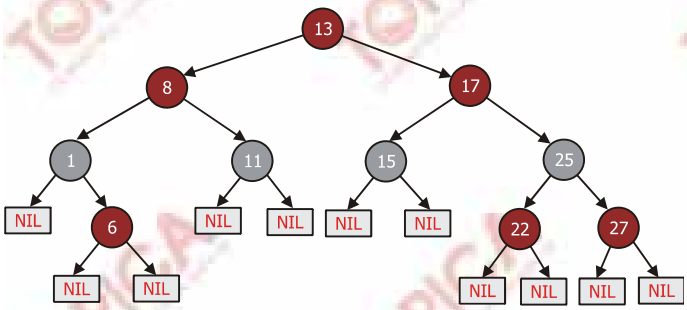
BÀI 7: CÂY



Nội dung

- Định nghĩa cây
- Các đặc trưng của cây
- Các ví dụ về cây
- Cây khung
- Rừng, rừng khung
- Cây phân cấp, các ứng dụng của cây

Giới thiệu

Cây là một lớp quan trọng của đồ thị có nhiều ứng dụng. Khái niệm cây được đưa ra lần đầu tiên vào năm 1857 bởi nhà toán học người Anh, Athur Cayley khi ông dùng nó để đếm các đồng đẳng của hợp chất hydrocacbon no C_nH_{2n+2}, từ đó cây được dùng trong nhiều lĩnh vực khác nhau, đặc biệt là trong tin học. Chẳng hạn, người ta đã xây dựng được nhiều thuật toán tìm kiếm hiệu quả trên cây, nhờ đó cây được dùng như một tổ chức dữ liệu cơ bản trong nhiều ứng dụng. Cây cũng cho mô hình xây dựng mạng máy tính với chi phí rẻ nhất cho các kênh thoại. Cây cũng là một công cụ tạo ra các mã hiệu quả trong việc lưu trữ và truyền dữ liệu, ...

Mục tiêu

Sau khi học bài này, các bạn có thể:

- Trình bày được định nghĩa, các đặc trưng và cho ví dụ về cây.
- Tìm, kiệt kê và đếm được số cây khung của một đồ thị
- Hiểu rõ khái niệm rừng, rừng khung
- Ứng dụng được các định nghĩa về cây phân cấp, cây nhị phân qua một số trường hợp điển hình.

Thời lượng học

- 6 tiết

TÌNH HUỐNG DẪN NHẬP

Tình huống: Mạng máy tính

Một mạng máy tính bao gồm một số đường nối hai máy với nhau. Hai máy được xem như trao đổi được thông tin với nhau nếu chúng có một đường nối trực tiếp hoặc một đường truyền đi qua một số máy của mạng. Thời gian truyền tin là một đơn vị khi đi qua một cạnh nối.

Câu hỏi

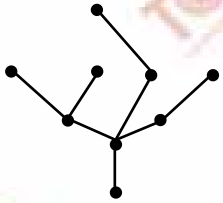
Tìm một thuật toán chọn một máy làm máy chủ và lược bỏ một số cạnh nối của mạng, sao cho từ máy chủ có duy nhất một đường truyền tin đến mỗi máy trong mạng và mọi máy của mạng nhận được tin với thời gian ngắn nhất.

Cây là một lớp quan trọng của đồ thị có nhiều ứng dụng. Khái niệm cây được đưa ra lần đầu tiên vào năm 1857 bởi nhà toán học người Anh, Athur Cayley khi ông dùng nó để đếm các đồng đẳng của hợp chất hydrocacbon no C_nH_{2n+2}, từ đó cây được dùng trong nhiều lĩnh vực khác nhau, đặc biệt là trong tin học. Chẳng hạn, người ta đã xây dựng được nhiều thuật toán tìm kiếm hiệu quả trên cây, nhờ đó cây được dùng như một tổ chức dữ liệu cơ bản trong nhiều ứng dụng. Cây cũng cho mô hình xây dựng mạng máy tính với chi phí rẻ nhất cho các kênh thoại. Cây cũng là một công cụ tạo ra các mã hiệu quả trong việc lưu trữ và truyền dữ liệu, ...

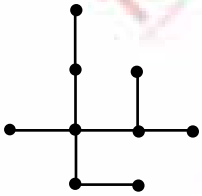
7.1. Định nghĩa cây

Cây được định nghĩa là một đồ thị vô hướng, liên thông, không có chu trình.

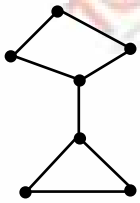
Hai hình vẽ đầu dưới đây là những cây, còn hình vẽ cuối là không phải:



Cây



Cây



Không phải cây

Từ định nghĩa cây suy ra cây không thể có khuyên và có cạnh bội, nghĩa là cây phải là đơn đồ thị.

Điều kiện không có chu trình của cây là rất quan trọng, nó quyết định một số đặc trưng của cây mà các đồ thị khác không có. Điều này cũng giải thích cho nhiều ứng dụng của cây.

7.2. Các đặc trưng của cây

Trước hết ta nhận xét rằng một cây có nhiều hơn một đỉnh luôn có ít nhất hai đỉnh treo (đỉnh bậc 1). Thật vậy, xét một đường đi đơn dài nhất trong cây (đường đi này luôn tồn tại do tính hữu hạn). Hai đỉnh đầu mút của đường đi này phải là đỉnh treo vì nếu trái lại, một cạnh mới sẽ được thêm vào các đầu mút này, khi đó hoặc ta nhận được một đường đi đơn dài hơn đường đi dài nhất đang xét, hoặc ta nhận được một chu trình trong cây, đó là điều mâu thuẫn.

Cây có nhiều định nghĩa tương đương, phản ánh các đặc điểm khác nhau của cây như nội dung định lý dưới đây:

Định lý. Giả sử T là một đồ thị n đỉnh. Các mệnh đề sau đây là tương đương:

1) T là một cây.
2) T không có chu trình và có $n - 1$ cạnh.
3) T liên thông và có $n - 1$ cạnh.
4) T liên thông và mỗi cạnh của T đều là cầu (nghĩa là nếu bỏ đi một cạnh bất kỳ của T thì T sẽ không liên thông).
5) Giữa hai đỉnh bất kỳ của T đều tồn tại duy nhất một đường đi.
6) T không chứa chu trình nhưng nếu thêm vào một cạnh mới nối hai đỉnh của T thì ta nhận được một chu trình duy nhất.

Chứng minh. Định lý được chứng minh theo kỹ thuật suy diễn vòng tròn:

$1) \Rightarrow 2) \Rightarrow 3) \Rightarrow 4) \Rightarrow 5) \Rightarrow 6) \Rightarrow 1)$

1) $\Rightarrow$ 2) : Quy nạp theo số đỉnh n. Với n = 1, kết quả là hiển nhiên. Giả sử n > 1 và kết quả đúng với mọi cây có n−1 đỉnh. Trong cây đang xét bỏ đi đỉnh treo (mà sự tồn tại đã được nhận xét) và cạnh ứng với nó, ta được một cây n−1 đỉnh. Theo giả thiết quy nạp, cây này có n−2 cạnh. Vậy cây đang xét có n−1 cạnh.

2) $\Rightarrow$ 3) : Giả sử T có k $\geq$ 1 thành phần liên thông T_1, T_2,..., T_k. Vì T không có chu trình nên các thành phần này đều là cây. Gọi tương ứng số đỉnh và số cạnh của các cây này là:

$$n_1, m_1, n_2, m_2, ..., n_k, m_k,$$

Ta được: $n = n_1 + n_2 +...+ n_k$ (số đỉnh của T)

$n - 1 = m_1 + m_2 +...+ m_k$ (số cạnh của T).

Mặt khác do T_1, T_2, ...,T_k là những cây nên:

$$m_1 = n_1 - 1, m_2 = n_2 - 1, ..., m_k = n_k - 1.$$

Từ đó nhận được:

$$n - 1 = (n_1 + n_2 +...+ n_k) - k = n - k, \text{ hay } k = 1.$$

Vậy T là đồ thị liên thông.

