Chapter 11: Trees

Trần Hoà Phú

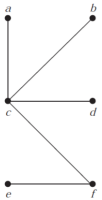
Ngày 26 tháng 6 năm 2023

Tree

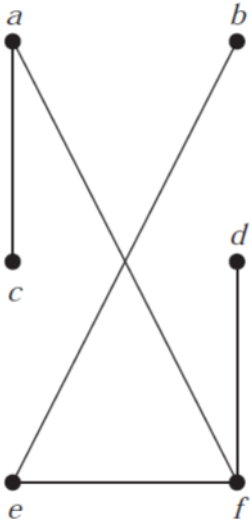
Definition A tree is a graph

- connected undirected (vô hướng, liên thông: giữa 2 đỉnh bất kỳ luôn có đường đi nối 2 đỉnh đó)
- no simple circuit (không có chu trình đơn: không có một đường đi khép kín mà không đi quá cạnh nào quá 1 lần)

Example 1 The graph G_1 is a tree?

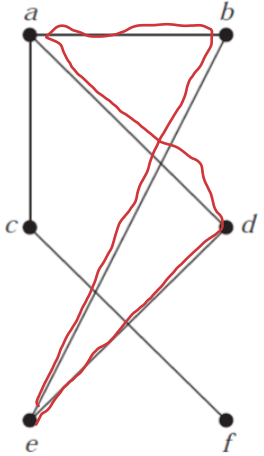


Example 2 The graph G_2 is a tree?



YES

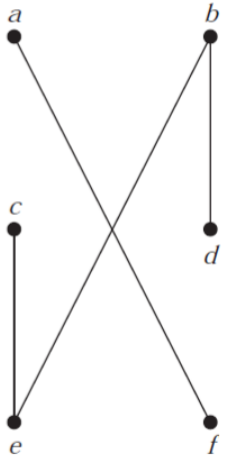
Example This graph is a tree?



KO

có đường đi khép kín

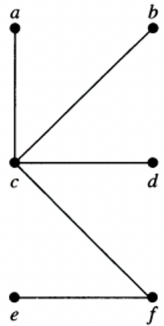
Example 4 This graph is a tree?



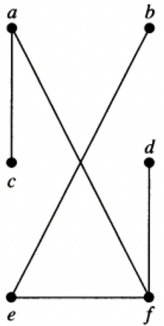
No

Theorem

An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.



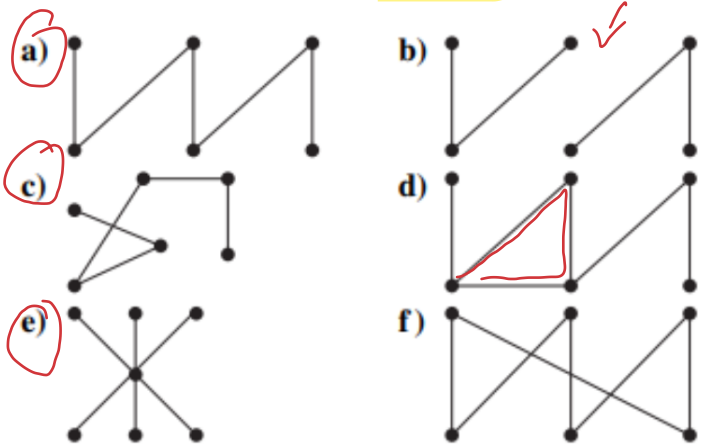
G_1



G_2

Exercise

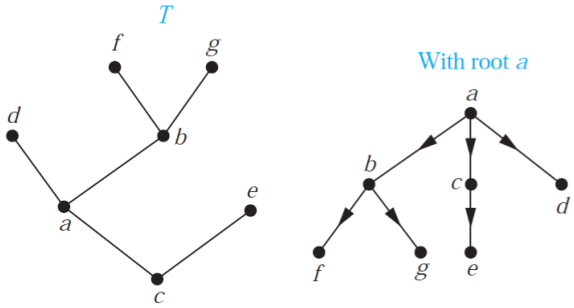
1. Which of these graphs are trees?

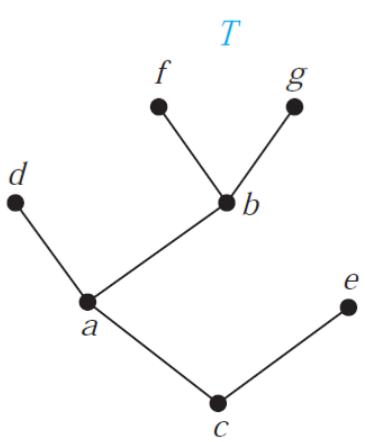


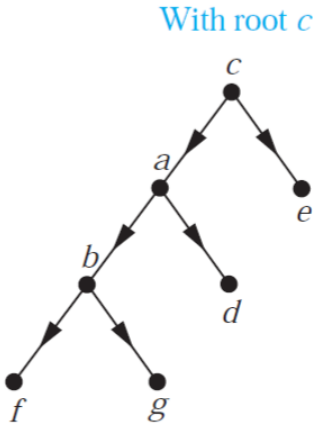
Definition

A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.

Example 1




T
f
g
d
b
e
a
c


With root c
c
a
e
b
d
f
g

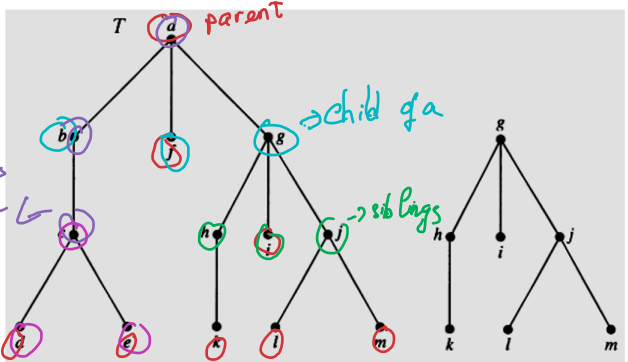


FIGURE 5 A Rooted Tree T.

FIGURE 6 The Subtree Rooted at g.

Some terminologies:

Page 686

- Subtree
- Root node (vertex)
- Internal node
- Leaf
- Parent
- Child
- Siblings
- Descendants
- Ancestors

In the tree G below:

The **parent** of c is b.

The children of g are h, i, j.

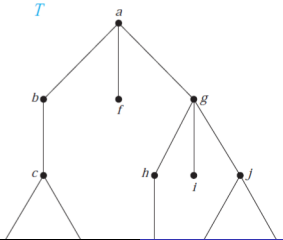
The **siblings** of h are i and j.

The **ancestors** of e are c, b, a.

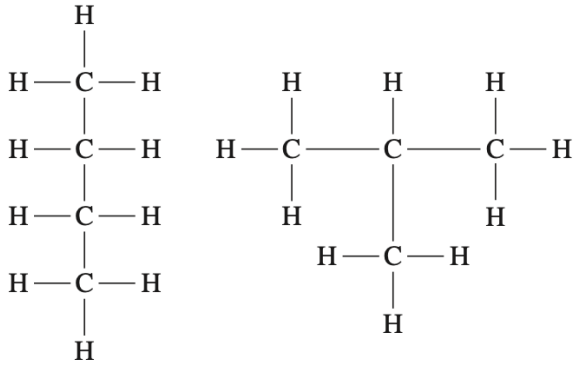
The **descendants** of b are c, d, e.

The internal vertices are a, b, c, g, h and j.

The leaves are d, e, f, i, k, l, m.



Tree as Models: Representing Molecules



Butane

Isobutane

CnH2n+2

(CH3)3CH

FIGURE 9 The Two Isomers of Butane.

Tree as Models: Representing Organizations

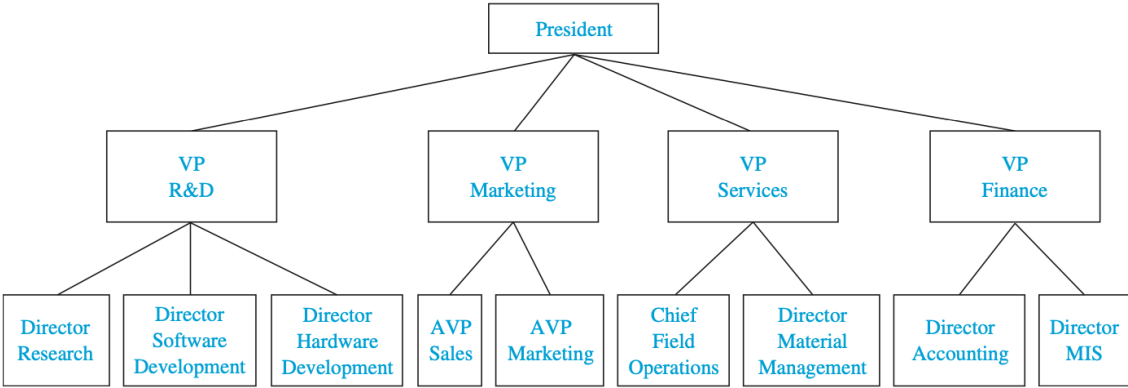


FIGURE 10 **An Organizational Tree for a Computer Company.**

Tree as Models: Family Tree

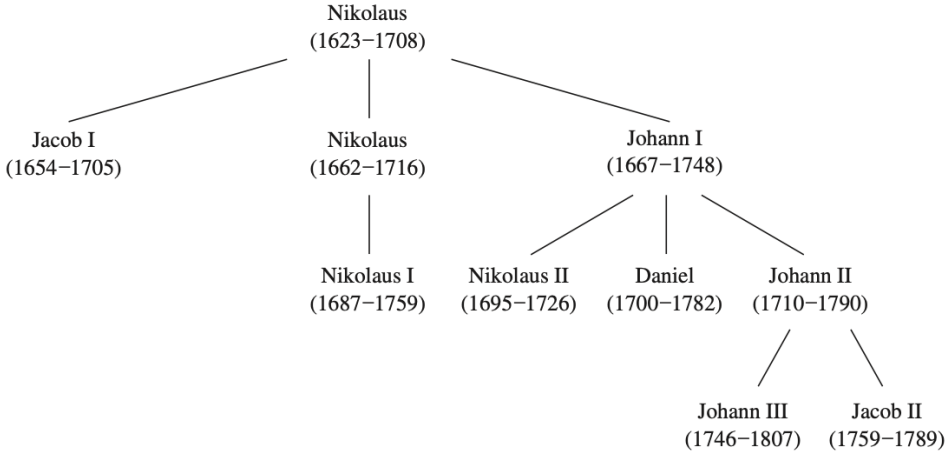


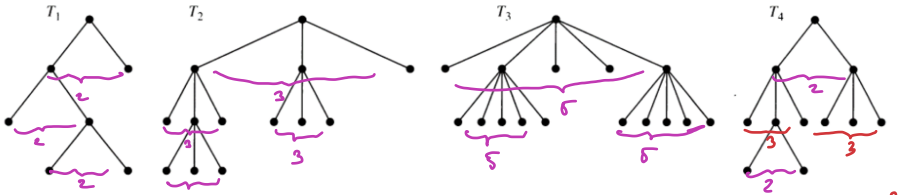
FIGURE 1 **The Bernoulli Family of Mathematicians.**

Introduction to Trees…

DEFINITION 3

A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children. The tree is called a *full m-ary tree* if every internal vertex has exactly m children. An m-ary tree with $m = 2$ is called a *binary tree*.

: m-ary: (≤ có quá m đỉnh con)



Some terminologies on binary tree:

Left child, right child, left subtree, right subtree

Theorem A tree with n vertices has n-1 edges

đỉnh = cạnh + 1
đỉnh = đỉnh trong + lá

$$n = e + 1$$

$$i + l = n$$

n: số đỉnh, e số cạnh, i số đỉnh trong, l số lá

Theorem

A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

con

$$n = m.i + 1$$ (đỉnh)

đỉnh trong

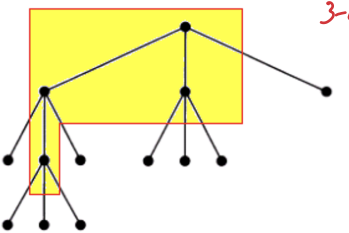
Introduction to Trees...

