TOÁN RỜI RẠC VÀ THUẬT TOÁN

(DISCRETE MATHEMATIC AND ALGORITHMS)

Bài 7

Cây và ứng dụng

(Tree and its applications)

Nguyễn Thị Hồng Minh

minhnth@hus.edu.vn

Nội dung

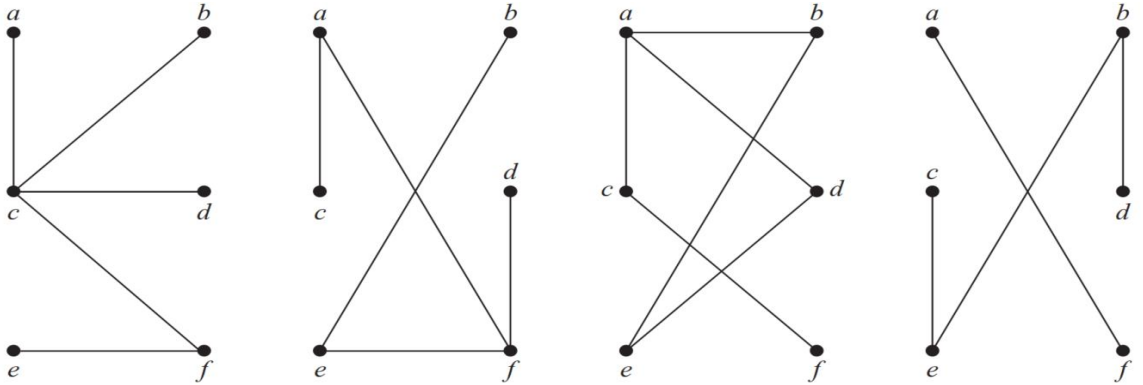
1. **Cây và một số khái niệm liên quan**
2. **Cây nhị phân và ứng dụng**
3. **Cây bao trùm tối thiểu của đồ thị và ứng dụng**
4. **Bài tập thực hành**

Chú ý: Hầu hết các hình vẽ trong các bài giảng được sưu tầm từ internet, sách Rosen và được trình bày theo quan điểm của giảng viên.

Cây và một số khái niệm liên quan

❖ Khái niệm cây (tree)

- Cây là một đồ thị vô hướng liên thông và không có bất kì chu trình nào.
  - Không có chu trình: không có đường đi khép kín qua các đỉnh
  - Chỉ có 1 đường đi duy nhất giữa hai đỉnh.
- Ví dụ:



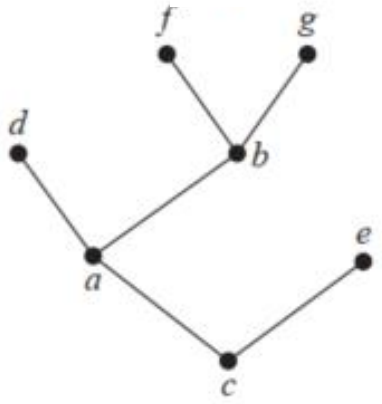
G_1 G_2 G_3 G_4

Cây và một số khái niệm liên quan

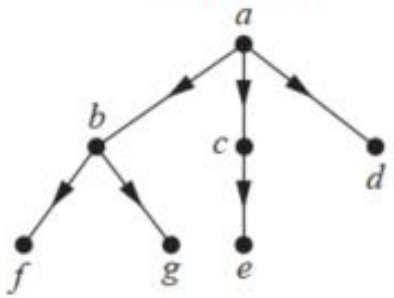
❖ Khái niệm cây (tree)

- Cây hướng gốc (Rooted Trees): Cây có 1 đỉnh đặc biệt được gọi là gốc và mọi cạnh tới các đỉnh khác đều hướng ra từ gốc.
- Ví dụ:

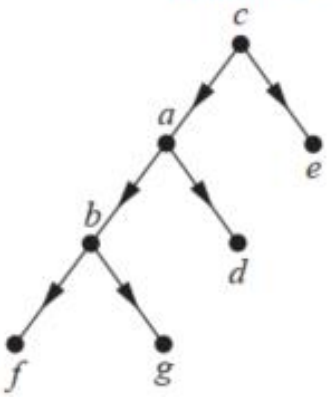
T



With root a



With root c

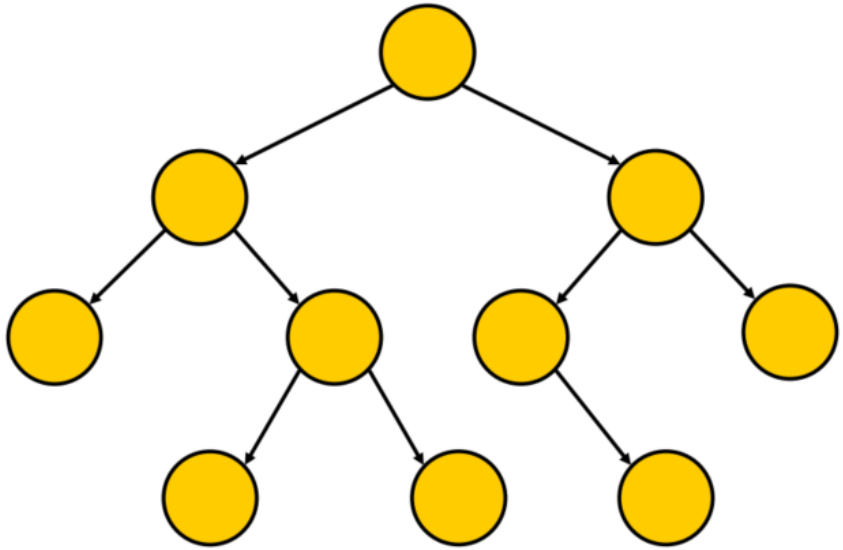


CÂY NHỊ PHÂN VÀ ỨNG DỤNG

BINARY TREE AND APPLICATIONS

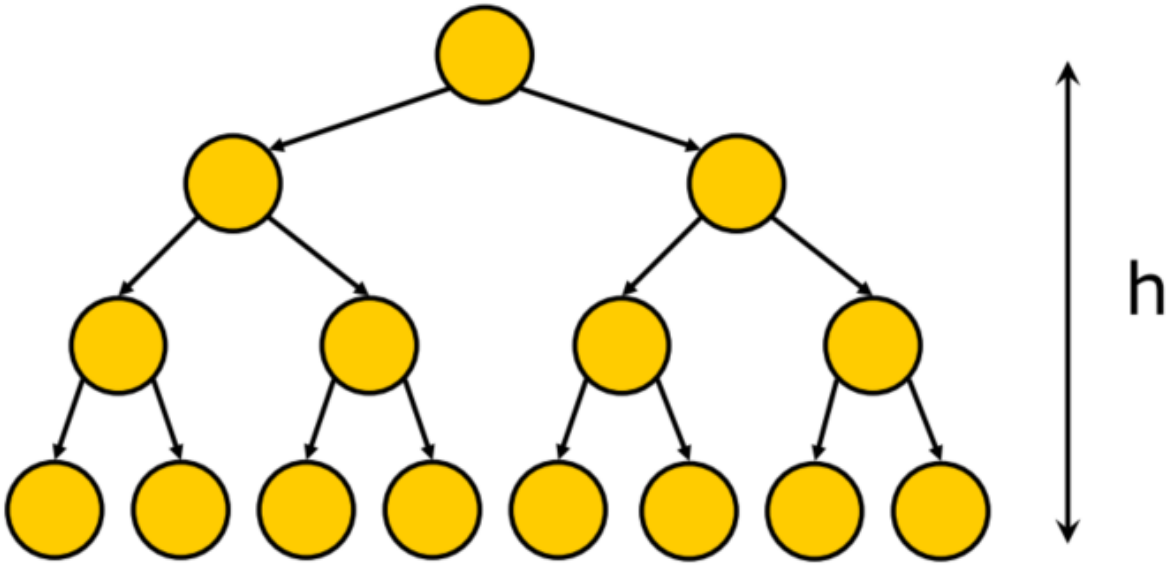
Cây nhị phân: Định nghĩa

- Binary Tree = 2-ary tree (cây nhị phân)
  - Cây có 1 nút gốc,
  - Mỗi nút có nhiều nhất 2 con
  - Hai con: con trái (left child), con phải (right child)



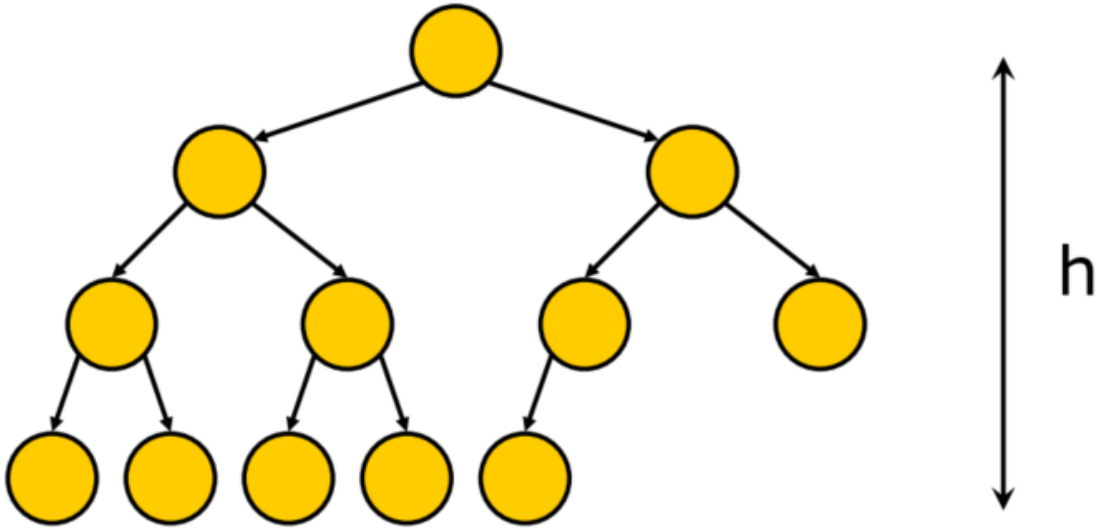
Cây nhị phân: Định nghĩa

- Full Binary Tree – Cây nhị phân đầy đủ
  - Tất cả các nút trong (level < h) có đủ hai con (h là chiều cao của cây)



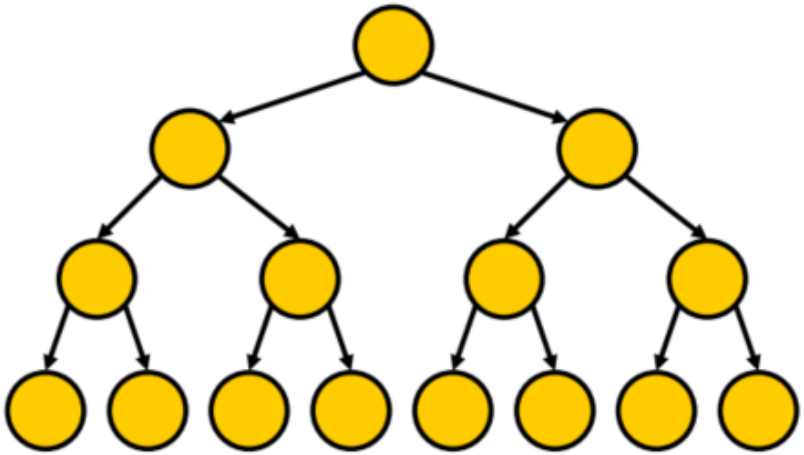
Cây nhị phân: Định nghĩa

- Complete Binary Tree – Cây nhị phân hoàn chỉnh
  - Các nút điền đủ tới mức *h*-1
  - Mức *h* điền từ trái sang phải



Cây nhị phân đầy đủ

- Tính chất cây nhị phân đầy đủ
  - Số nút trên cây đầu đủ chiều cao h là $N = 2^h - 1$.
  - Chiều cao cây nhị phân đầy đủ N nút là $O(\log N)$