3) $\Rightarrow$ 4) : Khi bỏ đi một cạnh bất kỳ của T ta nhận được đồ thị n đỉnh, n−2 cạnh nghĩa là đồ thị không liên thông (bài tập 10, Bài 5).

4) $\Rightarrow$ 5) : Việc tồn tại đường đi giữa hai đỉnh là do T liên thông, còn nếu có hai đường đi nối hai đỉnh nào đó thì các cạnh trên các đường đi này đều không phải là cầu.

5) $\Rightarrow$ 6) : Nếu T có chu trình thì bất cứ hai đỉnh nào thuộc chu trình này đều có hai đường đi đơn nối chúng. Bây giờ thêm vào T một cạnh mới nối hai đỉnh u, v nào đó của T, khi đó cạnh này, cùng với đường đi duy nhất nối u, v sẽ tạo ra một chu trình duy nhất trong T.

6) $\Rightarrow$ 1) : Giả sử T không liên thông nghĩa là có ít nhất hai thành phần liên thông. Khi đó nối hai đỉnh bất kỳ, mỗi đỉnh thuộc một thành phần liên thông, ta sẽ không tạo ra một chu trình nào cả. Điều này mâu thuẫn với 6).

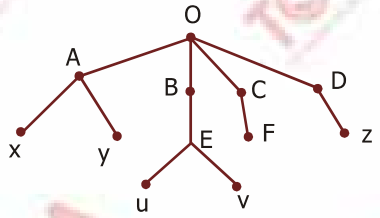
Định lý vừa chứng minh cho thấy có thể dùng một trong các mệnh đề từ 2) đến 6) làm định nghĩa của cây. Mỗi một định nghĩa cho ta một khía cạnh đặc trưng của cây, nói chung nó đều phản ánh một đặc điểm: cây là một đồ thị vô hướng liên thông *đơn giản nhất*. Dưới đây là một số nhấn mạnh những đặc tính đơn giản của cây:

- Số cạnh và số đỉnh của cây có liên hệ ràng buộc chặt: *số cạnh = số đỉnh−1*. Điều này nói lên cây có tính xác định hơn các đồ thị khác.
- Cây là một đồ thị liên thông có *số cạnh tối thiểu*, nghĩa là không thể bớt đi một cạnh nào để vẫn đảm bảo tính liên thông. Điều này giải thích tại sao cây là mô hình xây dựng mạng liên thông với chi phí rẻ nhất cho các cạnh nối.
- Giữa hai đỉnh của cây *có duy nhất* một đường đi vì vậy việc tìm kiếm trên cây là nhanh nhất. Điều này khiến cho cây được dùng như một tổ chức lưu trữ dữ liệu hiệu quả trong nhiều ứng dụng.

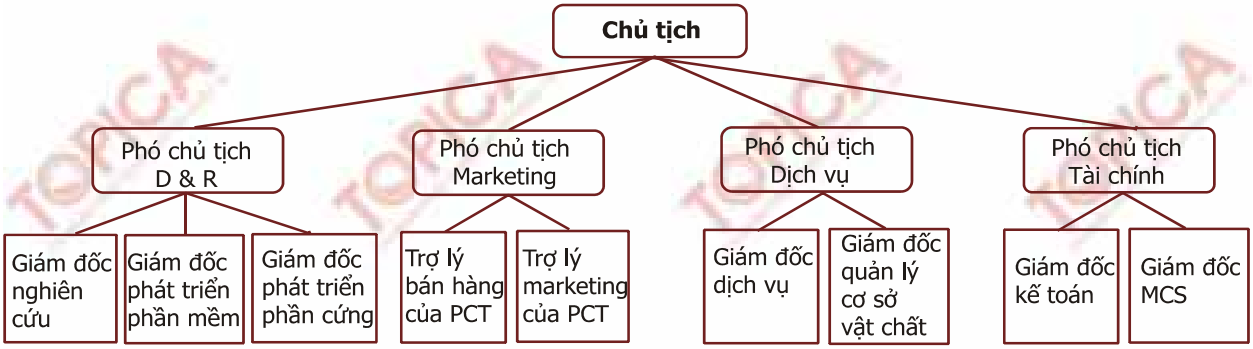
7.3. Các ví dụ về cây

Cây được gặp trong rất nhiều mô hình ứng dụng. Dưới đây chỉ giới thiệu những mô hình tiêu biểu nhất để thấy rõ những ứng dụng đa dạng của cây.

- **Cây thư mục.** Là một cách tổ chức lưu trữ các file trên đĩa của hệ điều hành. Từ thư mục gốc, có các cạnh nối đến các thư mục con và các file chứa trong thư mục này. Từ thư mục con A lại có các đường nối đến các thư mục con của nó và các file chứa trong nó, ... Mô hình là liên thông và không có chu trình. Nhờ giữa hai đỉnh có duy nhất một đường đi mà hệ điều hành nhanh chóng tìm đến thư mục hoặc file cần truy cập. Dưới đây là hình ảnh một cây thư mục, trong đó các chữ cái lớn là tên các thư mục (O là thư mục gốc), còn các chữ cái nhỏ là tên các file:

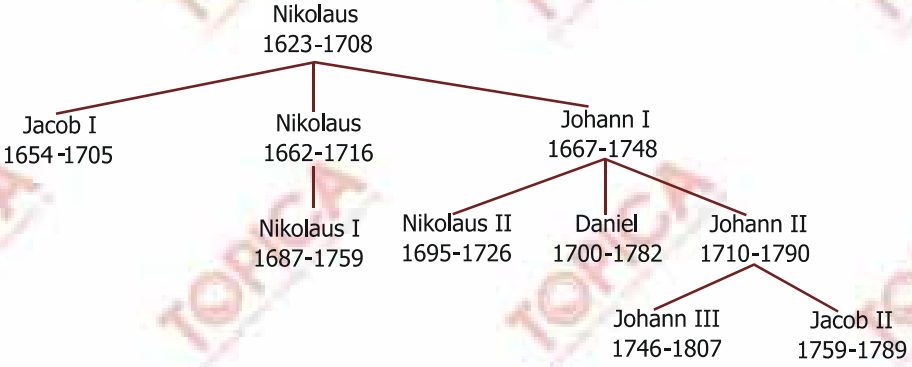


- **Cây hành chính.** Là một cây mô tả cách quản lý hành chính lý tưởng trong các tổ chức (quốc gia, công ty, nhà máy, ...). Cây mô tả các mối quan hệ trực tiếp (cạnh nối) giữa các đơn vị hay cá nhân trong tổ chức (đỉnh). Nhờ không có chu trình, việc quản lý được phân cấp trách nhiệm một cách minh bạch, không có các chồng chéo, đảm bảo được khẩu hiệu "một dấu, một cửa", ... Dưới đây là một cây mô tả tổ chức trong một công ty máy tính:



- **Cây gia phả.** Cây gia phả là một đồ thị mô tả mối quan hệ huyết thống của một dòng họ. Mỗi đỉnh là một thành viên, mỗi cạnh nối biểu thị mối quan hệ cha – con. Rõ ràng đồ thị không thể có chu trình.

  Dưới đây là cây gia phả gồm 4 đời của dòng họ Bernoulli, một gia đình toán học nổi tiếng người Thụy sĩ:

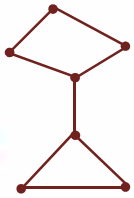


7.4. Cây khung

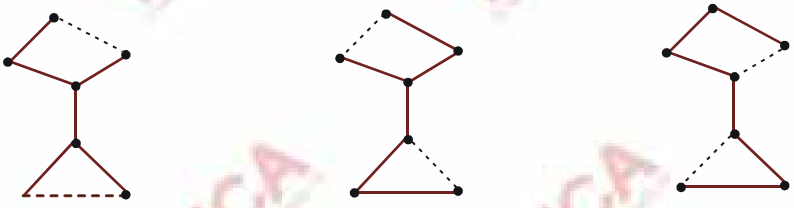
7.4.1. Định nghĩa

Cho một đồ thị vô hướng liên thông G. Nếu tất cả các cạnh của nó đều là cầu thì đồ thị đang xét là một cây. Trái lại, ta bỏ đi một cạnh nào đó không phải là cầu, đồ thị nhận được sẽ vẫn liên thông. Quá trình bỏ cạnh không phải là cầu được lặp lại cho đến khi tất cả các cạnh của đồ thị nhận được đều là cầu, khi đó ta nhận được một cây T. Cây T nhận được từ G như vậy được gọi là *cây khung* của đồ thị đã cho G. Như vậy cây khung T của G là một cây bao gồm một số cạnh của G sao cho tập đỉnh vẫn là tập đỉnh của G. Vì thế cây khung còn có tên gọi là *cây bao trùm*.