THEOREM 3

A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

3-ary

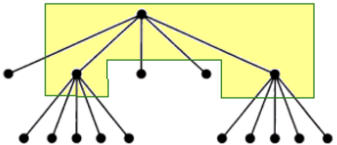
n = e + 1



m=3
i=4 (đỉnh trong)
n= 3.4+1=13 (đỉnh)

full - arry :

$e = m.i$



m=5
i=3
n= 3.5+1=16

a) n = e + 1
=> e = n - 1 = 999 (edges)

b) full 5-ary -> m = 5
i = 100 -> n = e + 1
= m.i + 1
= 501

c) full binary = 2-ary => m = 2
i = 1000
=> edge = m.i = 2000.

d) full 3-ary => m = 3
n = 100
l = ? (leaf)
n = m.i + 1
-> i = 33
n = i + l
-> l = n - i
= 67

Exercise

a) How many edges does a tree with 10000 vertices have?

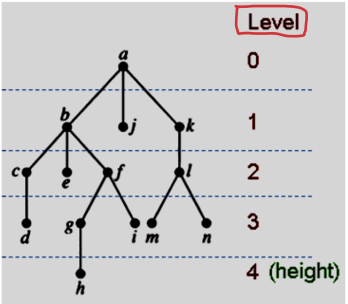
b) How many vertices does a full 5-ary tree with 100 internal vertices have?

c) How many edges does a full binary tree with 1000 internal vertices have?

d) How many leaves does a full 3-ary tree with 100 vertices have?

Introduction to Trees…

- **Level of a vertex: The length of the path from the root to this vertex.**
- **Height of Tree: The maximum of levels of vertices = The length of the longest path from the root to any vertex.**

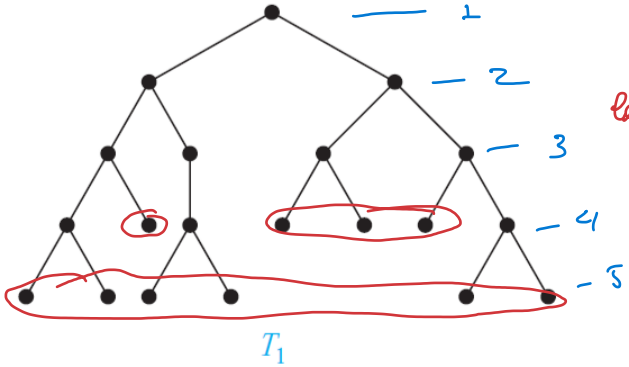


Height of tree = maximum level of vertices = 4

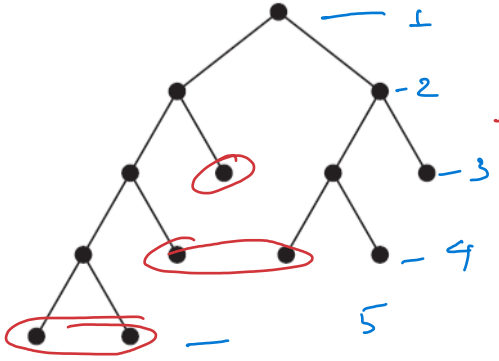
Definition

A rooted m-ary tree of height h is balanced if all leaves are at levels h or $h-1$.

Example 1 The rooted tree T_1 below is balanced ?



Example 2 The rooted tree T_2 is balanced?



T_2

Height levels: 5

leaves level: 3, 4, 5

⇒ not balanced

Theorem

There are at most m^h leaves in an m-ary tree of height h.

Theorem

- *If an m-ary tree of height h has l leaves, then $h \geq \lceil \log_m l \rceil$*
- *If the m-ary tree is full and balanced, then $h = \lceil \log_m l \rceil$*

m-ary tree $l \leq m^h \Leftrightarrow \lceil \log_m l \rceil \leq h$

vd: binary tree có 6 lá: $\Rightarrow h = \lceil \log_2 6 \rceil \geq 3$

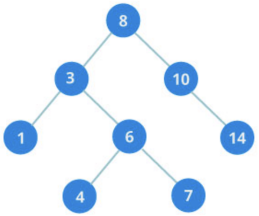
- Binary Search Tree
- Prefix Code

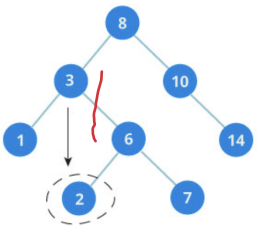
Problem How should items in a list be stored so that an item can be easily located?
This problem can be accomplished by the use of a Binary Search Tree

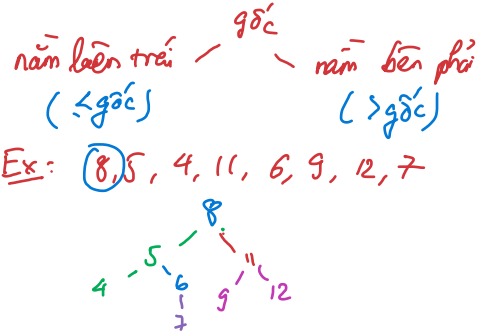
Binary Search Tree
(Cây tìm kiếm nhị phân)

"Cây tìm kiếm nhị phân" là cây nhị phân có thêm điều kiện:

- Giá trị của tất cả các Node ở cây con bên trái <= giá trị của Node gốc.
- Giá trị của tất cả các Node ở cây con bên phải > giá trị của Node gốc.
- Tất cả các cây con(bao gồm bên trái và phải) cũng đều phải đảm bảo 2 tính chất trên.





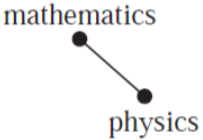


Comparison to locate "7": 4
(<8, >5, >6, =7)

Comparison to add "2": 3
(<8, <5, <4)

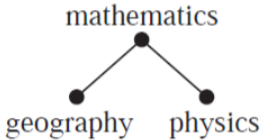
Example 1 Form a binary search tree for the words mathematics, physics, geography, zoology, meteorology, geology, psychology and *chemistry* (using alphabetical order).

mathematics, physics, geography, zoology, meteorology, geology, psychology and *chemistry*

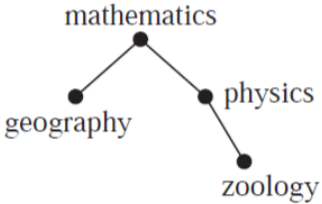


physics > mathematics

mathematics, physics, geography, zoology, meteorology, geology, psychology and *chemistry*

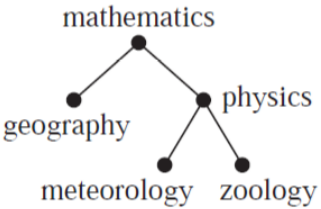


geography < mathematics

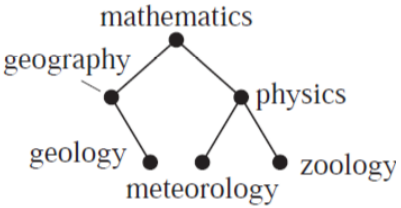


zoology > mathematics
zoology > physics

mathematics, physics, geography, zoology, meteorology, geology, psychology and *chemistry*

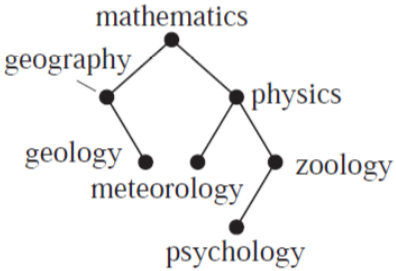


meteorology > mathematics
meteorology < physics

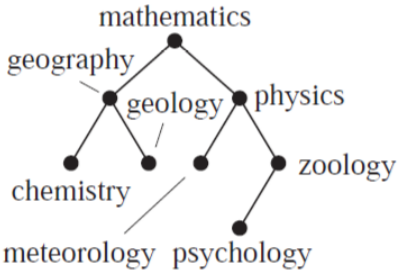


geology < mathematics
geology > geography

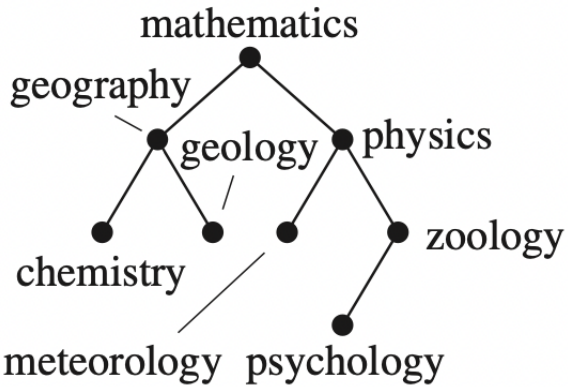
mathematics, physics, geography, zoology, meteorology, geology, psychology and *chemistry*



psychology > mathematics
psychology > physics
psychology < zoology



chemistry < mathematics
chemistry < geography



How to add "football"?

How many comparisons are needed to locate geology?

How many comparisons are needed to add "football"?

Exercise

1) Build a binary search tree for the words banana, peach, apple, pear, coconut, mango and papaya using alphabetical order.
2) How many comparisons are needed to locate or to add each of these words in the search tree , starting fresh each time?
a) pear
b) banana
c) kumquat
d) orange

ALGORITHM 1 Locating and Adding Items to a Binary Search Tree.

```
procedure insertion(T: binary search tree, x: item)
v := root of T
{a vertex not present in T has the value null}
while v ≠ null and label(v) ≠ x
begin
   if x < label(v) then
      if left child of v ≠ null then v := left child of v
      else add new vertex as a left child of v and set v := null
   else
      if right child of v ≠ null then v := right child of v
      else add new vertex as a right child of v to T and set v := null
end
if root of T = null then add a vertex v to the tree and label it with x
else if v is null or label(v) ≠ x then label new vertex with x and let v be this new vertex
{v = location of x}
```

Complexity: O(logn)
Proof: page 698

Prefix Codes

Standard ASCII character encoding: 8 bits to encode a character.

H: 01001000

e: 01100101

l: 01101100

o: 01101111

!: 00100001

"Hellooo!": $8 \times 8 = 64$ bits

H	000	0
e	001	1
l	010	2
o	011	3
!	100	4

H e l l o o o !

000	001	010	010	011	011	011	100

Only 24 bits needed to represent "Hellooo!"

=> mã tốt vì nó là prefix code.

Prefix Codes

Definition **Prefix codes are codes with the property: no bit string corresponds to more than one sequence of letters is to encode letters so that the bit string for a letter never occurs as the first part of the bit string for another letter.**

Example

$\{0, 10, 110, 101\}$ is not a prefix code.

$\{0, 1, 01\}$ is not a prefix code.

$\{0, 10, 110, 111\}$ is a prefix code.

Exercise

7. Which of these codes are prefix codes?

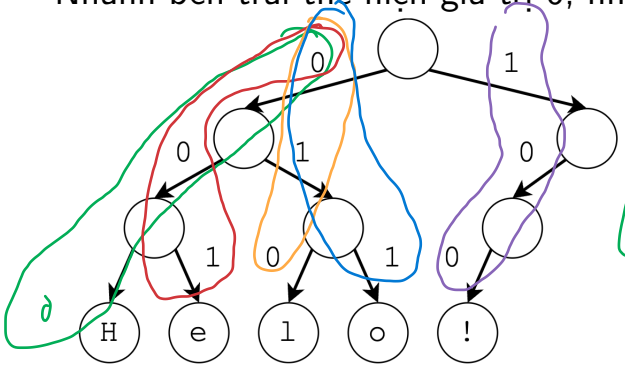
a) a: 11, e: 00, t: 10, s: 01

b) a: 0, e: 1, t: 01, s: 001

c) a: 101, e: 11, t: 001, s: 011, n: 010

d) a: 010, e: 11, t: 011, s: 1011, n: 1001, i: 10101

Prefix code có thể được biểu diễn thành cây nhị phân mã hóa.
Mỗi ký tự cần mã hóa sẽ nằm ở nút lá.
Giá trị mã hóa của ký tự đó thể hiện bằng đường đi từ nút gốc đến nút lá chứa ký tự đó.
Nhánh bên trái thể hiện giá trị 0, nhánh bên phải thể hiện giá trị 1.