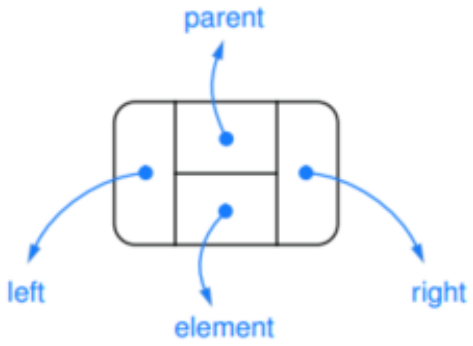
- $h = 4$
- $N = 15$

Cài đặt cây nhị phân

- Cài đặt cây nhị phân
  - Sử dụng cấu trúc liên kết
  - Sử dụng mảng

Cài đặt cây nhị phân

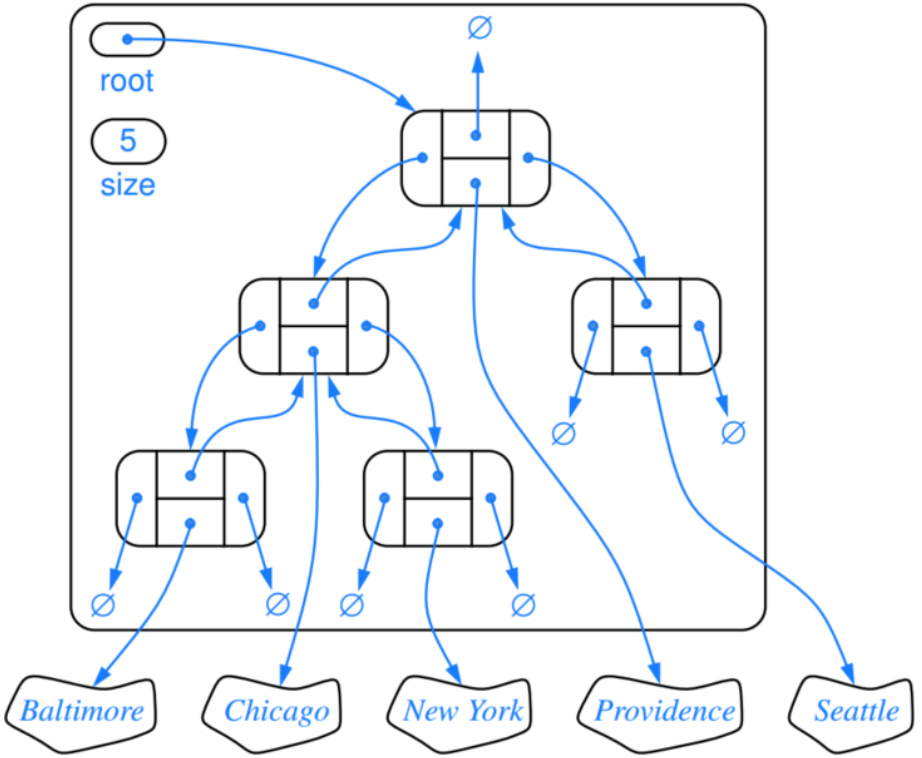
- Cài đặt cây nhị phân bằng cấu trúc liên kết



One node with:

parent, left, right: references

element: data



Binary Tree with five nodes

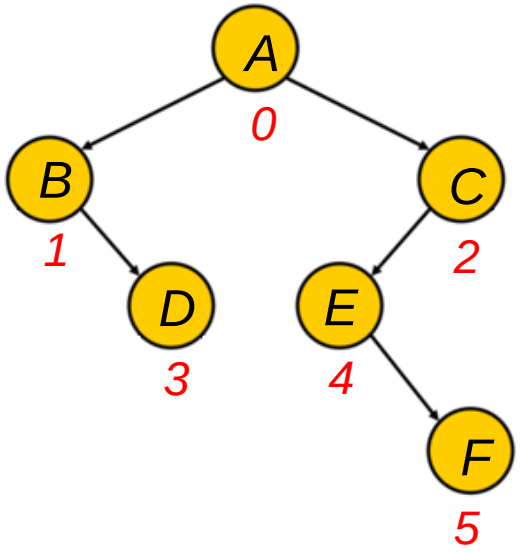
Source: [M.Goodrich, p323]

Cài đặt cây nhị phân

- Cài đặt cây nhị phân bằng mảng

Nodes T[]

	0	1	2	3	4	5
Data	A	B	C	D	E	F
Parent	-1	0	0	1	2	4
Left	1	-1	4	-1	-1	-1
Right	2	3	-1	-1	5	-1



- Ưu điểm: Dễ dàng truy cập các nút
- Nhược điểm: Tốn bộ nhớ

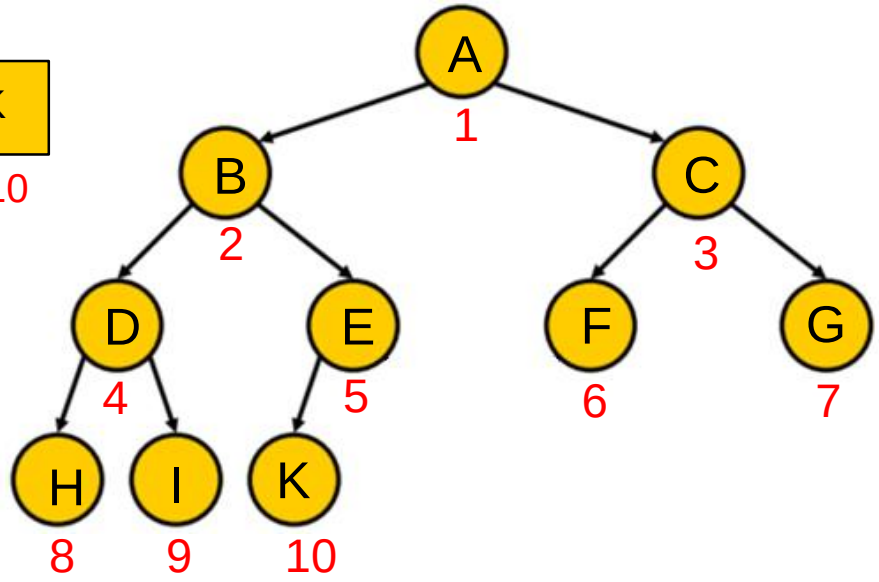
Cài đặt cây nhị phân

- Cài đặt cây nhị phân bằng mảng
  - Hiệu quả với cây nhị phân đầy đủ

Nodes T[]

A	B	C	D	E	F	G	H	I	K
1	2	3	4	5	6	7	8	9	10

Biết một nút ở vị trí i, vị trí nào là của **parent**? **left child**? **right child**?