Chú ý: Cây khung của một đồ thị là không duy nhất vì trong khi bỏ cạnh ta có nhiều phương án lựa chọn khác nhau. Chẳng hạn đồ thị dưới đây:



có thể có những cây khung khác nhau như sau (các cạnh không liền nét là các cạnh được chọn bỏ đi):



Từ định nghĩa cây khung, dễ dàng nhận thấy rằng một đồ thị vô hướng liên thông bao giờ cũng có cây khung và ngược lại, một đồ thị vô hướng có cây khung là một đồ thị liên thông. Trường hợp đồ thị đã là một cây thì cây khung của nó là chính nó.

Cây khung của đồ thị cho ta một mô hình "giản lược" đồ thị sao cho vẫn đảm bảo tính liên thông với số cạnh giữ lại là ít nhất. Vì thế một ứng dụng quan trọng của cây khung là cung cấp một mô hình xây dựng một mạng liên thông với chi phí cho các cạnh nối là rẻ nhất. Trong **Bài 8** ta sẽ bàn kỹ vấn đề này.

7.4.2. Cách xây dựng cây khung

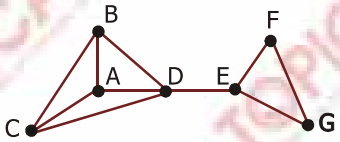
Có thể xây dựng cây khung của một đồ thị bằng cách bỏ dần các cạnh như trong định nghĩa. Tuy nhiên cách này không hiệu quả vì mỗi lần chọn cạnh để bỏ ta phải thử lại cạnh đó không phải là cầu. Thay vì chọn phương án bỏ cạnh, ta có thể xây dựng một cây khung bằng cách nạp dần các cạnh của đồ thị sao cho tại mỗi bước không có chu trình nào xuất hiện bằng thao tác tìm kiếm như sau:

- Khởi tạo cây T là rỗng (chưa có cạnh nào).
- Chọn một đỉnh xuất phát (tùy ý), tiến hành tìm kiếm bắt đầu từ đỉnh đó (theo chiều rộng hoặc theo chiều sâu). Cứ mỗi lần từ đỉnh u đến được đỉnh v thì cạnh (u, v) được kết nạp vào T. Vì không thăm những đỉnh đã thăm rồi nên các cạnh được nạp không tạo thành chu trình.

- Khi kết thúc tìm kiếm, nếu mọi đỉnh đều được thăm (nghĩa là đồ thị đang xét là liên thông) thì T là cây khung cần tìm, trái lại đồ thị đang xét là không liên thông, nghĩa là nó không có cây khung.

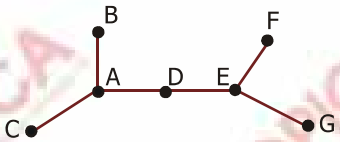
Chú ý: Cây khung được tìm theo cách vừa nêu là không duy nhất, nó phụ thuộc vào đỉnh xuất phát và thứ tự các đỉnh được thăm. Việc viết một chương trình minh họa xem như bài tập.

Ví dụ. Dùng tìm kiếm theo chiều rộng xuất phát từ A, xây dựng một cây khung của đồ thị sau:

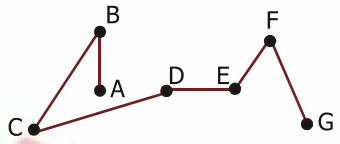


Hàng đợi Q	Đỉnh lấy ra	Đỉnh đưa vào	Cạnh nạp vào T
A	A	B, C, D	AB, AC, AD
B, C, D	B		
C, D	C		
D	D	E	DE
E	E	F, G	EF, EG
F, G	F		
G	G		
rỗng			

Kết quả tìm kiếm cho ta cây khung T gồm các cạnh AB, AC, AD, DE, EF, FG

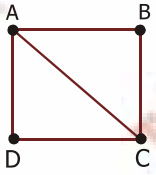


Làm lại ví dụ trên nhưng tìm theo chiều sâu bắt đầu từ A (thứ tự duyệt theo trật tự từ điển của tên đỉnh), ta được cây khung khác gồm các cạnh AB, BC, CD, DE, EF, FG



7.4.3. Liệt kê (đếm) cây khung

Việc đếm tất cả các cây khung của một đồ thị nói chung là không đơn giản. Trong trường hợp tổng quát nhất ta phải dựa vào cách liệt kê. Thông thường việc liệt kê dựa vào việc kết nạp dần các cạnh của đồ thị sao cho cạnh được nạp không tạo thành chu trình với những cạnh đã nạp trước. Khi được n−1 cạnh (n là số đỉnh của đồ thị), ta thu được một cây khung. Quá trình được tiến hành quay lui trên danh sách cạnh để có thể duyệt tất cả khả năng xảy ra. Ví dụ, đồ thị dưới đây:



Gồm 4 đỉnh và 5 cạnh là AB, AC, AD, BC, CD. Để liệt kê tất cả các cây khung, ta duyệt tất cả các khả năng lấy ra 3 cạnh không tạo thành chu trình bằng cách quay lui trên danh sách cạnh đã nêu của đồ thị và nhận được lần lượt 8 cây khung như sau:

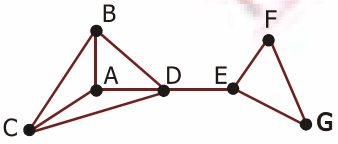
1) AB, AC, AD
2) AB, AC, CD
3) AB, AD, BC
4) AB, AD, CD
5) AB, BC, CD
6) AC, AD, BC
7) AC, BC, CD
8) AD, BC, CD

Ví dụ trên cũng cho thấy số cây khung của đồ thị nói chung là rất lớn. Đồ thị càng nhiều cạnh thì càng nhiều cây khung. Khi giải bài toán đếm số đồng đẳng của hợp chất hydrocacbon no C_nH_{2n+2}, Cayley đã dẫn về việc đếm số cây khung của đồ thị đầy đủ K_n (là đơn đồ thị liên thông n đỉnh với số cạnh nhiều nhất) và tìm được kết quả đếm nổi tiếng dưới đây:

Định lý Cayley. Số cây khung của đồ thị đầy đủ K_n bằng n^{n-2} ($n > 2$).

Kết quả của Cayley cho ta một đánh giá trên của số lượng các cây khung của một đơn đồ thị liên thông n đỉnh, con số này là rất lớn, chẳng hạn một đơn đồ thị liên thông 10 đỉnh (số đỉnh là không lớn) có thể có đến 100 triệu cây khung. Kết quả này cũng cho thấy những bài toán phải duyệt tất cả các cây khung là những bài toán có độ phức tạp không khả thi.

Mặc dù trong trường hợp tổng quát, phải dùng cách liệt kê đếm cây khung, nhưng trong một số đồ thị cụ thể, do những nhận xét riêng, ta có thể vận dụng các nguyên lý đếm để tính được số cây khung mà không phải liệt kê. Chẳng hạn quay lại đồ thị cho trong thí dụ ở mục 7.4.2



Ta nhận thấy mọi cây khung của đồ thị này đều được ghép từ một cây khung của đồ thị con ABCD, cạnh DE và một cây khung của đồ thị con EFG, từ đó theo nguyên lý nhân, số cây khung của đồ thị đang xét bằng số cây khung của đồ thị ABCD nhân với số cây khung của đồ thị EFG. Đồ thị ABCD là đồ thị đầy đủ K_4 và đồ thị EFG là đồ thị đầy đủ K_3 nên theo định lý Cayley, số cây khung của ương ứng của chúng là 16 và 3, Từ đó nhận được số cây khung của đồ thị đang xét là $16 \times 3 = 48$.