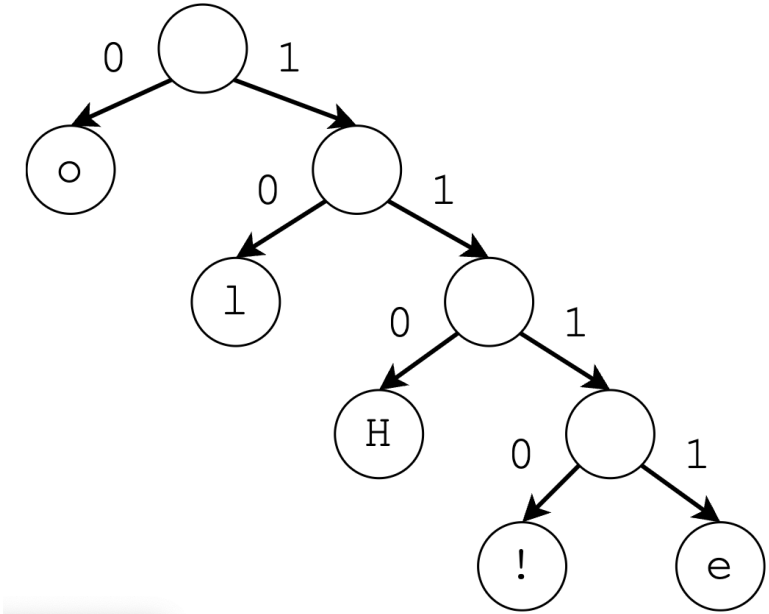
Chữ xuất hiện nhiều, gắn cho chúng một mã có độ dài ít nhất.
Chữ xuất hiện ít hơn, gắn cho chúng một mã có độ dài dài hơn.

H	110
e	1111
l	10
o	0
!	1110

H	e	l	l	o	o	o	!
110	1111	10	10	0	0	0	1110

"Hellooo!" : 18 bits

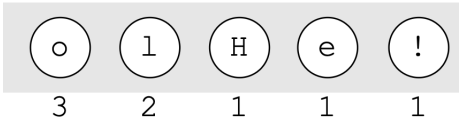
=> Prefix code.


0
1
o
0
1
l
0
1
H
0
1
!
e

Huffman Coding

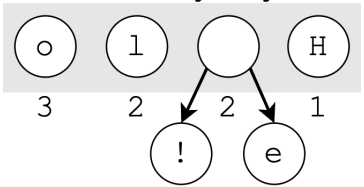
thuat toan mahoa data

Bước 1: Đếm tần số xuất hiện.

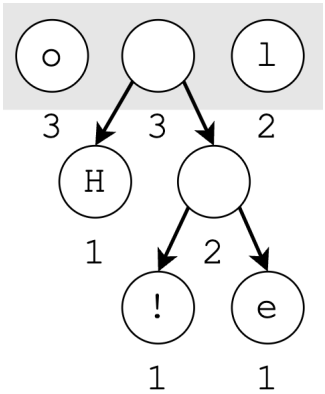


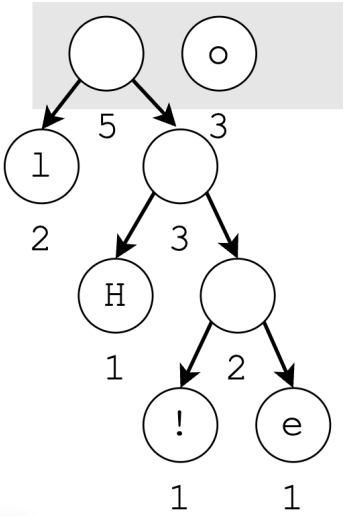
LL> Để tìm prefix code tối nhất (tốn ít bits nhất)

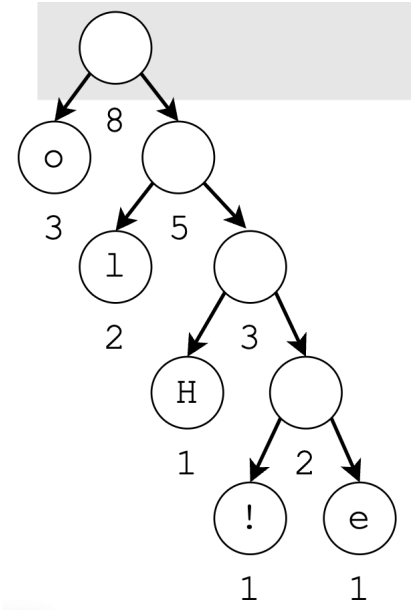
Bước 2: Gom 2 cây có trọng số nhỏ nhất thành một cây mới có trọng số = tổng 2 trọng số của 2 cây đó. Cây có trọng số nhỏ để bên trái, cây có trọng số lớn để bên phải. Lặp lại bước này cho đến khi còn 1 cây duy nhất.

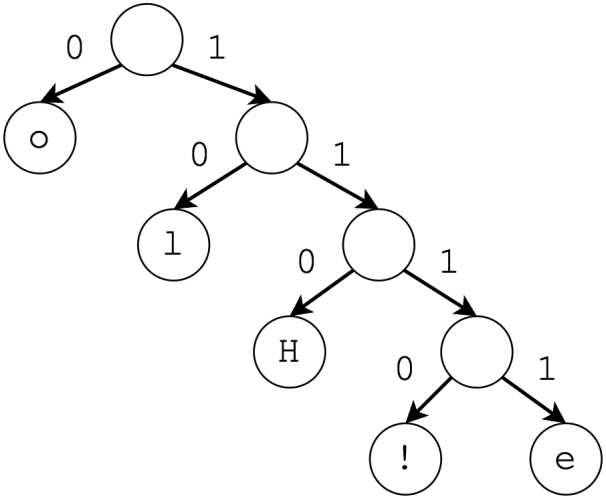


Hello000!
1 1 2 3 1


o
l
3
3
2
H
1
2
!
e
1
1


o
5
3
l
2
3
H
1
2
!
e
1
1


8
o
3
5
l
2
3
H
1
2
!
e
1
1


0
1
o
0
1
l
0
1
H
0
1
!
e

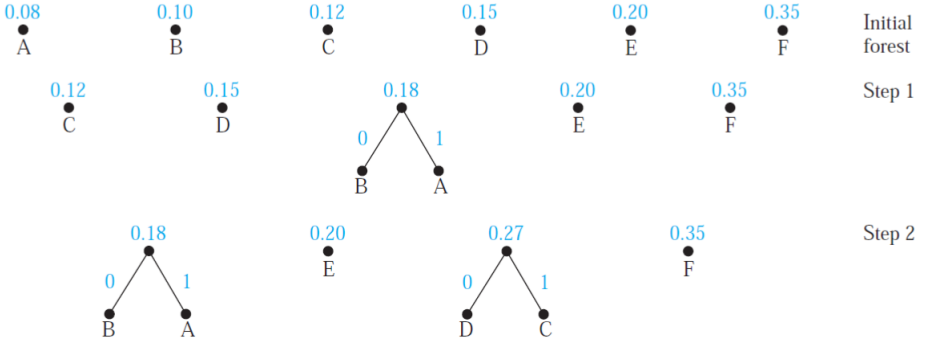
o : 0-1 bits (xuất hiện 3 lần)
l: 10- 2 bits (xuất hiện 2 lần)
H: 110 - 3 bits (xuất hiện 1 lần)
!: 1110: 4 bits (xuất hiện 1 lần)
e: 1111- 4 bits (xuất hiện 1 lần)

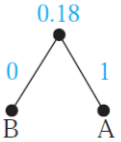
Số bit trung bình để mã hoá 1 ký tự = số lần xuất hiện x số bit / tổng số lần xh.

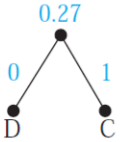
$$\frac{1 \times 3 + 2 \times 2 + 1 \times 3 + 4 \times 1 + 4 \times 1}{8} = 2.25$$

Example 1

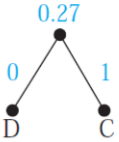
Use Huffman Coding to encode the following symbols with the frequencies listed
$A : 0.08, B : 0.1, C : 0.12, D : 0.15, E : 0.20, F : 0.35$. What is the average number of bits used to encode a character?

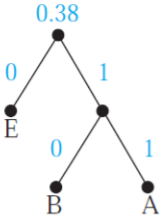

0.08
A
0.10
B
0.12
C
0.15
D
0.20
E
0.35
F
Initial forest
0.12
C
0.15
D
0.18
0
1
B
A
0.20
E
0.35
F
Step 1
0.18
0
1
B
A
0.20
E
0.27
0
1
D
C
0.35
F
Step 2



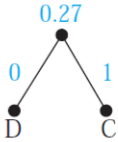


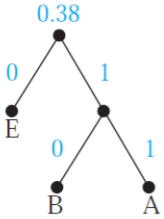
Step 2



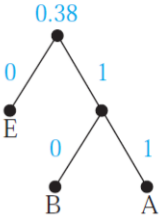


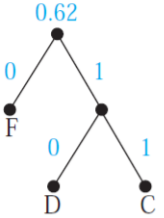
Step 3



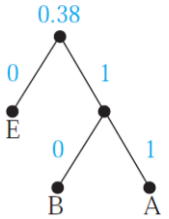


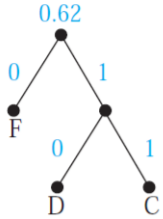
Step 3

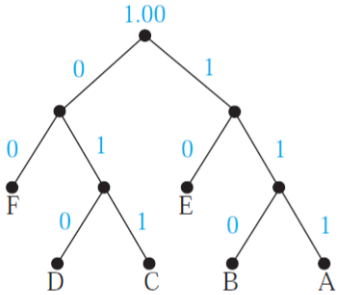




Step 4


0.38
0
1
E
0
1
B
A


0.62
0
1
F
0
1
D
C


1.00
0
1
0
1
0
1
F
0
1
E
0
1
D
C
B
A

A : 111 - 3 bits (frequency:0.08)
B : 110 - 3 bits (frequency: 0.10)
C : 011 - 3 bits (frequency: 0.12)
D : 010 - 3 bits (frequency: 0.15)
E : 10 - 2 bits (frequency: 0.2)
F : 00 - 2 bits (frequency: 0.35)

Then, the **average number of bits** used to encode a symbol using this coding is

$$3 \times 0.08 + 3 \times 0.10 + 3 \times 0.12 + 3 \times 0.15 + 2 \times 0.20 + 2 \times 0.35 = 2.45$$

ALGORITHM 2 Huffman Coding.

procedure *Huffman*(C: symbols a_i with frequencies w_i, $i = 1, \ldots, n$)
$F :=$ forest of n rooted trees, each consisting of the single vertex a_i and assigned weight w_i
while F is not a tree
  Replace the rooted trees T and T' of least weights from F with $w(T) \geq w(T')$ with a tree having a new root that has T as its left subtree and T' as its right subtree. Label the new edge to T with 0 and the new edge to T' with 1.
  Assign $w(T) + w(T')$ as the weight of the new tree.
{the Huffman coding for the symbol a_i is the concatenation of the labels of the edges in the unique path from the root to the vertex a_i}

Exercise

1. Use Huffman Coding Algorithm to encode these symbols with given frequencies

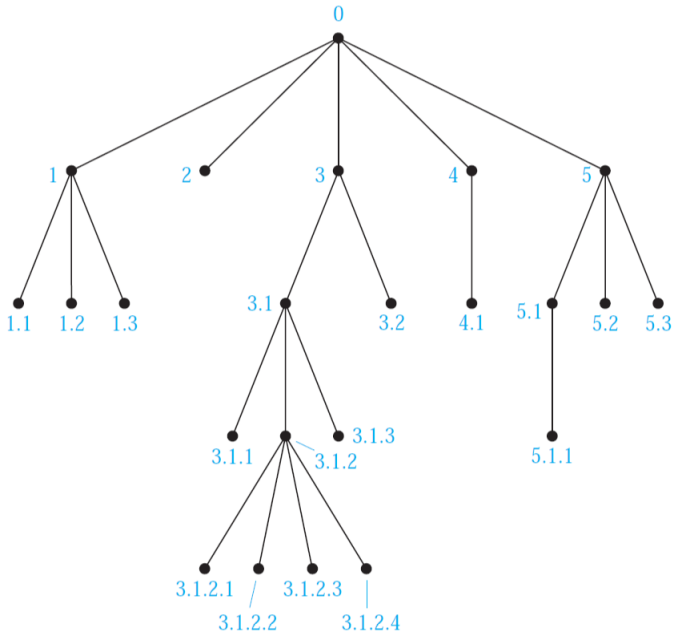
$$a : 0.2,\ b : 0.1,\ c : 0.15,\ d : 0.25,\ e : 0.3$$

2. Use Huffman Coding Algorithm to encode the word "google". What is the average number of bits required to encode a character?

