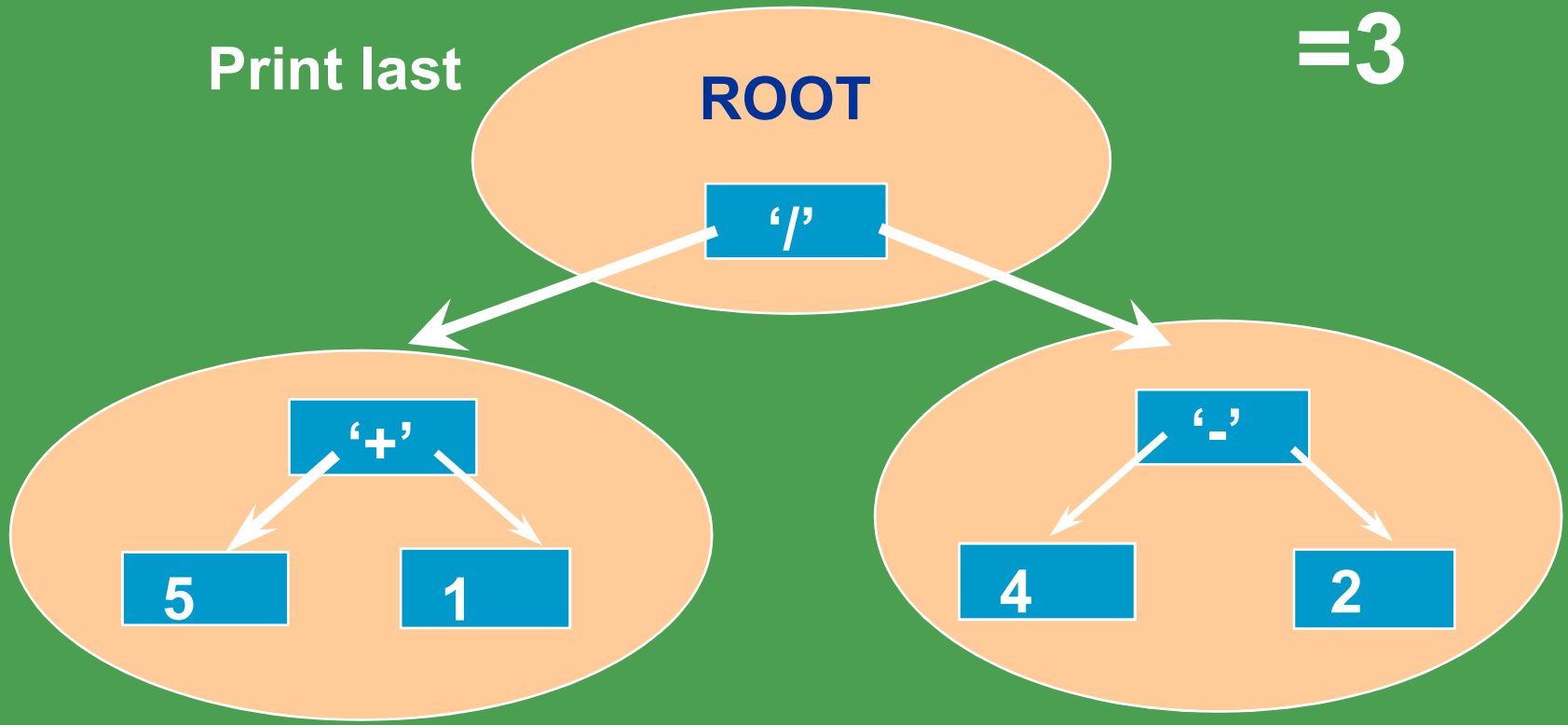
=3



Postorder Traversal: 5 1 + 4 2 – /=

= 6 4 2 –/ =6 2 /

=3



Print left subtree first

Print right subtree second