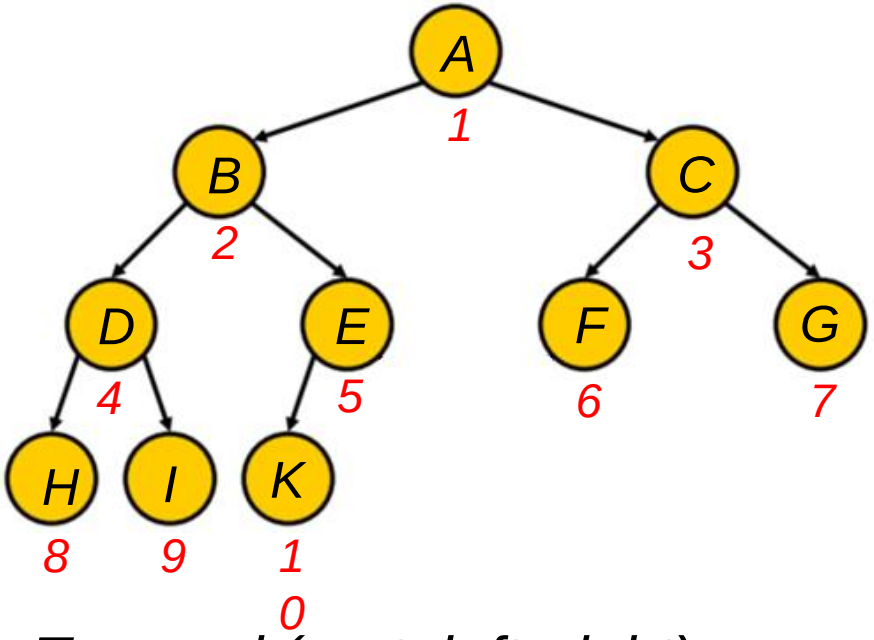
Thuật toán duyệt cây

- Cây nhị phân có thể được định nghĩa một cách đệ quy: cây bao gồm nút gốc, 1 cây con trái và 1 cây con phải
- Duyệt cây là việc thăm tất cả các nút, mỗi nút một lần. Thăm (visit)
- Phép duyệt cây có thể được thực hiện đệ quy
- Vì cây có 2 phần, có 6 cách để duyệt cây theo các thứ tự như sau:
  - *root, left, right*
  - *left, root, right*
  - *left, right, root*
  - *root, right, left*
  - *right, root, left*
  - *right, left, root*

Thuật toán duyệt cây

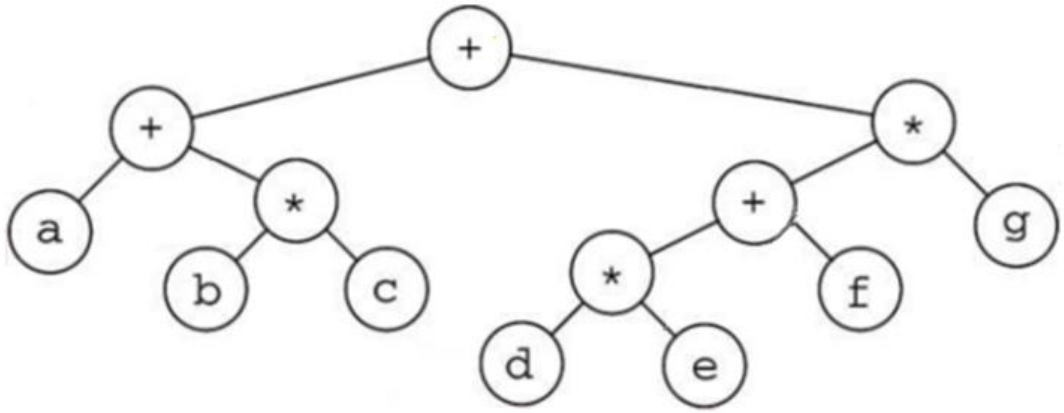
- Cây nhị phân có thể được định nghĩa một cách đệ quy: cây bao gồm nút gốc, 1 cây con trái và 1 cây con phải
- Duyệt cây là việc thăm tất cả các nút, mỗi nút một lần. Thăm (visit)
- Phép duyệt cây có thể được thực hiện đệ quy
- Vì cây có 2 phần, có 6 cách để duyệt cây theo các thứ tự như sau:
  - *root, left, right* ➡ *Pre-order Travesal*
  - *left, root, right* ➡ *In-order Travesal*
  - *left, right, root* ➡ *Post-order Travesal*

Thuật toán duyệt cây



- *Pre-order Travesal (root, left, right)* ➡ *?*
- *In-order Travesal (left, root, right)* ➡ *?*
- *Post-order Travesal (left, right, root)* ➡ *?*

Thuật toán duyệt cây



- *Pre-order Travesal (root, left, right)* ➡ *?*
- *In-order Travesal (left, root, right)* ➡ *?*
- *Post-order Travesal (left, right, root)* ➡ *?*

Thuật toán duyệt cây: Pre-order

- Duyệt thứ tự trước (pre-order), nút gốc được thăm trước
  - Order: Root - Left - Right

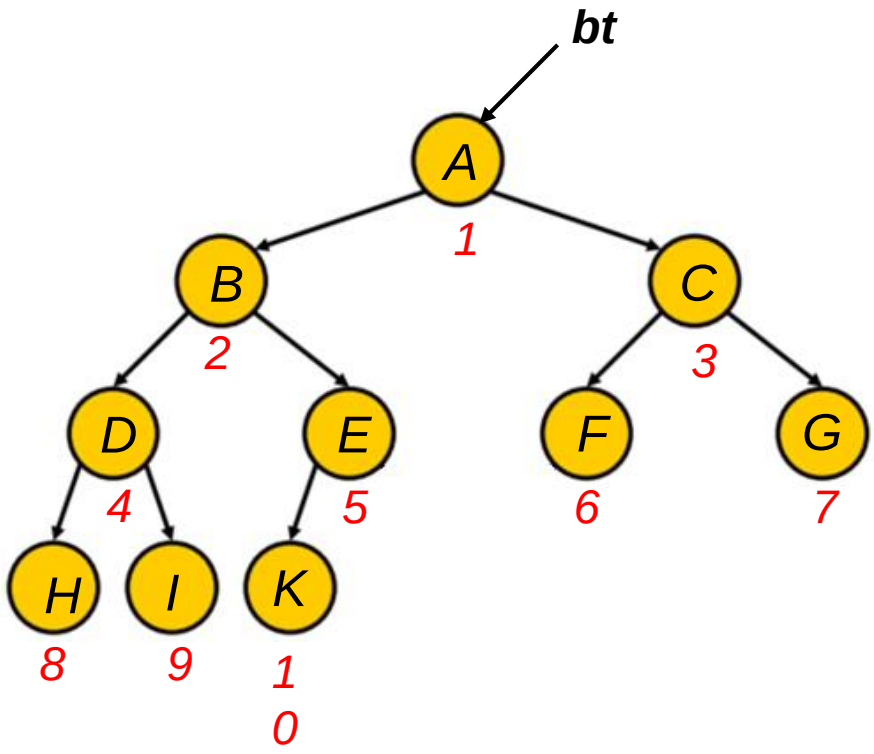
- Thuật toán duyệt thứ tự trước và in ra các phần tử trên cây nhị phân bt:

```
preorderPrint(BinaryTree bt) {
        if (bt == null) return;
        print(bt.element);
        preorderPrint(bt.left);
        preorderPrint(bt.right);
}
```