Universal Address Systems is a way of labeling all the vertices by recursively:

• Label the root with the integer 0. Then label its k children (at level 1) from left to right with $1, 2, ..., k$.

• For each vertex v at level n with label A, label its k_v children, as they are drawn from left to right, with $A.1, A.2, ..., A.k_V$.


0
1
2
3
4
5
1.1
1.2
1.3
3.1
3.2
4.1
5.1
5.2
5.3
3.1.1
3.1.2
3.1.3
5.1.1
3.1.2.1
3.1.2.2
3.1.2.3
3.1.2.4

Traversal Algorithms

- Traversal Algorithms are procedure for systematically visiting every vertex of an ordered rooted tree.
  - preorder traversal
  - postorder traversal
  - inorder traversal

Preorder traversal: visit root, visit subtrees left to right

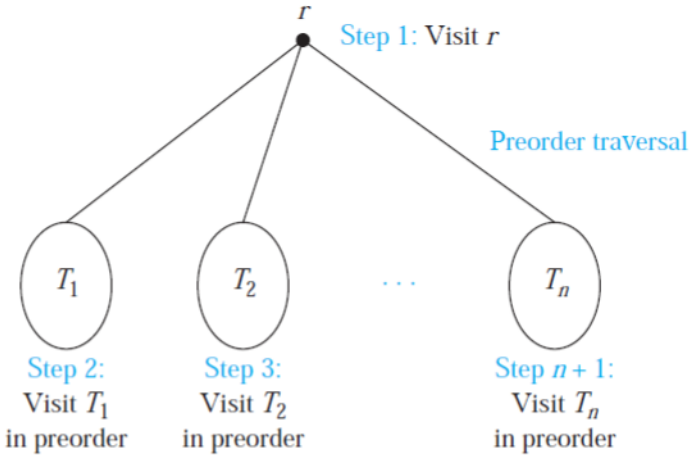
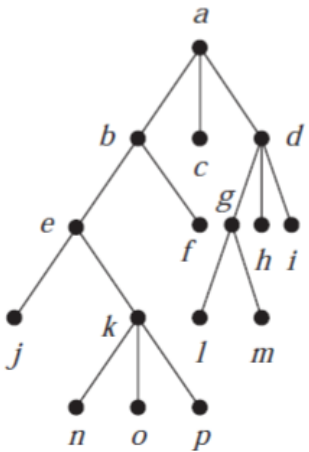
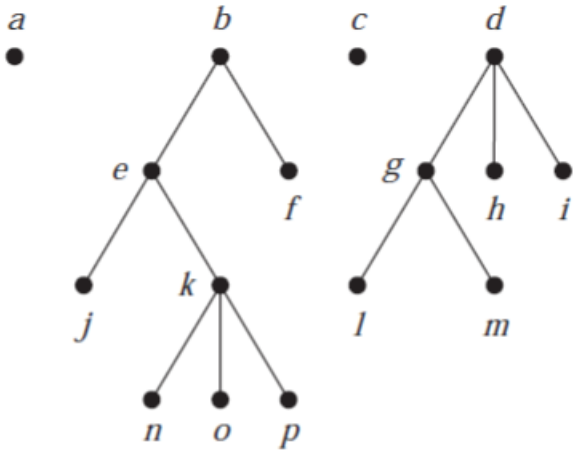
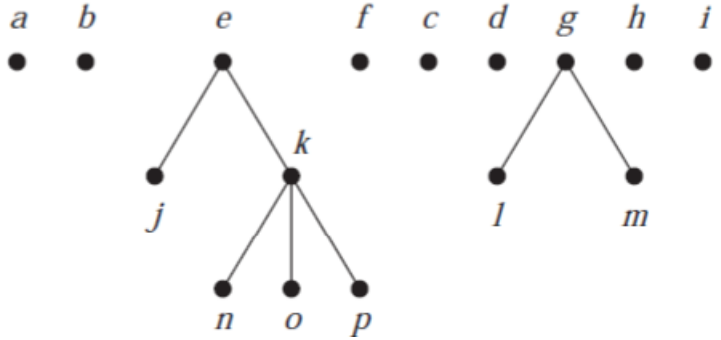


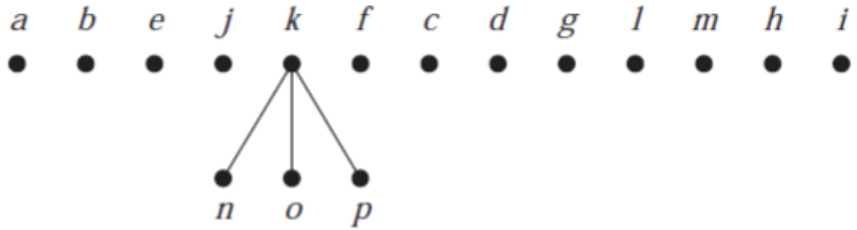
FIGURE 2 **Preorder Traversal.**

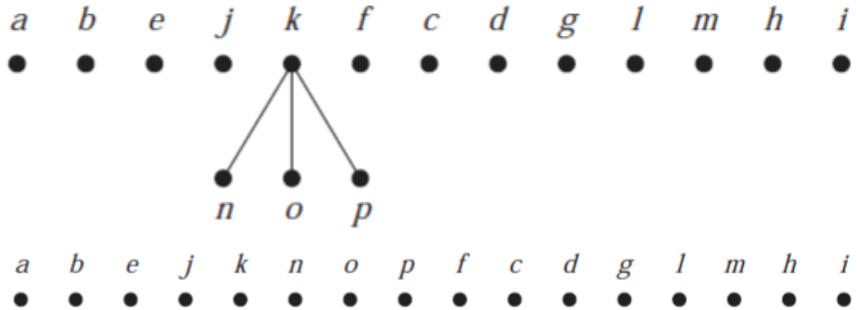
Example 1 Preorder traversal (N-L-R)



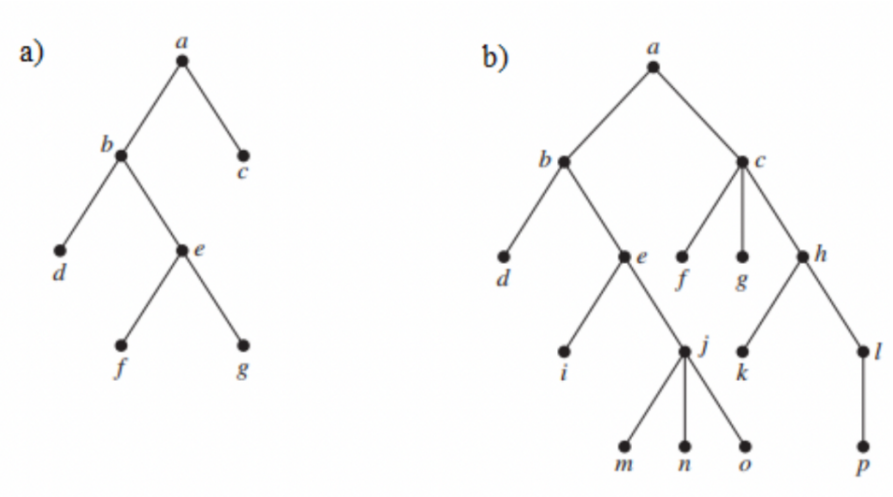

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p


a
b
e
f
c
d
g
h
i
k
j
l
m
n
o
p


a b e j k f c d g l m h i
n o p


a b e j k f c d g l m h i
n o p
a b e j k n o p f c d g l m h i

Exercise: Preorder traversal



Inorder Traversal: Visit subtree lefts, vitsit root, visit subtree right

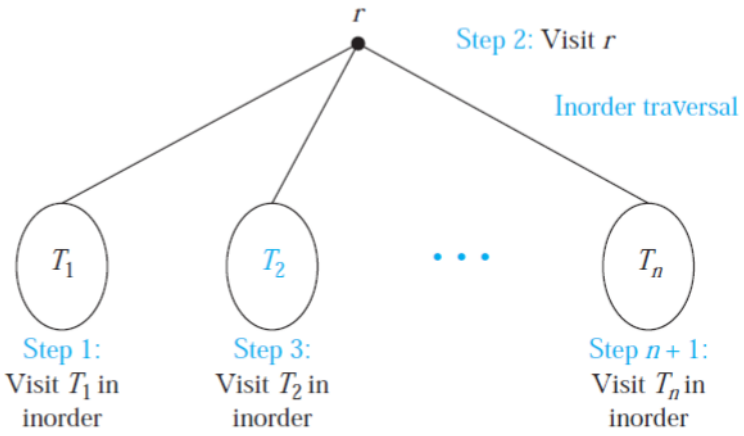
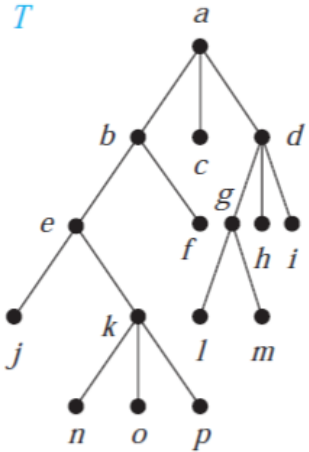
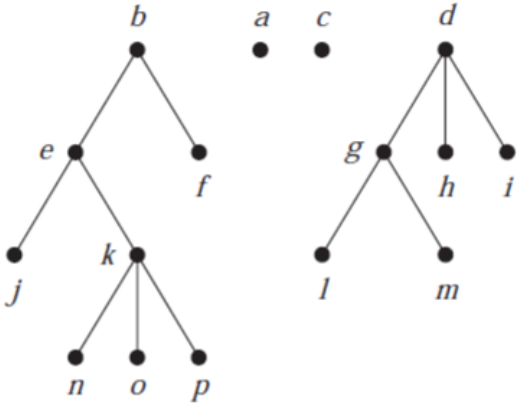
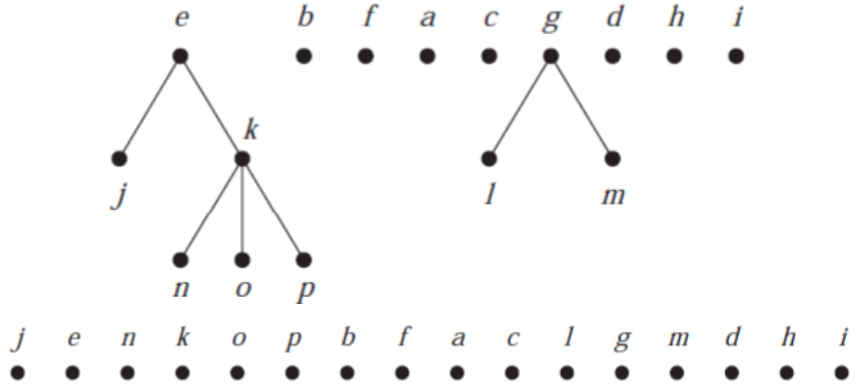


FIGURE 5 Inorder Traversal.