7.4.4. Chu số

Để thu được một cây khung, ta cần lược bớt một số cạnh (không phải là cầu) của đồ thị (liên thông) đã cho. Giả sử đồ thị này có m cạnh, n đỉnh. Gọi số cạnh bỏ đi là c. Do cây khung thu được luôn có n−1 đỉnh nên ta nhận được hệ thức:

$$m - c = n - 1 \text{ hay } c = m - n + 1.$$

Số c nhận được bằng hệ thức trên, chỉ phụ thuộc vào đồ thị đang xét mà không phụ thuộc vào cây khung nhận được. Nó được gọi là *chu số* của đồ thị đã cho. Sở dĩ có tên gọi như vậy vì nó đặc tả số chu trình có trong đồ thị. Chu số của đồ thị càng lớn thì đồ thị càng nhiều chu trình (và do đó càng phức tạp). Cây là đồ thị liên thông có chu số bằng 0.

Chẳng hạn, xét đồ thị đầy đủ K_n. Đồ thị này có n đỉnh và có số cạnh bằng

$$m = \frac{n(n-1)}{2}$$

từ đó ta nhận được chu số của nó là:

$$c = \frac{n(n-1)}{2} - n + 1 = \frac{n(n-3)}{2} + 1$$

Đây là đồ thị có chu số lớn nhất (do đó cũng phức tạp nhất) trong lớp các đơn đồ thị liên thông n đỉnh.

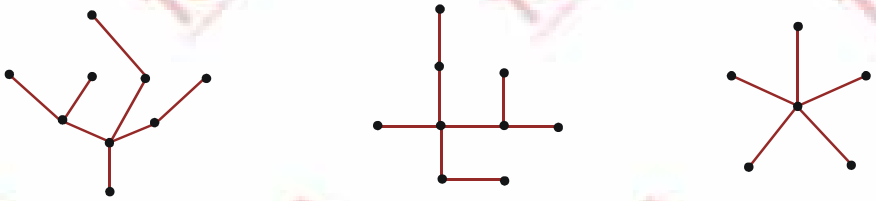
Định nghĩa chu số vừa nêu được áp dụng cho các đồ thị vô hướng liên thông. Tuy nhiên nó dễ dàng mở rộng cho một đồ thị vô hướng bất kỳ bằng cách mở rộng khái niệm "cây" thành khái niệm "rừng" như trong trình bày của mục dưới đây.

7.5. Rừng và rừng khung

7.5.1. Định nghĩa rừng

Nếu trong định nghĩa cây, ta chỉ giữ lại điều kiện *không có chu trình* còn bỏ qua điều kiện liên thông thì đồ thị được gọi là *rừng*.

Do một đồ thị vô hướng bao giờ cũng được phân hoạch thành các thành phần liên thông nên nếu nó là một rừng thì các thành phần liên thông của nó sẽ là những cây. Như thế một rừng có thể xem như nhiều cây hợp lại và cây được xem như một trường hợp riêng của rừng (rừng chỉ có một cây). Chẳng hạn rừng dưới đây gồm 3 cây:



Như vậy những khái niệm được xây dựng trên cây, dễ dàng được mở rộng một cách tự nhiên cho rừng.

7.5.2. Rừng khung

Giống như việc xây dựng cây khung, ta lược bỏ các cạnh của một đồ thị vô hướng (không nhất thiết liên thông) một cách tối đa sao cho không làm tăng số thành phần liên thông của đồ thị đang xét. Khi đó nhận được một rừng, được gọi là *rừng khung* của đồ thị đã cho. Dễ dàng thấy rằng rừng khung này gồm các cây khung của các thành phần liên thông của đồ thị đang xét.

Giả sử đồ thị đang xét gồm m cạnh, n đỉnh và p thành phần liên thông. Gọi c là số cạnh bị lược bỏ của đồ thị để thu được rừng khung, ta nhận được:

$$c = c_1 + c_2 + \ldots + c_p$$

trong đó c_1, c_2, ..., c_p là các chu số tương ứng của các thành phần liên thông. Một cách tự nhiên, c được gọi là *chu số* của đồ thị đang xét và từ các công thức xác định các c_i (i = 1, 2, ..., p), ta dễ dàng nhận được công thức xác định c:

$$c = m - n + p$$

mà nó được xem như sự mở rộng tự nhiên của định nghĩa chu số của đồ thị liên thông (p = 1).

7.6. Cây phân cấp (cây có gốc)

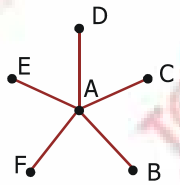
Có thể định hướng các cạnh của cây để thu được một đồ thị có hướng. Một dạng định hướng quan trọng của cây là cây phân cấp.

7.6.1. Các định nghĩa

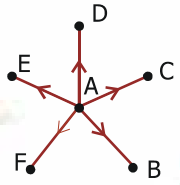
Chọn một đỉnh nào đó trong cây gọi là đỉnh gốc. Vì từ gốc có duy nhất một đường đi (đơn) đến mọi đỉnh khác nên ta định hướng các cạnh theo chiều của các đường đi này. Cây định hướng thu được, được gọi là *cây phân cấp* (hay là *cây có gốc*). Như vậy, cây phân cấp là một cây có hướng, trong đó có một đỉnh đặc biệt gọi là *gốc*, sao cho từ gốc có đường đi (có hướng) duy nhất đến mọi đỉnh khác. Chú ý rằng, hướng của các cạnh được xác định duy nhất từ gốc đã chọn và khi thay đổi gốc, ta nhận được các cây phân cấp khác nhau.

Để xác định đường đi từ gốc đến các đỉnh khác, ta có thể dùng các thuật toán tìm kiếm (theo chiều rộng hoặc theo chiều sâu) đã thảo luận trong Bài 6, trong đó mỗi lần từ đỉnh u thăm đỉnh v, ta gán hướng cho cạnh đi từ u đến v.

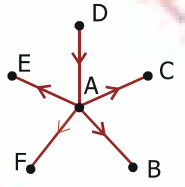
Ví dụ, từ cây vô hướng:



bằng cách chọn gốc A, ta nhận được cây phân cấp như sau:

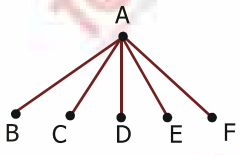


còn nếu chọn gốc D, ta được một cây phân cấp khác:

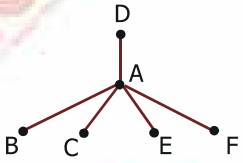


Thông thường để biểu diễn một cây phân cấp trên mặt phẳng, người ta vẽ các đỉnh thành từng tầng từ trên xuống dưới: trên cùng là gốc (đỉnh mức 0), tầng tiếp theo là các đỉnh kề với gốc (các đỉnh mức 1), tầng tiếp theo là các đỉnh kề với đỉnh mức 1 (các đỉnh mức 2),... Với cách vẽ này, hướng của các cạnh luôn đi từ các đỉnh tầng trên

xuống các đỉnh tầng dưới, vì thế không cần phải vẽ các mũi tên trên các cạnh. Chẳng hạn, cây phân cấp gốc A và cây phân cấp gốc D trong thí dụ trên có thể vẽ lại (không có mũi tên) như sau:



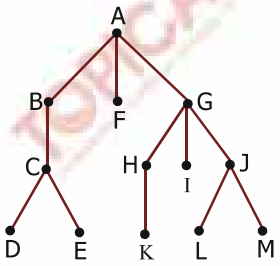
Cây gốc A (2 tầng)



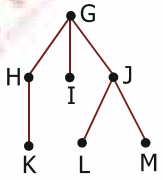
Cây gốc D (3 tầng)