Thuật toán duyệt cây: Pre-order

- *Ví dụ*

```
preorderPrint(BinaryTree bt) {
        if (bt == null) return;
        print(bt.element);
        preorderPrint(bt.left);
        preorderPrint(bt.right);
}
```



Kết quả: ***A B D H I E K C F G***

Thuật toán duyệt cây: Post-order

- Duyệt thứ tự sau (post-order), nút gốc được thăm sau
  - Order: Left - Right - Root

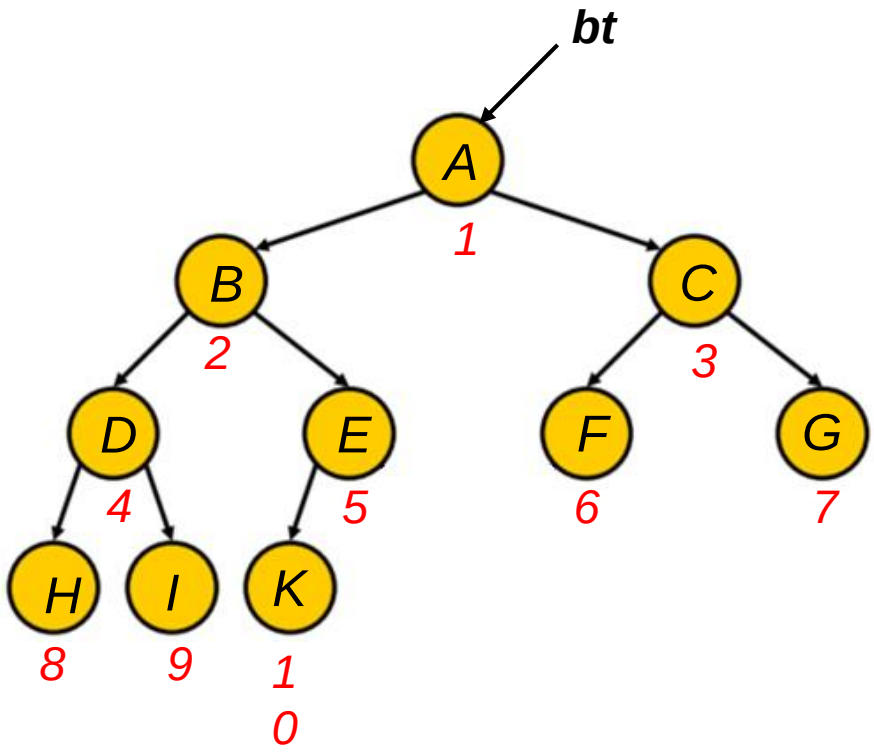
- Thuật toán duyệt thứ tự sau và in ra các phần tử trên cây nhị phân bt:

```
postorderPrint(BinaryTree bt) {
        if (bt == null) return;
        preorderPrint(bt.left);
        preorderPrint(bt.right);
        print(bt.element);
}
```

Thuật toán duyệt cây: Post-order

- *Ví dụ*

```
postorderPrint(BinaryTree bt) {
        if (bt == null) return;
        preorderPrint(bt.left);
        preorderPrint(bt.right);
        print(bt.element);
}
```



Kết quả: ***H I D K E B F G C A***

Thuật toán duyệt cây: In-order

- Duyệt thứ tự giữa (in-order), nút gốc được thăm giữa
  - Order: Left - Root - Right

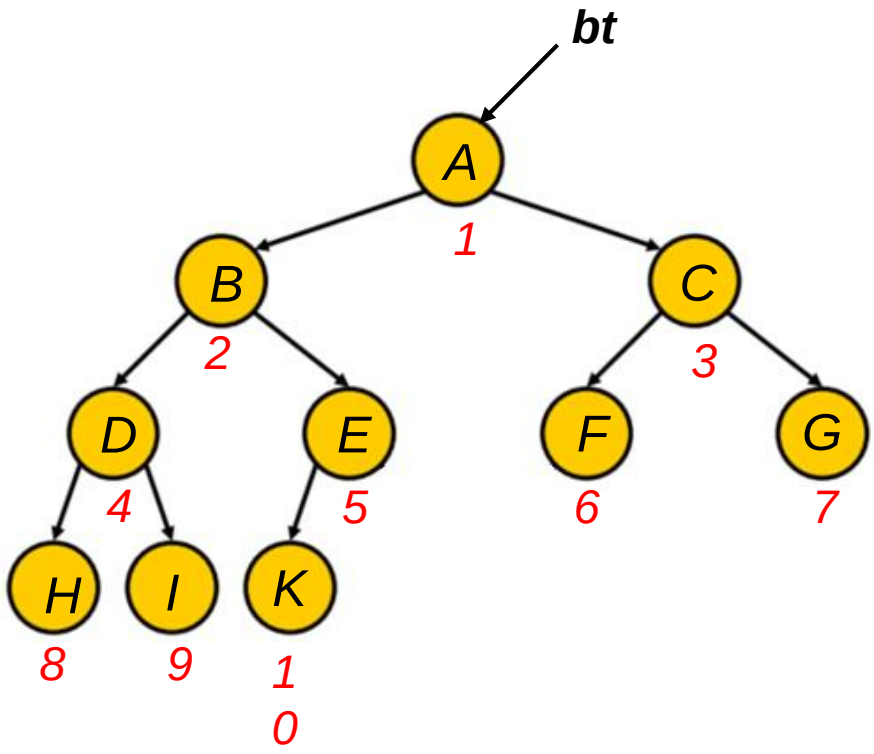
- Thuật toán duyệt thứ tự giữa và in ra các phần tử trên cây nhị phân bt:

```
inorderPrint(BinaryTree bt) {
        if (bt == null) return;
        preorderPrint(bt.left);
        print(bt.element);
        preorderPrint(bt.right);
}
```

Thuật toán duyệt cây: In-order

- *Ví dụ*

```
postorderPrint(BinaryTree bt) {
        if (bt == null) return;
        preorderPrint(bt.left);
        print(bt.element);
        preorderPrint(bt.right);
}
```