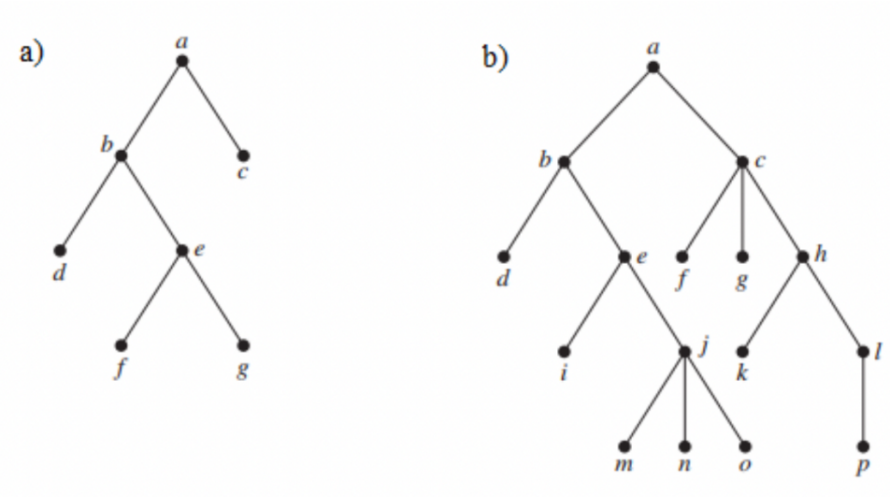
Example 1 Inorder Traversal (L-N-R)




b
a
c
d
e
f
g
h
i
k
j
l
m
n
o
p


e
b
f
a
c
g
d
h
i
k
j
l
m
n
o
p
j
e
n
k
o
p
b
f
a
c
l
g
m
d
h
i

Exercise: Inorder Traversal



Postorder traversal: Visit subtrees left to right, visit root

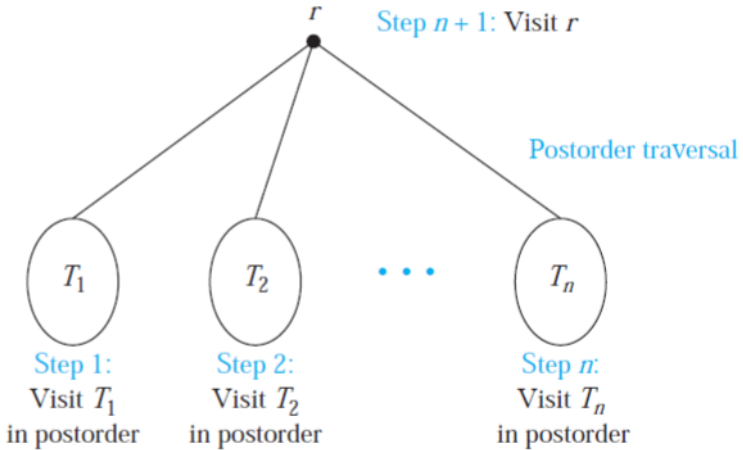
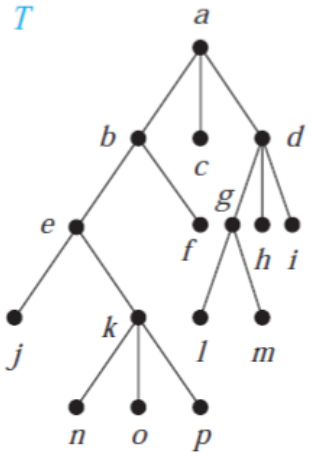
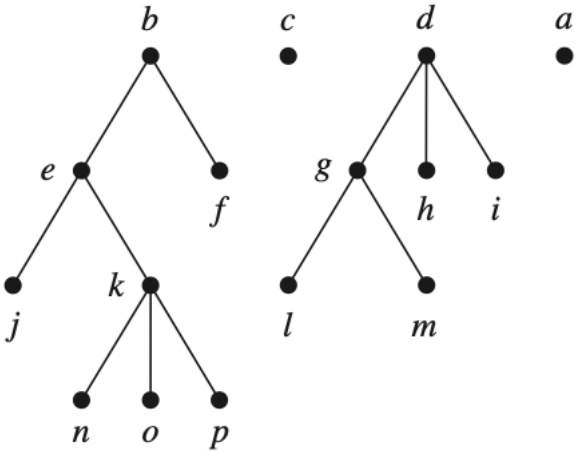
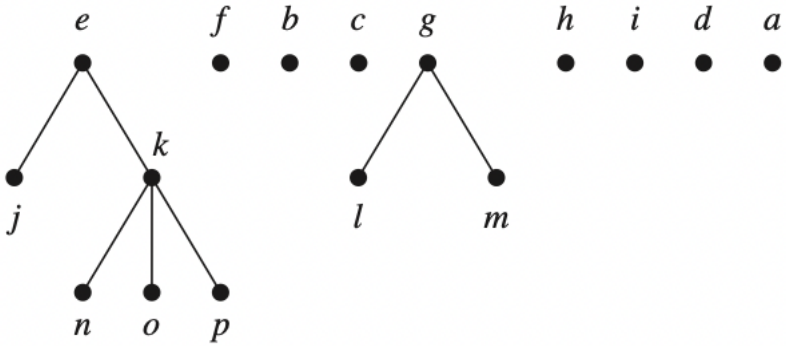


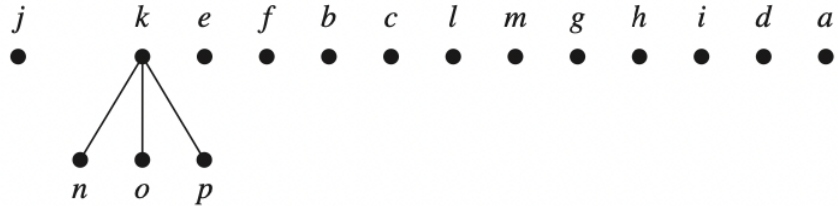
FIGURE 7 **Postorder Traversal.**

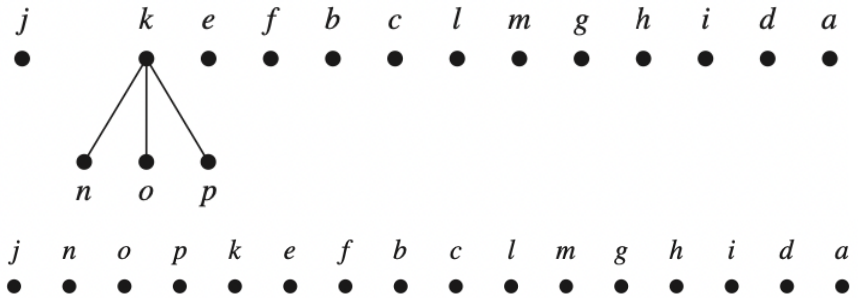
Example 1 postorder traversal (L - R - N)



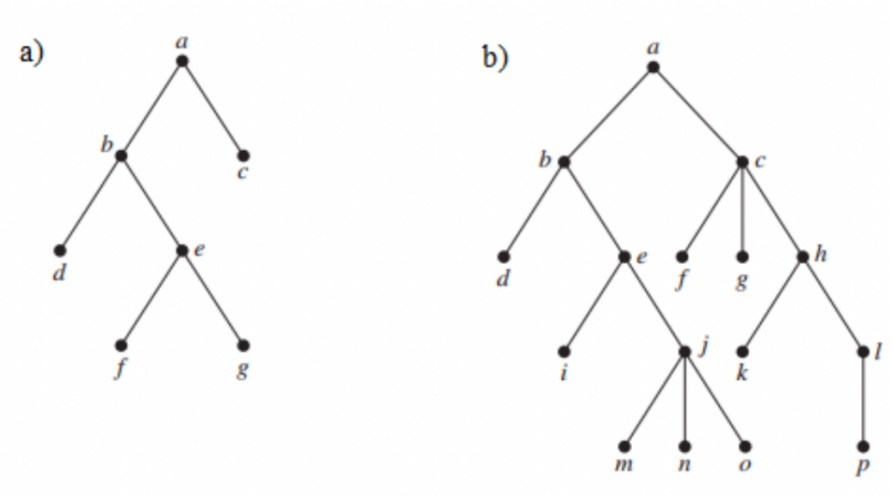

b
c
d
a
e
g
f
h
i
k
j
l
m
n
o
p


e
f
b
c
g
h
i
d
a
k
j
l
m
n
o
p


j
k
e
f
b
c
l
m
g
h
i
d
a
n
o
p


j k e f b c l m g h i d a
n o p
j n o p k e f b c l m g h i d a

Exercise: Postorder Traversal



• Ordered rooted trees can be used to represent complicated expressions, such as compound propositions, combinations of sets, and arithmetic expressions.

Example 1 What is the ordered rooted tree that represents the expression $((x + y) \uparrow 2) + ((x - 4)/3)$??

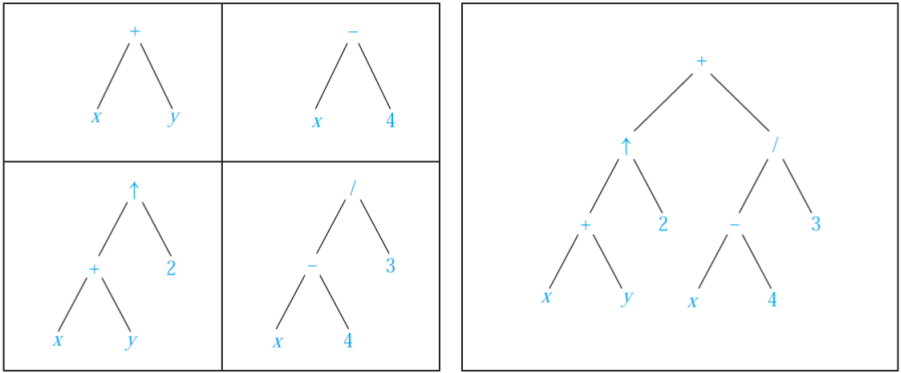
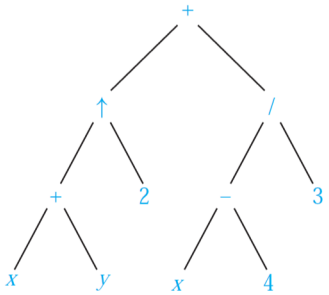


FIGURE 10 **A Binary Tree Representing $((x + y) \uparrow 2) + ((x - 4)/3)$.**

Example 2 What is the prefix form for $((x + y) \uparrow 2) + ((x - 4)/3)$?