Các ví dụ trong mục 7.3 đều là những cây phân cấp. Một số thuật ngữ trên cây phân cấp bắt nguồn từ cây gia phả. Nếu có cạnh có hướng nối từ đỉnh u đến đỉnh v thì u được gọi là *cha* của v và v được gọi là *con* của u. Rõ ràng, trừ đỉnh gốc, các đỉnh khác của cây phân cấp, mỗi đỉnh đều có duy nhất một cha. Các đỉnh có chung một cha được gọi là *anh em*. Các đỉnh đi trước v trên đường đi từ gốc đến v được gọi là *tổ tiên* của v, còn các đỉnh đi sau v trên các đường đi từ gốc qua v được gọi là *con cháu* của v. Mọi đỉnh khác gốc trên cây phân cấp đều là con cháu của gốc (cũng có nghĩa gốc là tổ tiên của mọi đỉnh còn lại). Các đỉnh của cây phân cấp không có con được gọi là *lá*, hay *đỉnh ngoài*. Các đỉnh có con (cũng có nghĩa không phải là lá) được gọi là *đỉnh trong*. Nếu u là một đỉnh của cây phân cấp thì *cây con với gốc u* là đồ thị con của cây đang xét, bao gồm u và các con cháu của nó cùng với tất cả các cạnh liên thuộc. Người ta cũng gọi *chiều cao* của một cây phân cấp là độ dài của đường đi dài nhất từ gốc đến lá (nghĩa là bằng số tầng của cây trừ đi 1).

Ví dụ trong cây phân cấp gốc A dưới đây:



Cây đang xét (gốc A)



Cây con với gốc G của cây đang xét

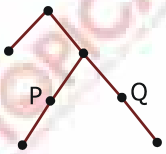
Ta có cha của C là B, con của G là H, I và J, anh em của H là I và J, tổ tiên của E là C, B và A, con cháu của B là C, D và E, các đỉnh trong là A, B, C, G, H và J, các lá (đỉnh ngoài) là D, E, F, I, K, L và M. Cây con với gốc G của cây đang xét được vẽ ở hình bên cạnh. Chiều cao của cây đang xét (gốc A) là 3, chiều cao của cây con gốc G của nó là 2.

7.6.2. Cây nhị phân

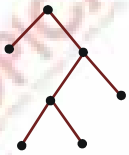
Một trường hợp riêng của cây phân cấp có tầm quan trọng đặc biệt trong các ứng dụng là cây nhị phân với định nghĩa dưới đây:

Định nghĩa. Một cây phân cấp được gọi là *nhị phân* nếu mọi đỉnh trong của nó có nhiều nhất là 2 con. Cây nhị phân được gọi là *đầy đủ* nếu mọi đỉnh trong của nó có đúng 2 con.

Ví dụ.



Cây T_1 (nhị phân không đầy đủ)



Cây T_2 (nhị phân đầy đủ)

Các cây T_1 và T_2 trong ví dụ này đều là những cây nhị phân (T_1 có 3 lá, T_2 có 4 lá), trong đó T_1 là không đầy đủ vì các đỉnh trong P, Q chỉ có 1 con, còn T_2 là đầy đủ vì mọi đỉnh trong đều có đúng 2 con.

Chú ý:

Khái niệm cây nhị phân được mở rộng một cách tự nhiên thành khái niệm cây *k-phân* trong đó nhị phân là trường hợp riêng với k = 2. Tuy nhiên do tầm quan trọng của cây nhị phân và thời lượng có hạn, nên trong bài này chỉ giới thiệu cây nhị phân. Trường hợp tổng quát *k-phân,* bạn đọc có thể tự mở rộng hoặc xem tham khảo [3].

Trong cây nhị phân, mỗi cây con thường được sắp theo thứ tự bên trái hoặc bên phải và được vẽ vào vị trí tương ứng so với đỉnh cha, chúng được gọi tương ứng là *cây con trái* hoặc *cây con phải* của đỉnh này (có thể có đỉnh chỉ có một cây con, hoặc chỉ có cây con trái, hoặc chỉ có cây con phải), chẳng hạn cây T_1 trong thí dụ trên, đỉnh P chỉ có cây con trái còn đỉnh Q chỉ có cây con phải. Cây T_2 là đầy đủ vì thế mọi đỉnh trong của nó đều có đủ hai cây con.

Một số tính chất:

Trong cây nhị phân đầy đủ, các tham số *n* (số đỉnh), *i* (số đỉnh trong), *l* (số lá) có một mối quan hệ chặt chẽ. Cụ thể nếu biết một trong ba tham số này, hai tham số còn lại sẽ được xác định duy nhất theo định lý dưới đây:

Định lý 1. Cây nhị phân đầy đủ với:

1) i đỉnh trong, có $n = 2i + 1$ đỉnh và $l = i + 1$ lá.

2) n đỉnh, có $i = \frac{n-1}{2}$ đỉnh trong và $l = \frac{n+1}{2}$ lá.

3) l lá, có $n = 2l - 1$ đỉnh và $i = l - 1$ đỉnh trong.

Chứng minh

1) Mỗi đỉnh trong của cây nhị phân đầy đủ có 2 đỉnh con, vậy số đỉnh con của i đỉnh trong là 2i. Các đỉnh con này là tất cả các đỉnh của cây đang xét trừ đỉnh gốc. Từ đó nhận được số đỉnh n của cây đang xét là 2i + 1. Theo định nghĩa, số lá l cộng với số đỉnh trong i bằng toàn bộ số đỉnh n, từ đó số lá $l = n - i = 2i + 1 - i = i + 1$.

2) Từ kết quả 1 suy ra $i = \frac{n-1}{2}$ và $l = n - i = n - \frac{n-1}{2} = 2$.

3) Từ kết quả 2 suy ra $n = 2l - 1$ và $i = n - l = 2l - 1 - l = l - 1$.

Nếu biết chiều cao h của cây nhị phân, ta có thể tính được giới hạn trên của số lá của nó qua nội dung định lý sau:

Định lý 2. Số lá của một cây nhị phân chiều cao h không vượt quá 2^h.

Định lý được chứng minh dễ dàng bằng quy nạp theo chiều cao h. Chúng tôi dành cho bạn đọc xem như bài tập.

7.6.3. Các thuật toán duyệt cây nhị phân

Có nhiều cách duyệt cây nhị phân, chúng khác nhau về thứ tự duyệt. Dưới đây trình bày ba cách duyệt cây nhị phân được dùng thường xuyên nhất vì các ứng dụng của chúng. Giả sử cần duyệt cây nhị phân với gốc r. Các thuật toán này đều được phát biểu đệ quy theo các cây con của cây nhị phân đang xét.

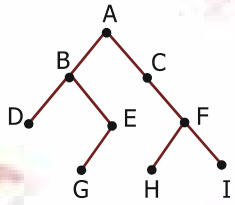
Duyệt theo thứ tự trước (Preorder)

1) Thăm gốc r.

2) Duyệt cây con bên trái r theo tiền thứ tự (nếu có).

3) Duyệt cây con bên phải r theo tiền thứ tự (nếu có).

Ví dụ. Xét cây nhị phân với gốc A:



Các bước duyệt tiền thứ tự như sau:

1. Thăm A.
2. Duyệt cây con B theo tiền thứ tự:
    2.1. Thăm B.
    2.2. Duyệt cây con D theo tiền thứ tự:
        2.2.1. Thăm D.
    2.3. Duyệt cây con E theo tiền thứ tự:
        2.3.1 Thăm E.
        2.3.2. Duyệt cây con G theo tiền thứ tự:
            2.3.2.1. Thăm G.
3. Duyệt cây con C theo tiền thứ tự:
    3.1. Thăm C.
    3.2. Duyệt cây con F theo tiền thứ tự:
        3.2.1. Thăm F.
        3.2.2. Duyệt cây con H theo tiền thứ tự:
            3.2.2.1. Thăm H.
        3.2.3. Duyệt cây con I theo tiền thứ tự:
            3.2.3.1. Thăm I.