Kết quả: ***H D I B K E A F C G***

Ứng dụng của cây nhị phân

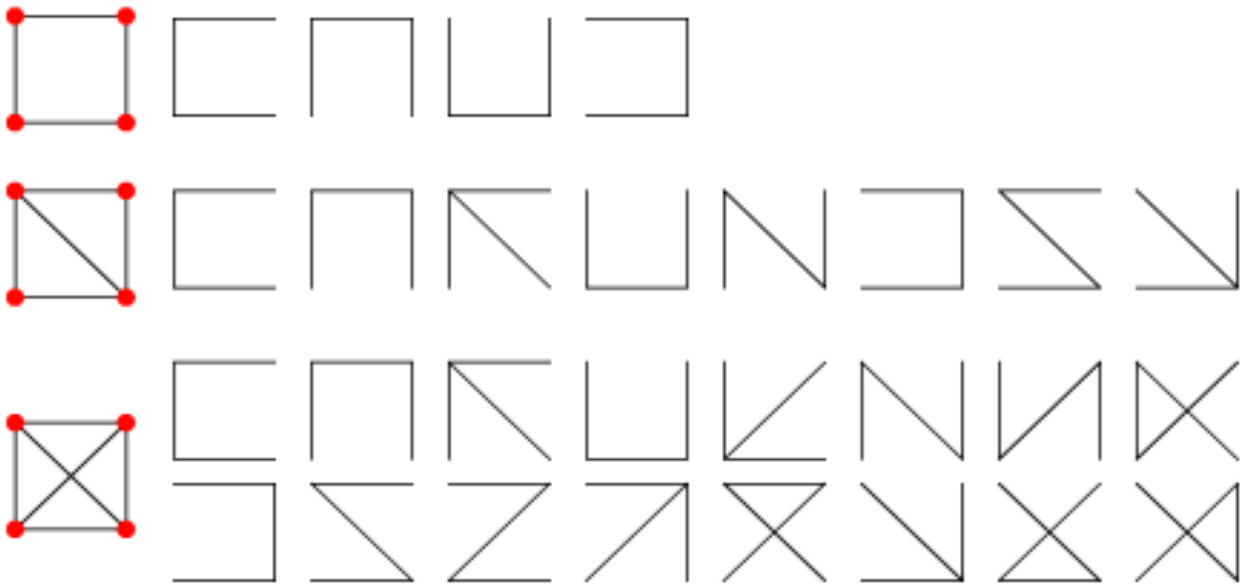
- Cây và cây nhị phân có nhiều ứng dụng trong các bài toán tin học.
- Ứng dụng cây nhị phân:
  - Arithmetic Expression Trees – Cây biểu thức
  - Tìm kiếm hiệu quả (Binary Search Tree)
  - Sắp xếp hiệu quả (Heap Sort)

CÂY BAO TRÙM TỐI THIỂU

MINIMUM SPANNING TREE

Cây bao trùm tối thiểu

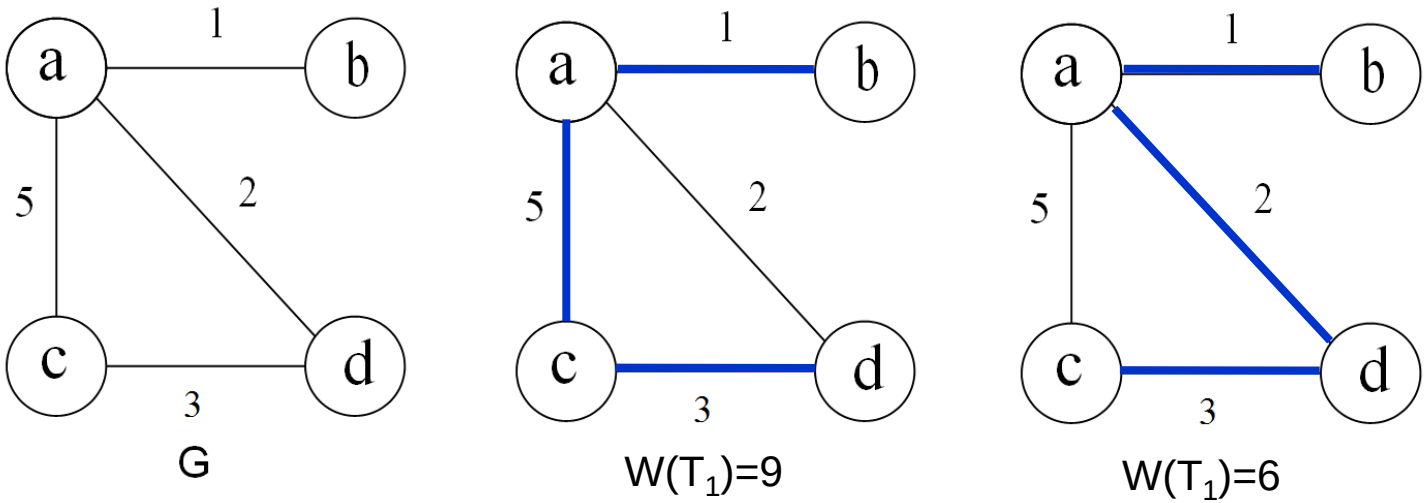
- **Khái niệm: Cây bao trùm (Spanning Tree)**
  - Cây bao trùm của một đồ thị: là cây (gồm các đỉnh, cạnh của G) có chứa tất cả các đỉnh của G.



https://mathworld.wolfram.com/SpanningTree.html

Cây bao trùm tối thiểu

- **Khái niệm Cây bao trùm tối thiểu (Minimum Spanning Tree)**
  - Cây bao trùm tối thiểu của một đồ thị liên thông có trọng số G là cây bao trùm có tổng trọng số các cạnh là nhỏ nhất.