Solution

Step 1: Establishing binary rooted tree



Step 2: traversing the tree in preorder

$+ \quad \uparrow \quad + \quad x \quad y \quad 2 \quad / \quad - \quad x \quad 4 \quad 3$

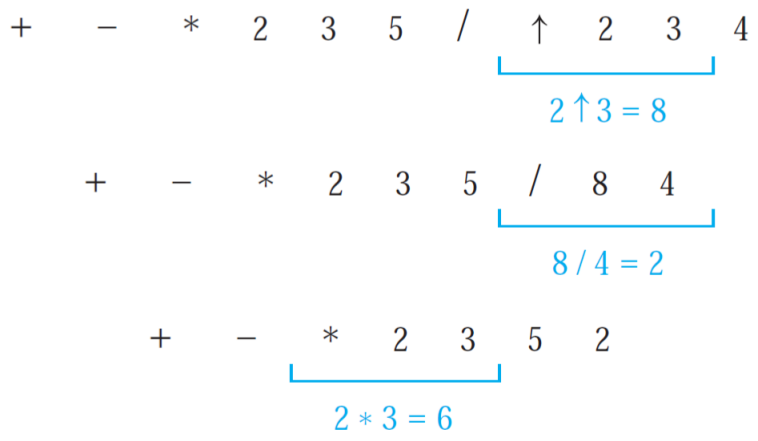
Example 3

What is the value of the prefix expression

$+ - \times \ 2 \ 3 \ 5 \ / \ \uparrow \ 2 \ 3 \ 4$?

Solution

The steps used to evaluate this expression by working right to left.


+ − * 2 3 5 / ↑ 2 3 4
2 ↑ 3 = 8
+ − * 2 3 5 / 8 4
8 / 4 = 2
+ − * 2 3 5 2
2 * 3 = 6

$$+ \quad - \quad \underbrace{* \quad 2 \quad 3} \quad 5 \quad 2$$

$$2 * 3 = 6$$

$$+ \quad \underbrace{- \quad 6 \quad 5} \quad 2$$

$$6 - 5 = 1$$

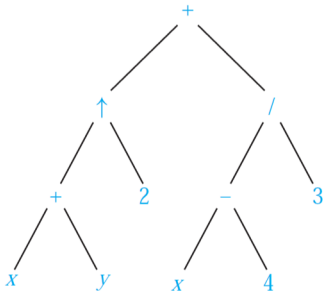
$$\underbrace{+ \quad 1 \quad 2}$$

$$1 + 2 = 3$$

Example 4 What is the postfix form of the expression $((x + y) \uparrow 2) + ((x - 4)/3)$?

Solution

Step 1: Establishing binary rooted tree



Step 2: traversing the tree in postorder: $x\ y\ +\ 2\ \uparrow\ x\ 4\ -\ 3\ /\ +$

Example 5

What is the value of the postfix expression $7\ 2\ 3\ *\ -\ 4\ \uparrow\ 9\ 3\ /\ +$?

Solution

Starting evaluating at the left and carrying out operations when two operands are followed by an operator.

$$7\ \underbrace{2\ 3\ *}\ -\ 4\ \uparrow\ 9\ 3\ /\ +$$

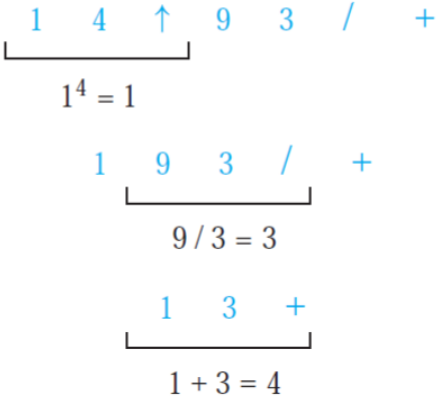
$$2 * 3 = 6$$

$$\underbrace{7\ 6\ -}\ 4\ \uparrow\ 9\ 3\ /\ +$$

$$7 - 6 = 1$$

$$\underbrace{1\ 4\ \uparrow}\ 9\ 3\ /\ +$$

$$1^4 = 1$$


1 4 ↑ 9 3 / +
$1^4 = 1$
1 9 3 / +
9 / 3 = 3
1 3 +
1 + 3 = 4

Exercise

1. What is the ordered rooted tree that represents the expression $((x + y) \uparrow 2) + ((x - 4)/3)$?

2. What is the prefix form for $((x + y) \uparrow 2) + ((x - 4)/3)$?

3. What is the value of the prefix expression
$+ \; - \; * \; 2 \; 3 \; 5 \; / \; \uparrow \; 2 \; 3 \; 4$?

4. What is the postfix form of the expression
$((x + y) \uparrow 2) + ((x - 4)/3)$?

Exercise

5. What is the value of the postfix expression
$7\ 2\ 3\ *\ -\ 4\ \uparrow\ 9\ 3\ /\ +$?

6. a) Represent the expression $((x+2)\uparrow 3)*(y-(3+x))-5$ using a binary tree. Write this expression in

b) prefix notation

c) postfix notation

d) infix notation

Exercise

7. a) Represent the expressions $(x + x * y) + (x/y)$ and $x + ((x * y + x)/y))$ using binary trees.

Write these expressions in
b) prefix notation

c) postfix notation

d) infix notation

Exercise

Find the postfix notation for the expression [x + ((y – 5)*4)] ↑ 3

A. x + y – 5 * 4 ↑ 3

B. x 5 y – 4 * + ↑ 3

C. x y 5 – 4 * + 3 ↑

D. None of these other choices is correct

Exercise

What is the value of this prefix expressions?

$$+\ 4\ /\ *\ 7\ \ 3\ +\ 5\ -\ 18\ \uparrow\ 2\ \ 4$$

A.5 B.6 C.7 D.8

Exercise

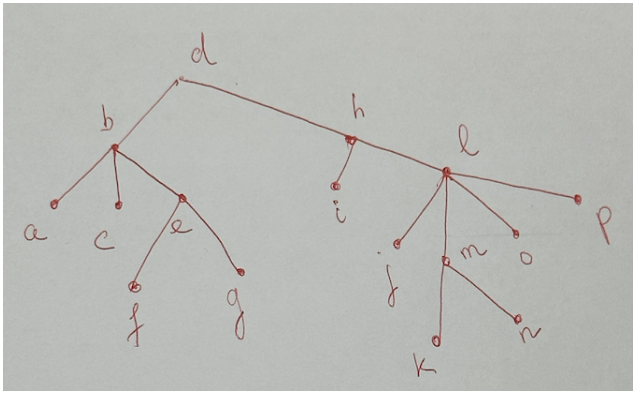
What is the value of this postfix expressions?

$$4\ \ 3\ *\ 2\ \uparrow\ 7\ \ 5\ -\ 9\ \ 3\ /\ *\ /$$

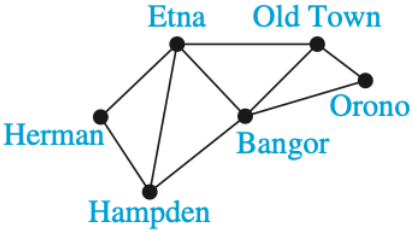
A.24 B.25 C.26 D.27

Exercise

If inorder traversal is used, what is the position of the vertex "d" in the list?



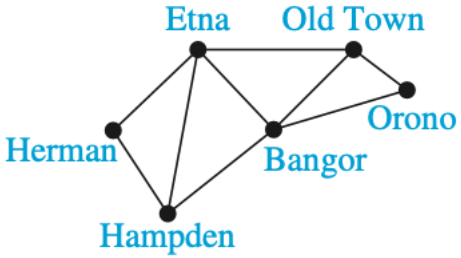
A.3 B.4 C.5 D.6 E.7



The only way the roads can be kept open in the winter is by frequently plowing them

The highway department wants to plow the fewest roads so that there will always be cleared roads connecting any two towns.How can this be done?

At least five roads must be plowed to ensure that there is a path between any two towns. Figure 1(b) shows one such set of roads



Là 1 cây đi qua hết tất cả các đỉnh của đồ thị gốc.

Definition

Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.

Cho đồ thị, tìm spanning tree!

Example 1 Find an spanning tree of the simple graph G below?

liên thông
không có cạnh bội khép kín } cây

△ ; □ : khép kín
→ bỏ bớt 1 cạnh

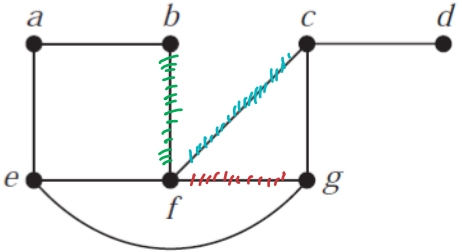
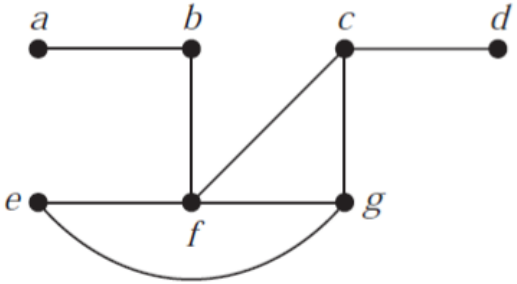
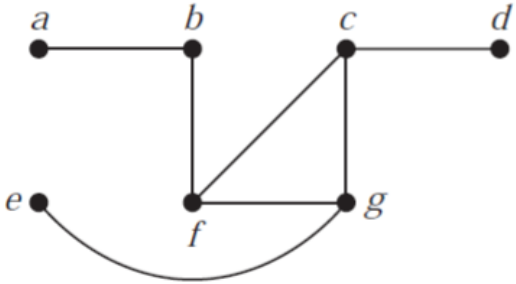


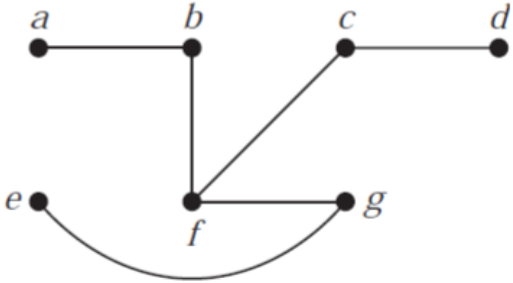
FIGURE 2 The Simple Graph G.

Solution



Edge removed: $\{a, e\}$


a
b
c
d
e
f
g
{e, f}


a
b
c
d
e
f
g
{c, g}

Theorem

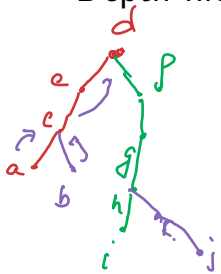
A simple graph is connected if and only if it has a spanning tree.

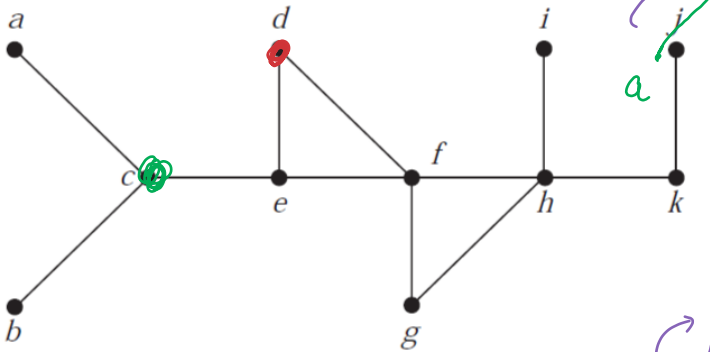
Depth-First Search

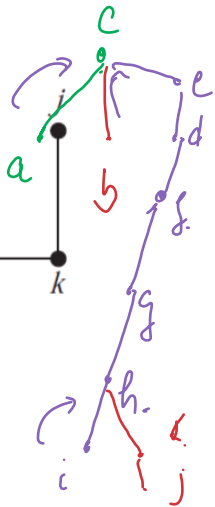
Depth-first Search là 1 thuật toán xây dựng "spanning tree" từ 1 connected simple graph bằng cách

- Chọn 1 đỉnh nào đó làm gốc
- Đi một con đường bằng cách bắt đầu tại đỉnh này và thêm vào cạnh mới (cạnh mới là cạnh có 1 đỉnh cũ và 1 đỉnh mới)
- Đi được cứ đi, không đi được thì lùi lại rồi đi tiếp con đường mới (nếu có thể), lùi lại mà vẫn không đi được thì lùi lại nữa rồi đi (nếu có thể)
- Tiếp tục quá trình này cho đến khi ta đi hết tất cả các đỉnh

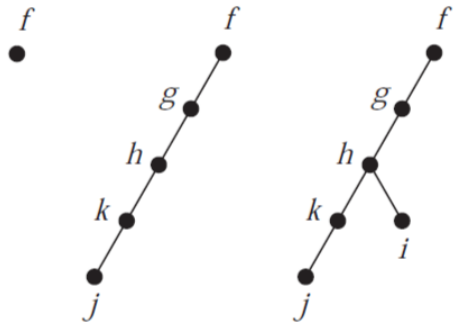
Example 1 Find a spanning tree of the graph below by using Depth-first Search algorithm?