Đến đây, việc duyệt kết thúc và thứ tự đi thăm các đỉnh theo tiền thứ tự là A, B, D, E, G, C, F, H, I.

Duyệt theo thứ tự giữa (Inorder)

1) Duyệt cây con bên trái r theo trung thứ tự (nếu có).

2) Thăm gốc r.

3) Duyệt cây con bên phải r theo trung thứ tự (nếu có).

Quay lại ví dụ trên, các bước duyệt trung thứ tự như sau:

1. Duyệt cây con B theo trung thứ tự.
    1.1. Duyệt cây con D theo trung thứ tự.
        1.1.1. Thăm D.
    1.2. Thăm B.
    1.3. Duyệt cây con E theo trung thứ tự.
        1.3.1. Duyệt cây con G theo trung thứ tự:
            1.3.1.1. Thăm G.
        1.3.2. Thăm E.
2. Thăm A.
3. Duyệt cây con C theo trung thứ tự:
    3.1. Thăm C.
    3.2. Duyệt cây con F theo trung thứ tự
        3.2.1. Duyệt cây con H theo trung thứ tự:
            3.2.1.1. Thăm H.
        3.2.2. Thăm F.
        3.2.3. Duyệt cây con I theo trung thứ tự:
            3.2.3.1. Thăm I.

Việc duyệt kết thúc và thứ tự đi thăm các đỉnh theo trung thứ tự là: D, B, G, E, A, C, H, F, I.

Duyệt theo thứ tự sau (Postorder)

1) Duyệt cây con bên trái r theo hậu thứ tự (nếu có).
2) Duyệt cây con bên phải r theo hậu thứ tự (nếu có).
3) Thăm gốc r.

Trở lại ví dụ ban đầu, các bước duyệt hậu thứ tự như sau:

1. Duyệt cây con B theo hậu thứ tự:
    1.1. Duyệt cây con D theo hậu thứ tự:
        1.1.1. Thăm D.
    1.2. Duyệt cây con E theo hậu thứ tự:
        1.2.1. Duyệt cây con G theo hậu thứ tự:
            1.2.1.1. Thăm G.
        1.2.2. Thăm E.
    1.3. Thăm B.
2. Duyệt cây con C theo hậu thứ tự:
    2.1. Duyệt cây con F theo hậu thứ tự:
        2.1.1. Duyệt cây con H theo hậu thứ tự:
            2.1.1.1. Thăm H.
        2.1.2. Duyệt cây con I theo hậu thứ tự
            2.1.2.1. Thăm I.
        2.1.3. Thăm F.
    2.2. Thăm C.
3. Thăm A.

Việc duyệt kết thúc và thứ tự đi thăm các đỉnh theo hậu thứ tự là: D, G, E, B, H, I, F, C, A.

Để minh họa, bạn đọc có thể cài đặt một chương trình (trên PASCAL hoặc C) cho phép duyệt cây nhị phân bằng một trong ba cách, trong đó xây dựng các thủ tục (hàm) đệ quy thực hiện các thuật toán đã nêu. Dưới đây là nội dung các thủ tục đó (mô phỏng trên PASCAL), bạn đọc có thể dựa vào để cụ thể hóa chương trình của mình (chú ý xem thêm việc lưu trữ cây nhị phân trong các giáo trình về cấu trúc dữ liệu và giải thuật), sau đó chạy lại ví dụ đã nêu.

```
PROCEDURE PREORDER(root: INTEGER);
{duyệt tiền thứ tự cây nhị phân có gốc là root, giả thiết
kiểu của đỉnh là INTEGER}
BEGIN
    IF (có root) THEN
    BEGIN
          (thăm root);
          PREORDER(con trái của root);
          PREORDER(con phải của root);
    END;
END;
```

```
PROCEDURE INORDER(root: INTEGER);
{duyệt trung thứ tự cây nhị phân có gốc là root, giả thiết
kiểu của đỉnh là INTEGER}
BEGIN
    IF (có root) THEN
    BEGIN
          INORDER(con trái của root);
          (thăm root);
          INORDER(con phải của root);
    END;
END;
```

```
PROCEDURE POSTORDER(root: INTEGER);
{duyệt hậu thứ tự cây nhị phân có gốc là root, giả thiết kiểu
của đỉnh là INTEGER}
BEGIN
    IF (có root) THEN
    BEGIN
          POSTORDER(con trái của root);
          POSTORDER(con phải của root);
          (thăm root);
    END;
END;
```

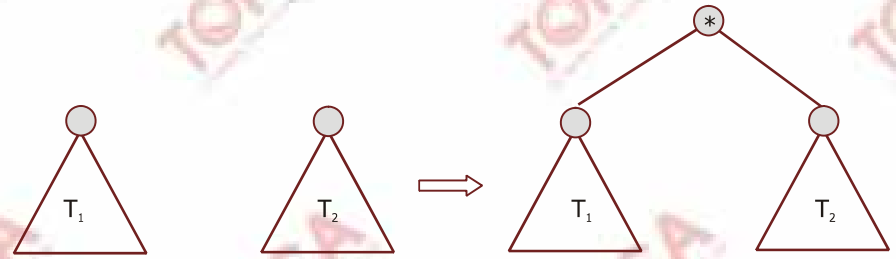
7.6.4. Một số ứng dụng của cây nhị phân

Cây nhị phân có rất nhiều ứng dụng, đặc biệt trong lĩnh vực tìm kiếm và sắp xếp dữ liệu. Những ứng dụng loại này được thảo luận khá kỹ trong các giáo trình cấu trúc dữ liệu và giải thuật (chẳng hạn, bạn đọc có thể tham khảo [4]). Ở đây, chúng tôi chỉ giới thiệu hai lĩnh vực ứng dụng của cây nhị phân trong việc tính toán biểu thức và mã hóa.

7.6.4.1. Cây biểu thức

Định nghĩa. *Cây biểu thức* là một cây nhị phân có các đỉnh trong là các toán tử (phép toán) còn các lá (đỉnh ngoài) là các giá trị (hằng, biến hoặc hàm) nhằm biểu diễn một biểu thức nào đó (biểu thức số, biểu thức lôgic, biểu thức tập hợp, ...). Các toán tử ở đây là những toán tử hai ngôi hay một ngôi.

Cây biểu thức được xây dựng từ một biểu thức theo cách đệ quy giống như định nghĩa một biểu thức. Đầu tiên, nếu biểu thức chỉ gồm một giá trị thì cây biểu thức của nó chỉ có một đỉnh (vừa là gốc, vừa là lá) biểu diễn giá trị này. Nếu E là biểu thức kết hợp hai biểu thức E_1 và E_2 (theo thứ tự) nhờ toán tử (2 ngôi) *, và T_1, T_2 tương ứng là những cây biểu diễn E_1, E_2 thì cây T biểu diễn E được xây dựng bằng cách thêm đỉnh * làm gốc, nối hai gốc của T_1 và T_2, xem như T_1 và T_2 tương ứng là các cây con trái và cây con phải của *, như hình vẽ dưới đây:



Cây biểu diễn E_1 **Cây biểu diễn E_2** **Cây biểu diễn biểu thức $E = E_1 * E_2$**

Tương tự nếu toán tử * là một ngôi thì gốc * chỉ có một con (được hiểu là trái hay phải cũng được).

Như vậy, để xây dựng một cây biểu thức, ta xây dựng từ dưới lên trên, bắt đầu từ biểu thức trong cùng nhất rồi mở rộng ra ngoài (dựa vào vị trí các dấu ngoặc).

Ví dụ 1. Xây dựng cây biểu thức số ((x + y) ^ 3) * ((x − 2)/4) với các ký hiệu + (cộng, hai ngôi), ^ (lũy thừa, hai ngôi), * (nhân, hai ngôi), − (trừ, hai ngôi), / (chia, hai ngôi).