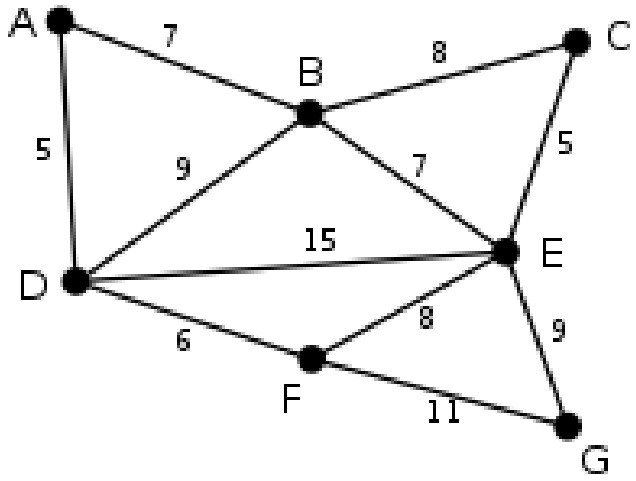


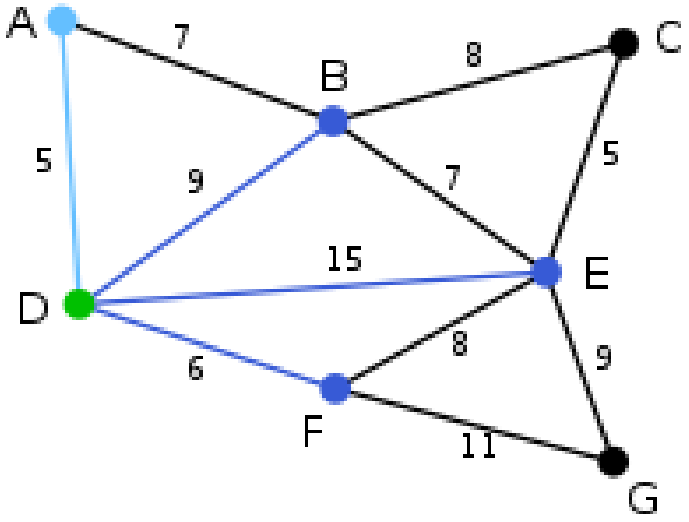
Cây bao trùm tối thiểu

- **Thuật toán Prim**
  - Bắt đầu với cây chỉ có 1 đỉnh T_0.
  - Phát triển cây theo các bước, mỗi bước thêm một đỉnh vào cây đã có bằng một cạnh. Dãy các cây được phát triển $T_1, T_2, \ldots, T_{n-1}$.
  - Chiến lược tham lam: Tại mỗi bước dựng cây T_{i+1} từ cây T_i với việc thêm vào đỉnh "gần nhất".
  - Đỉnh gần nhất với T_i: Đỉnh không thuộc T_i và được nối với T_i bằng cạnh có trọng số nhỏ nhất.
  - Thuật toán dừng lại khi tất cả các đỉnh đã được thêm vào.