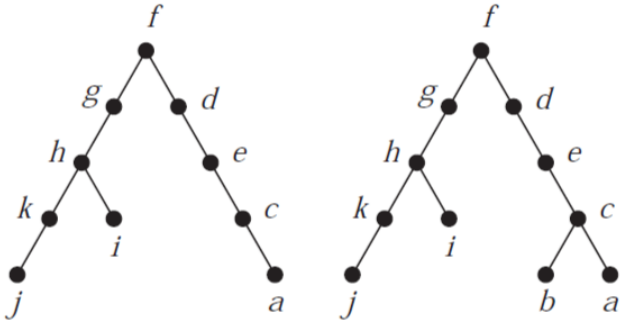


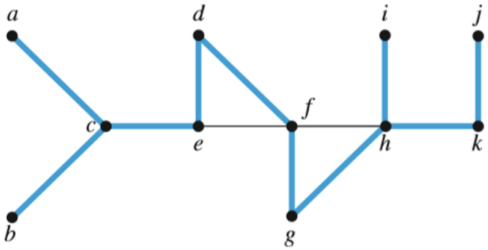




Solution We arbitrarily start with the vertex f.




f
g
d
h
e
k
c
i
j
a
f
g
d
h
e
k
c
i
j
b
a



The edges selected by depth-first search of a graph are called tree edges.
All other edges are called back edges.

FIGURE 8 **The Tree Edges and Back Edges of the Depth-First Search in Example 4.**

Depth-first Search Algorithm

procedure *DFS*(G: connected graph with vertices $v_1, v_2, \ldots, v_n$)
$T :=$ tree consisting only of the vertex v_1
visit(v_1)

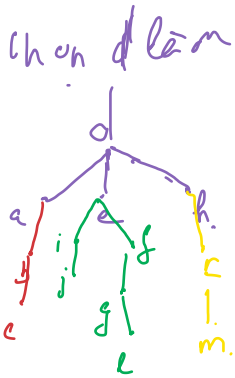
procedure *visit*(v: vertex of G)
for each vertex w adjacent to v and not yet in T
    add vertex w and edge $\{v, w\}$ to T
    visit(w)

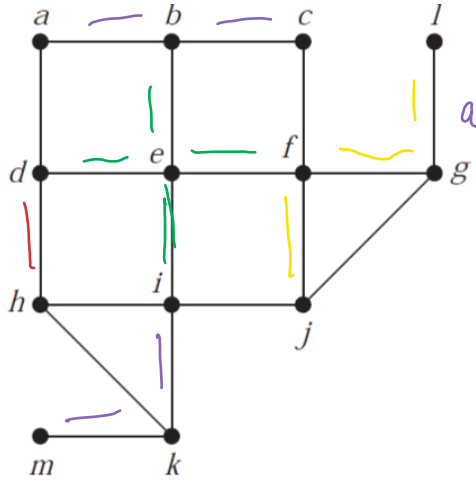
Breath-First Search

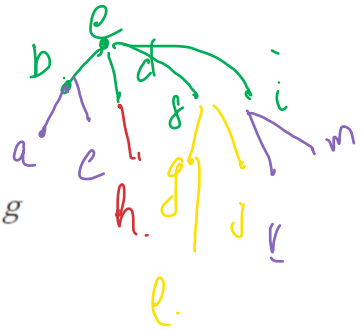
Breath-first Search là 1 thuật toán xây dựng "spanning tree" từ 1 connected simple graph bằng cách

- Chọn 1 đỉnh nào đó làm gốc
- Thêm những cạnh kề với đỉnh này, ta được những đỉnh có mức 1
- Tại những đỉnh có mức 1, thêm những cạnh kề với đỉnh đó, miễn là không tạo ra một đường khép kín. Ta được những đỉnh có mức là 2.
- Tiếp tục quá trình này cho đến khi ta đi hết tất cả các đỉnh

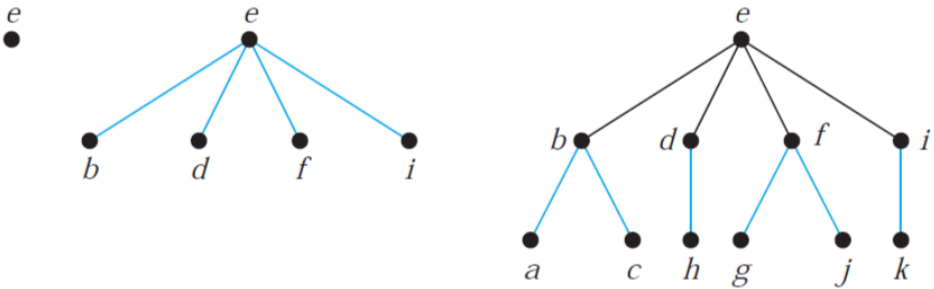
Example 1 Use Breadth-First search to find a spanning tree for the graph below?

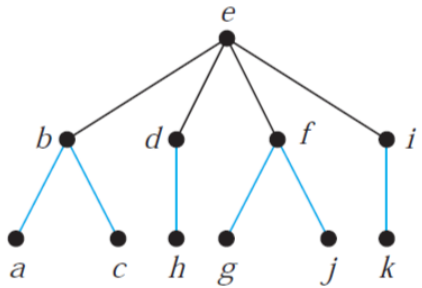


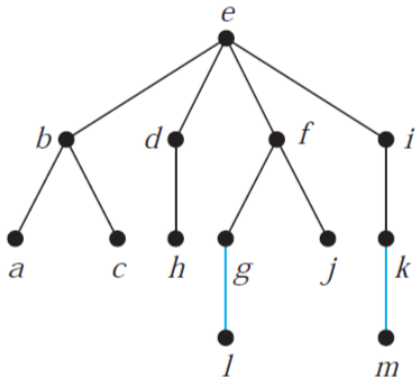




Solution

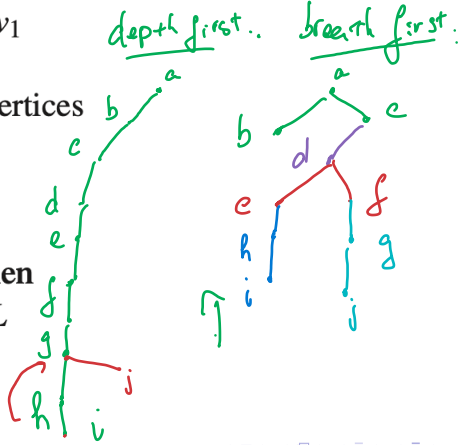



e
b
d
f
i
a
c
h
g
j
k


e
b
d
f
i
a
c
h
g
j
k
l
m

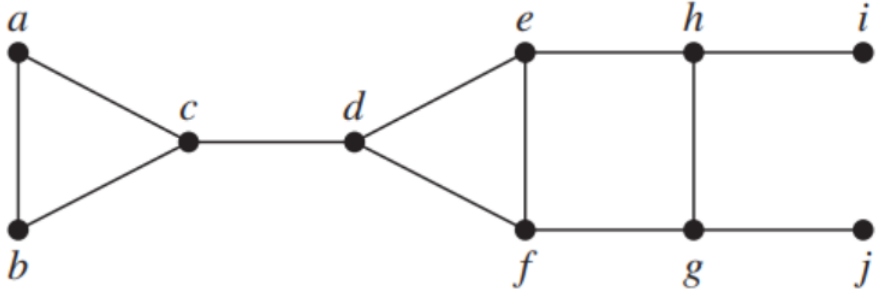
Breath-First Algorithm

procedure *BFS* (G: connected graph with vertices $v_1, v_2, \ldots, v_n$)
$T :=$ tree consisting only of vertex v_1
$L :=$ empty list
put v_1 in the list L of unprocessed vertices
while L is not empty
    remove the first vertex, v, from L
    for each neighbor w of v
        if w is not in L and not in T **then**
            add w to the end of the list L
            add w and edge $\{v, w\}$ to T



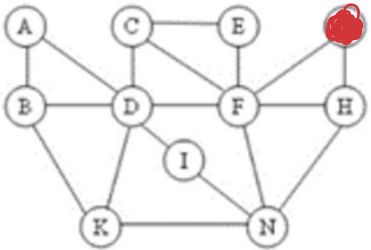
Exercise

Use depth-first search and breath-first search to find a spanning tree of the graph below. Choose a as the root of this spanning tree and assume that the vertices are ordered alphabetically.



Exercise

Use depth-first search (DFS) to produce a spanning tree for the given graph. Choose G as the root of this spanning tree and assume that the vertices are ordered alphabetically.



A. G, F, I, N, K, B, A, C, D, E, H

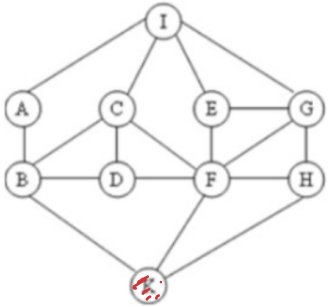
B. G, F, C, D, A, B, K, N, H, I, E

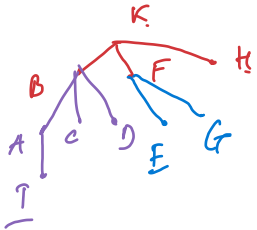
C. G, H, N, K, B, A, C, D, E, I, F

D. G, A, B, C, D, E, F, N, K, H, I

Exercise

Use breadth-first search (BFS) to produce a spanning tree for the given graph. Choose K as the root of this spanning tree and assume that the vertices are ordered alphabetically.





A. K, B, F, H, A, C, D, E, G, I

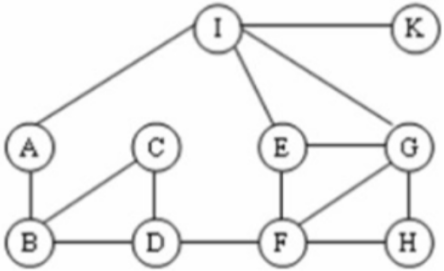
B. K, B, D, F, H, A, C, E, G, I

C. K, B, A, C, D, E, F, G, H, I

D. K, E, F, G, H, A, B, C, D, I

Exercise

Use breadth-first search (BFS) to produce a spanning tree for the given graph. Choose K as the root of this spanning tree and assume that the vertices are ordered alphabetically.



A. K, A, B, C, D, E, F, G, H, I

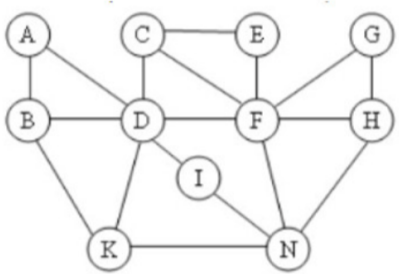
B. K, A, C, E, G, B, D, F, H, I

C. K, I, E, G, F, H, A, B, C, D

D. K, I, A, E, G, B, F, H, C, D

Exercise

Use depth-first search (DFS) to produce a spanning tree for the given graph. Choose A as the root of this spanning tree and assume that the vertices are ordered alphabetically.



A. A, B, D, K, I, N, C, E, G, H, F

B. A, C, E, F, D, B, I, N, K, H, B

C. A, B, D, C, E, F, G, H, N, I, K

D. A, K, N, D, F, H, G, E, C, I, B

Minimum Spanning Tree

Definition

A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

Example 1 Find a minimum spanning tree.