Cây bao trùm tối thiểu

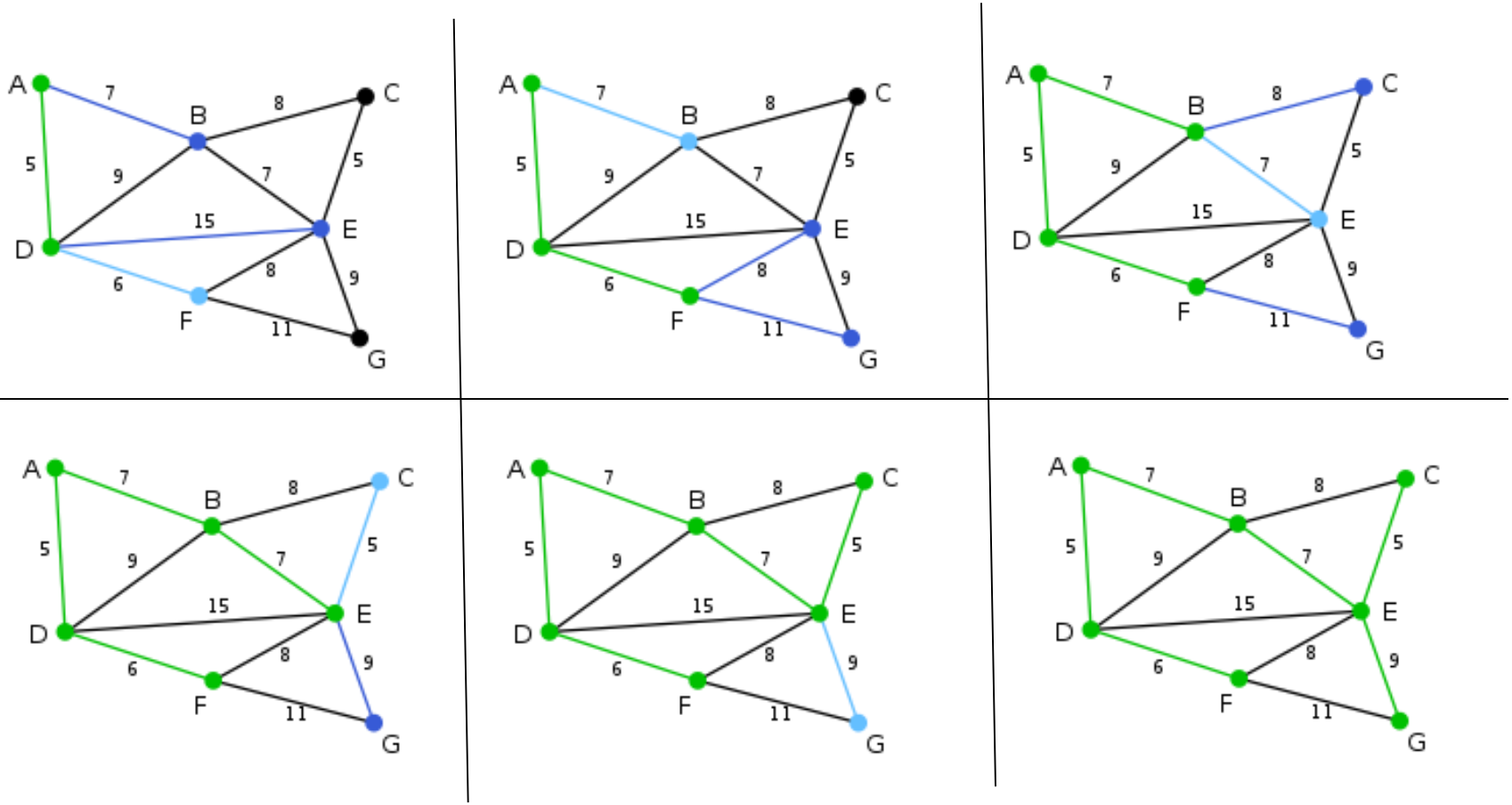
- **Thuật toán Prim**





Cây bao trùm tối thiểu

- **Thuật toán Prim**



http://en.wikipedia.org/wiki/Prim's_algorithm

Cây bao trùm tối thiểu

- **Lược đồ thuật toán Prim**

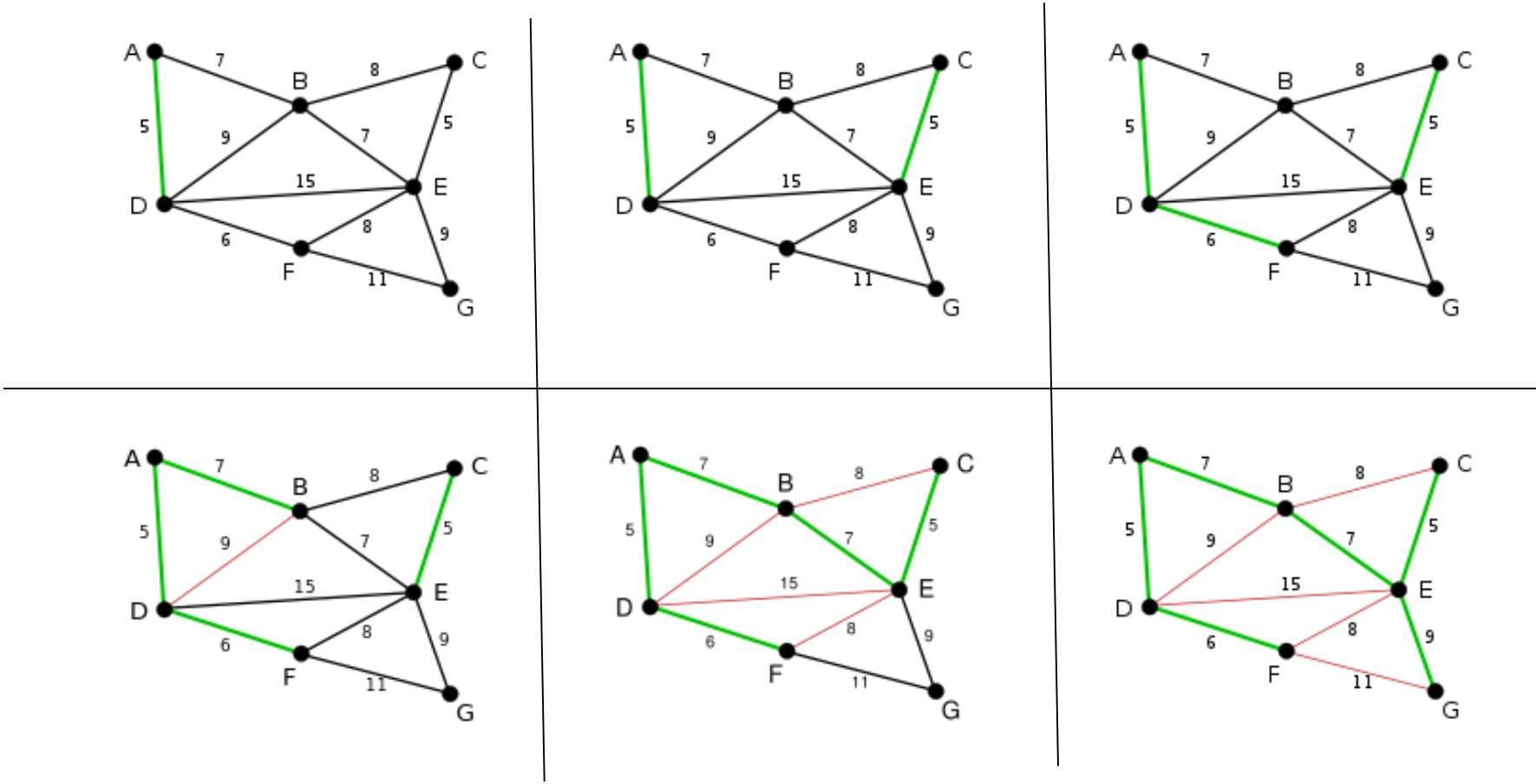
```
Prim(G) ≡
    // Input: G = (V, E)
    // Output: E_T , tập các cạnh của cây bao trùm tối thiểu của G
    V_T = {v_0}
    E_T = {}
    for i=1..|V| - 1
        tìm cạnh có trọng số nhỏ nhất e*=(v*,u*) trong tất cả
        các cạnh (v, u) mà v ∈ V_T và u ∈ V - V_T;
        V_T = V_T U {u*}
        E_T = E_T U {e*}
    endf;
    return G_T=(V_T,E_T)
End.
```

Cây bao trùm tối thiểu

- **Thuật toán Kruskal**
  - Các cạnh được sắp xếp theo thứ tự tăng dần của trọng số.
  - Bắt đầu bằng một rừng (forest) rỗng.
  - Xây dựng MST theo các bước, mỗi bước thêm một cạnh
    - Trong quá trình dựng MST luôn có một "rừng": các cây không liên thông.
    - Thêm vào cạnh có trọng số nhỏ nhất trong các cạnh chưa thêm vào cây và không tạo thành chu trình.
    - Như vậy tại mỗi bước một cạnh có thể:
      - Mở rộng một cây đã có
      - Nối hai cây thành một cây mới
      - Tạo cây mới
  - Thuật toán dừng lại khi tất cả các đỉnh đã được thêm vào.

Cây bao trùm tối thiểu

- **Thuật toán Kruskal**



http://en.wikipedia.org/wiki/Kruskal%27s_algorithm

Cây bao trùm tối thiểu

- **Lược đồ thuật toán Kruskal**

```
Kruskal(G) ≡
    // Input: G = (V, E)
    // Output: E_T , tập các cạnh của cây bao trùm tối thiểu của G
    V_T = {}
    E_T = {}
    for i=1..n - 1
        tìm cạnh có trọng số nhỏ nhất e=(v,u) mà khi thêm vào
        G_T=(V_T,E_T)không tạo thành chu trình;
        V_T = V_T U {u,v}
        E_T = E_T U {e}
    endf;
    return G_T=(V_T,E_T)
End.
```

Cây bao trùm tối thiểu

- **Ứng dụng**
  - Thiết kế mạng (Network Design): máy tính, viễn thông, giao thông, điện, nước…
  - Thiết kế các thuật toán xấp xỉ cho bài toán NP-khó: ví dụ bài toán TSP.
  - Bài toán phân cụm dữ liệu (Clustering).

https://www.geeksforgeeks.org/applications-of-minimum-spanning-tree/
http://www.utdallas.edu/~besp/teaching/mst-applications.pdf

Bài tập thực hành

- **Triển khai thuật toán tìm cây bào trùm tối thiểu, phát triển ứng dụng**
  - Thuật toán Prim, Kruskal
  - Tìm hiểu thuật toán phân cụm dữ liệu biểu diễn gen theo tiếp cận sử dụng cây bao trùm tối thiểu.
  - Tài liệu:
    - Kenneth H. Rosen, section 11.4, 11.5
    - Xu, Y., Olman, V., & Xu, D. (2002). *Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees*. Bioinformatics, 18(4), 536–545. doi:10.1093/bioinformatics/18.4.536