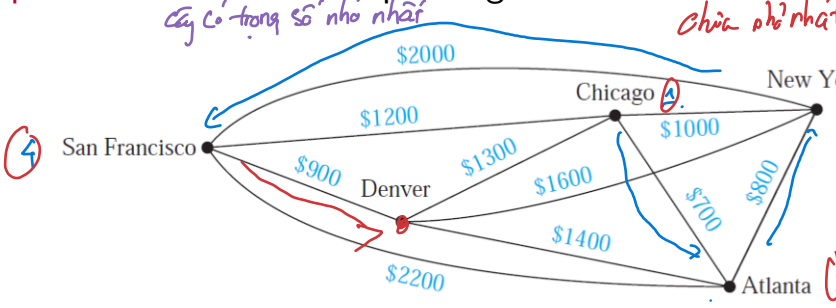
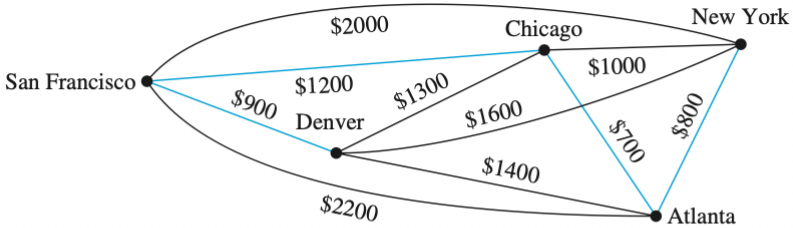


FIGURE 1 **A Weighted Graph Showing Monthly Lease Costs for Lines in a Computer Network.**

Prim's Algorithm

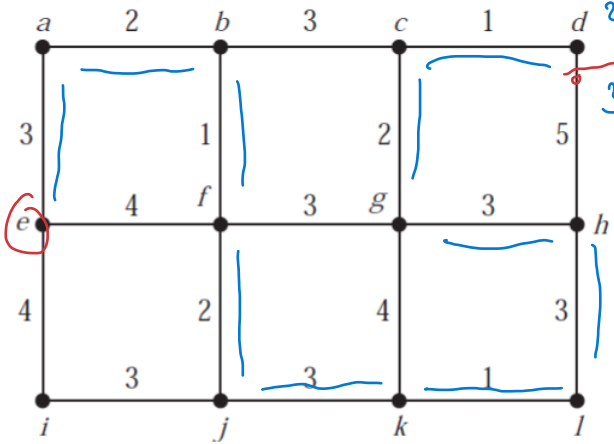
Prim's Algorithm là một thuật toán tìm cây "minimum spanning tree" từ đồ thị liên thông bằng cách

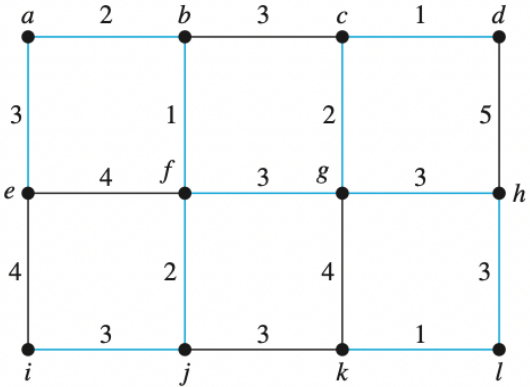
- Lập cây với cạnh nhỏ nhất
- Chọn cạnh có trọng số nhỏ nhất kề với các đỉnh của cây, miễn là không tạo ra một đường khép kín.
- Dừng lại khi có $n - 1$ cạnh đã thêm vào. (n: số đỉnh)



Choice	Edge	Cost
1	{Chicago, Atlanta}	$ 700
2	{Atlanta, New York}	$ 800
3	{Chicago, San Francisco}	$1200
4	{San Francisco, Denver}	$ 900
	Total:	$3600

Example 2





(a)

Choice	Edge	Weight
1	$\{b, f\}$	1
2	$\{a, b\}$	2
3	$\{f, j\}$	2
4	$\{a, e\}$	3
5	$\{i, j\}$	3
6	$\{f, g\}$	3
7	$\{c, g\}$	2
8	$\{c, d\}$	1
9	$\{g, h\}$	3
10	$\{h, l\}$	3
11	$\{k, l\}$	1
	Total:	24

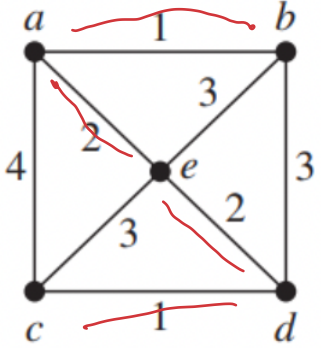
(b)

ALGORITHM 1 Prim's Algorithm.

procedure *Prim*(G: weighted connected undirected graph with n vertices)
$T :=$ a minimum-weight edge
for $i := 1$ **to** $n - 2$
  $e :=$ an edge of minimum weight incident to a vertex in T and not forming a simple circuit in T if added to T
  $T := T$ with e added
return T {T is a minimum spanning tree of G}

Exercise

Use Prim's algorithm to find a minimum spanning tree for the given weighted graph.



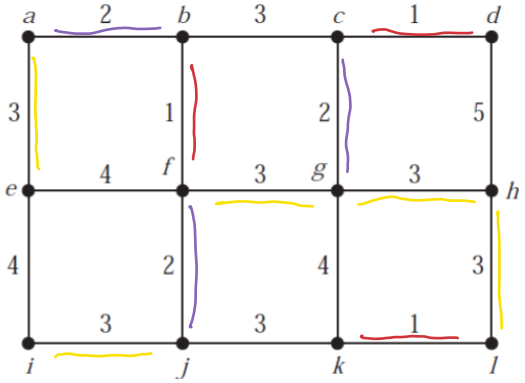
thêm cạnh nhỏ nhất, ko nhất thiết kề, ko khép kín

Kruskal's Algorithm

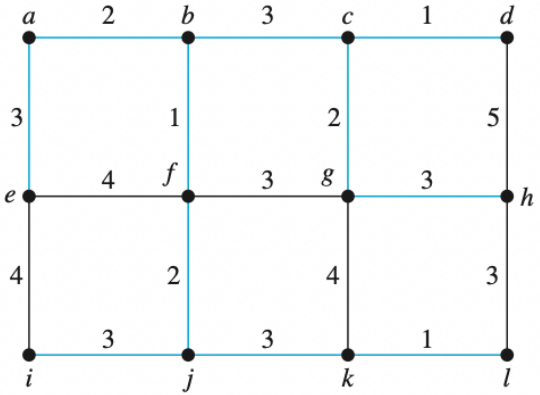
Kruskal's Algorithm là một thuật toán tìm cây "minimum spanning tree" từ đồ thị liên thông bằng cách

- Lập cây với cạnh nhỏ nhất
- Chọn cạnh có trọng số nhỏ nhất (không nhất thiết kề với các đỉnh của cây), miễn là không tạo ra một đường khép kín với các cạnh đã thêm.
- Dừng lại khi có $n-1$ cạnh đã thêm vào. (n: số đỉnh)

Example 2



Cạnh nhỏ nhất: 1.
Cạnh nhỏ nhì: 2.
Cạnh nhỏ ba: 3.



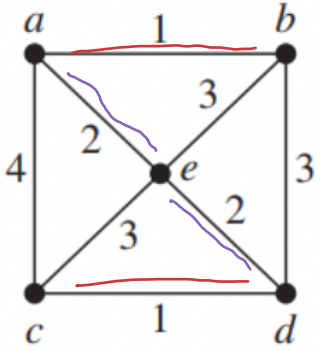
Choice	Edge	Weight
1	$\{c, d\}$	1
2	$\{k, l\}$	1
3	$\{b, f\}$	1
4	$\{c, g\}$	2
5	$\{a, b\}$	2
6	$\{f, j\}$	2
7	$\{b, c\}$	3
8	$\{j, k\}$	3
9	$\{g, h\}$	3
10	$\{i, j\}$	3
11	$\{a, e\}$	3
	Total:	24

ALGORITHM 2 Kruskal's Algorithm.

procedure *Kruskal*(G: weighted connected undirected graph with n vertices)
$T :=$ empty graph
for $i := 1$ **to** $n - 1$
  $e :=$ any edge in G with smallest weight that does not form a simple circuit when added to T
  $T := T$ with e added
return T {T is a minimum spanning tree of G}

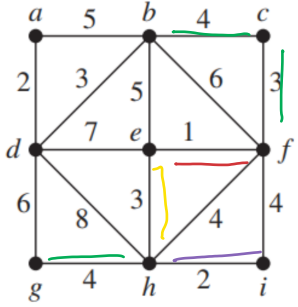
Exercise

Use Krushkal's algorithm to find a minimum spanning tree for the given weighted graph.

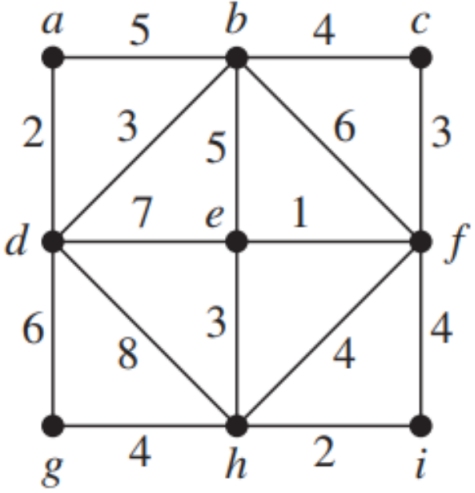


IF we use the Prim algorithm to find a minimum tree, what are the fifth and sixth edges (in correct order) added?

We always assume that an edge is written with two endpoints following the alphabet order (which means we write edge "ad" instead of "da"). We suppose that at each stage if we have many choices, we will choose in reverse direction order. For example, if we have to choose between edges 'bc', 'de', 'az', we will choose 'de' first.



A. Gh fi B. bc bd C. fi bd D. gh bc E. fi bc F. gh bd

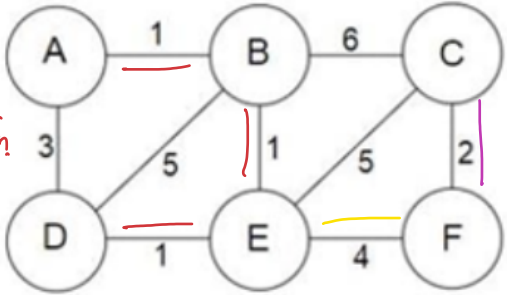


A. gh,fi B. bc,bd C. fi,bd D. gh,bc E. fi,bc F. gh,bd

Exercise

If using the Kruskal's algorithm to find a minimum spanning tree T from the graph below, which edge is added to T in the last step?

Kruskal:
thêm cạnh nhỏ nhất
rồi cân liên thông không khép kín

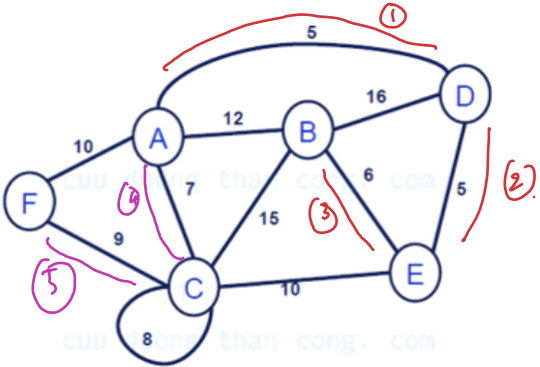


→ EF: cạnh cuối cùng

A. AD B. BD C. EC D. EF

Exercise

If using the Prim's algorithm to find a minimum spanning tree T from the graph below , which edge is added to T in the last step?



A. CE B. AF C. CF D. AF