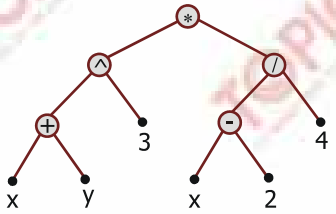
Giải. Đầu tiên xây dựng cây biểu diễn x + y và x − 2:

Cây biểu diễn x + y **Cây biểu diễn x − 2**

sau đó xây dựng cây biểu diễn (x + y) ^ 3 và (x – 2)/4

Cây biểu diễn (x + y) ^ 3 **Cây biểu diễn (x – 2)/4**

cuối cùng là cây biểu thức cần tìm:



Cây biểu thức số ((x+y)^3) * ((x−2)/4)

Ký pháp trung tố, tiền tố, hậu tố. Bây giờ duyệt cây biểu thức đã tìm theo trung thứ tự, ta được x + y ^ 3 * x − 2 / 4. Thứ tự này giống như thứ tự viết trong biểu thức đã cho ngoại trừ không có dấu ngoặc. Cách viết này được gọi là *ký pháp trung tố* của biểu thức đang xét. Chú ý rằng, do có nhiều cách đặt dấu ngoặc khác nhau nên có nhiều biểu thức khác nhau (tương ứng là những cây biểu thức khác nhau) có cùng một ký pháp trung tố, vì thế trong ký pháp này, ta cần phải tường minh các dấu ngoặc mỗi khi gặp một phép toán để có thể tính đúng biểu thức.

Với cách duyệt tiền thứ tự và hậu thứ tự cây biểu thức đã cho ta được tương ứng * ^ + x y 3 / − x 2 4 và x y + 3 ^ x 2 − 4 / *. Các cách viết này được gọi tương ứng là *ký pháp tiền tố* và *ký pháp hậu tố* của biểu thức đang xét. Ký pháp tiền tố còn được gọi là *ký pháp Ba lan* và ký pháp hậu tố còn được gọi là *ký pháp Ba lan ngược*. Khác với ký pháp trung tố, các ký pháp tiền tố và hậu tố hoàn toàn xác định biểu thức tương ứng mà không cần dấu ngoặc, vì thế chúng đặc biệt có lợi để tính biểu thức khi xây dựng các bộ dịch. Với cách viết hậu tố, máy đọc từ trái sang phải và dùng một ngăn xếp để lưu trữ các đại lượng như sau: mỗi khi đọc được một phép toán, máy thực hiện phép toán này với các đại lượng lấy ra từ đỉnh của ngăn xếp (tùy phép toán là một hoặc hai ngôi mà một hoặc hai đại lượng được lấy ra), kết quả tính được lại đặt vào đỉnh của ngăn xếp này. Cứ như vậy cho đến khi thực hiện xong phép toán cuối cùng. Với cách viết tiền tố, máy cũng làm tương tự nhưng đọc theo chiều từ phải sang trái. Chẳng hạn, quá trình tính biểu thức ở dạng hậu tố trong thí dụ trên:

$$E = x\ y + 3\ \hat{}\ x\ 2 - 4\ /\ *$$

được thực hiện trên ngăn xếp theo các bước mô tả trong hình vẽ dưới đây:

						2		4		
	y		3		x	x	x - 2	x - 2	(x - 2)/4	
x	x	x + y	x + y	(x+y)^3	(x+y)^3	(x+y)^3	(x+y)^3	(x+y)^3	(x+y)^3	E

Ví dụ 2. Hãy tìm cây nhị phân biểu diễn biểu thức lôgic:

$$\neg(p \wedge q) \rightarrow (r \vee \neg s)$$

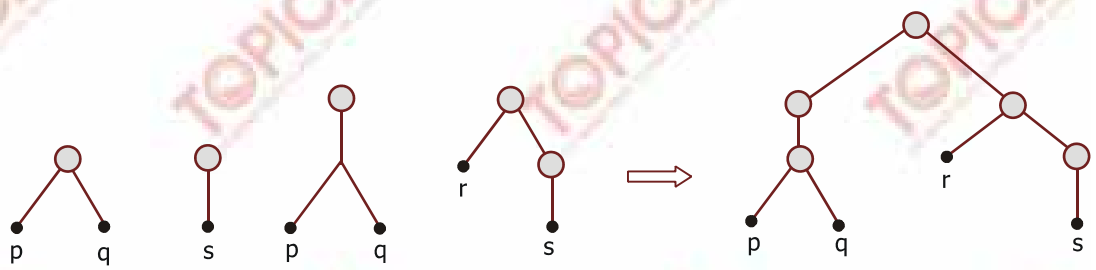
(chú ý: phép toán một ngôi luôn được thực hiện trước các phép toán hai ngôi nên không cần viết trong ngoặc), sau đó tìm dạng tiền tố, hậu tố và trung tố của biểu thức này.

Giải.

Lần lượt xây dựng các cây con thực hiện các biểu thức

$$p \wedge q,\ \neg s,\ \neg(p \wedge q),\ r \vee \neg s$$

cuối cùng ta nhận được cây biểu diễn biểu thức $\neg(p \wedge q) \rightarrow (r \vee \neg s)$



Các dạng tiền tố, hậu tố và trung tố của biểu thức đang xét là:

- Tiền tố $\rightarrow \neg \wedge p\ q \vee r \neg s$
- Hậu tố $p\ q \wedge \neg r\ s \neg \vee \rightarrow$
- Trung tố $\neg p \wedge q \rightarrow r \vee \neg s$

Chú ý

Đối với phép toán một ngôi, việc coi con của nó là trái hay phải không ảnh hưởng gì đến các dạng tiền tố và hậu tố, tuy nhiên đối với dạng trung tố có ảnh hưởng đôi chút: nếu coi là con bên trái thì dấu phép toán đi sau toán hạng, còn trái lại thì dấu phép toán đi trước toán hạng. Vì thế, để phù hợp với cách viết thông thường của biểu thức (dấu phép toán một ngôi luôn đi trước toán hạng), trong dạng trung tố, cây con của phép toán một ngôi luôn được xem là cây con bên phải.

7.6.4.2. Cây mã tiền tố và mã Huffman

Ta xét bài toán mã các ký hiệu của một tập ký hiệu nào đó bằng chuỗi nhị phân (chuỗi bit 0, 1). Thông thường, người ta chọn độ dài của mã là cố định đối với mọi ký hiệu sao cho hai ký hiệu khác nhau ứng với hai chuỗi mã khác nhau. Chẳng hạn để mã các chữ cái trong bảng chữ cái tiếng Anh (gồm 26 ký hiệu, không phân biệt chữ to, chữ nhỏ), ta có thể dùng độ dài là 5 cho mọi chuỗi mã (vì có $2^5 = 32$ chuỗi mã khác nhau, đủ để mã 26 ký hiệu). Với cách mã này, dựa vào bảng mã từng ký hiệu và độ dài cố định của một mã, ta dễ dàng giải mã một chuỗi bit (mã của bản tin nào đó) để nhận được bản tin ban đầu. Chẳng hạn để mã những bản tin thành lập từ tập 4 chữ cái {M, I, S, P}, ta có thể chọn độ dài cố định cho mã của mỗi chữ cái là 2 ($2^2 = 4$) với bảng mã như sau:

Chữ cái	Mã
M	00
I	01
S	10
P	11

và gửi đi chuỗi mã là 000110100110011101 (gồm 18 bit). Từ bảng mã và độ dài cố định là 2 cho mỗi mã, ta dễ dàng giải mã bản tin được gửi bằng cách tách thành từng nhóm 2 bit và tra bảng mã để nhận được bản tin ban đầu:

00 01 10 10 01 10 01 11 01

M I S S I S I P I

Bây giờ đặt lại vấn đề mã hóa mỗi ký hiệu không cùng độ dài, sao cho ký hiệu nào xuất hiện càng nhiều thì độ dài mã của nó càng ngắn, như thế sẽ tiết kiệm được số bit cần mã hóa. Chẳng hạn có thể dùng bảng mã như sau:

Chữ cái	Mã
M	00
I	0
S	1
P	01

bảng mã này vẫn thỏa mãn hai ký hiệu khác nhau có hai mã khác nhau. Tuy nhiên, một vấn đề phát sinh là việc gửi mã thì xác định còn việc giải mã là không xác định vì không biết dấu hiệu kết thúc của mỗi ký hiệu có trong chuỗi mã. Thí dụ, với bảng mã vừa nêu, dễ dàng mã hóa bản tin MISSISIPI để được chuỗi bit 00011010010 (gồm 11 bit) nhưng khi giải mã thì không xác định, chẳng hạn dưới đây là hai kết quả, tùy thuộc vào việc chọn vị trí kết thúc mã của từng ký hiệu:

0 00 1 1 01 00 1 0

I M S S P M S I

00 01 1 01 0 01 0

M P S P I P I

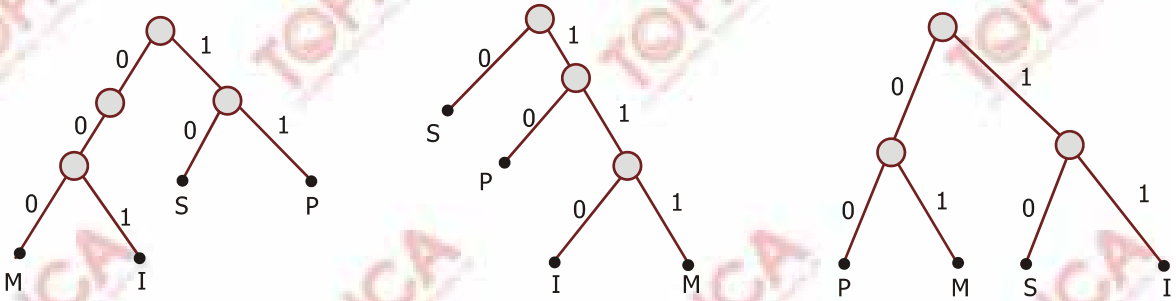
Sở dĩ xảy ra tình trạng này là vì trong bảng mã xuất hiện các ký hiệu mà mã của nó lại là phần đầu của mã của một ký hiệu khác, chẳng hạn mã chữ I là 0, mã chữ M là 00, mỗi khi gặp chuỗi 000, ta có thể hiểu là III hoặc IM hoặc MI, tất cả đều được. Để tránh điều này, bảng mã cần phải thỏa mãn thêm điều kiện: *không có một mã của một ký hiệu nào lại là phần đầu của mã của một ký hiệu khác*. Một mã nhị phân như vậy được gọi là một *mã tiền tố*. Chú ý rằng, bảng mã với độ dài chung cho mọi mã cũng là một mã tiền tố.

Để xây dựng một mã tiền tố, người ta dùng một cây nhị phân có các lá là các ký hiệu cần mã hóa, cạnh đi từ đỉnh xuống con trái của nó được gán nhãn 0 còn cạnh đi từ đỉnh xuống con phải của nó được gán nhãn 1. Khi đó dãy các nhãn đi từ gốc đến lá sẽ là mã tiền tố của ký hiệu tương ứng. Cây được xây dựng như vậy, được gọi là *cây mã tiền tố*.

Khi có cây mã tiền tố, việc giải mã là hoàn toàn xác định (mặc dù độ dài của mỗi mã không được biết trước). Đi theo đường đi được chỉ dẫn bởi dãy các bit trong chuỗi mã bắt đầu từ gốc trên cây đang xét, đến lá nào thì ký hiệu tương ứng được tìm thấy. Lặp lại thao tác trên với các bít tiếp theo ... cho đến khi hết chuỗi mã, ta nhận được bản tin ban đầu. Các đường đi từ gốc đến lá trên cây là xác định duy nhất theo các lá, điều này đảm bảo các ký hiệu tìm thấy là hoàn toàn xác định.

Có nhiều cây mã tiền tố khác nhau trên cùng một tập ký hiệu, điều này cũng có nghĩa, có nhiều mã tiền tố khác nhau trên cùng một tập ký hiệu. Các mã này cho các cách mã hóa khác nhau một bản tin, với tổng số bit của chuỗi mã (mà ta gọi là độ dài mã của bản tin đang xét) là khác nhau.

Ví dụ dưới đây là ba cây mã tiền tố của tập 4 chữ cái {M. I, S, P}, ứng với ba bộ mã khác nhau:



Bộ mã 1 **Bộ mã 2** **Bộ mã 3**

Ký hiệu	Bộ mã 1	Bộ mã 2	Bộ mã 3
M	000	111	01
I	001	110	11
S	10	0	10
P	11	10	00

Bây giờ mã hóa bản tin MISSISIPI bằng ba bộ mã trên, ta được các kết quả tương ứng:

- Bộ mã 1: 000001101000110001110 01 (23 bit)
- Bộ mã 2: 11111000110011010110 (20 bit)
- Bộ mã 3: 011110101110110011 (18 bit)

Bạn đọc có thể giải mã các chuỗi bit trên theo cách thức vừa nêu trên cây tương ứng để nhận được bản tin ban đầu.

Trong ví dụ trên, đối với bản tin đang xét, bộ mã 3 có độ dài ngắn hơn cả, tuy nhiên nó chưa phải là ngắn nhất vì trong mã này ta chưa để ý đến vấn đề ưu tiên độ dài ngắn hơn cho mã của ký hiệu có số lần xuất hiện trong bản tin nhiều hơn (I 4 lần, S 3 lần, M và P 1 lần). Bây giờ giữ nguyên cây tiền tố của bộ mã 2, ta đảo các chữ cái (nhãn các lá) sao cho I có mã ngắn nhất (mã 0), tiếp theo là S (mã 10) và cuối cùng là M (mã 111) và P (mã 110) để được bộ mã 4:

Ký hiệu	Bộ mã 4
M	111
I	0
S	10
P	110

Khi đó kết quả mã hóa bản tin MISSISIPI theo bộ mã 4 sẽ là 1110101001001100 (16 bit). Có thể thấy với bản tin đang xét, mã vừa tìm có độ dài là ngắn nhất. Mã tối ưu này có thể định nghĩa tổng quát như sau:

Xét một bản tin nào đó (một bản tin có thể xem là một chuỗi ký hiệu, trong đó có chứa các ký hiệu chữ cái, chữ số, các dấu chính tả, các dấu phép toán, các dấu ngăn cách, các dấu xuống dòng, và một số ký hiệu khác). Một mã tiền tố trên tập các ký hiệu xuất hiện trong bản tin (gọi là *mã của bản tin*) có độ dài của chuỗi mã bản tin (nghĩa là tổng số bit có trong chuỗi này) ngắn nhất (so với mọi mã của bản tin đang xét), gọi là

mã Huffman của bản tin này. Cây thực hiện mã Huffman của bản tin được gọi là *cây mã Huffman* của bản tin. Chẳng hạn với bản tin MISSISIPI vừa nêu, các bộ mã 1, 2, 3, 4 là các mã của bản tin này, trong đó bộ mã 4 là một mã Huffman. Chú ý có thể có nhiều mã Huffman (cây mã Huffman) của một bản tin, nhưng chúng cùng cho một độ dài chuỗi mã tối ưu.

Gọi S là tập các ký hiệu có trong bản tin. Với mỗi ký hiệu x của S, gọi $t(x)$ là số lần xuất hiện x trong bản tin và $d(x)$ là độ dài mã của x trong bộ mã đang xét. Khi đó độ dài của chuỗi mã bản tin được tính theo công thức:

$$\delta = \sum_{x \in S} t(x)d(x)$$

và việc tìm mã Huffman của bản tin đang xét là việc giải bài toán tối ưu

$$\delta = \sum_{x \in S} t(x)d(x) \to \min .$$

Thuật toán dưới đây (gọi là thuật toán Huffman) cho phép xây dựng cây mã Huffman của một bản tin cho trước với giả thiết với mỗi ký hiệu x, biết số lần xuất hiện $t(x)$ của nó trong bản tin.

Bỏ qua trường hợp tầm thường, ta giả thiết S gồm l ký hiệu với $l > 1$. Thuật toán Huffman gồm các bước sau:

- Bước 1: Xây dựng rừng T gồm l cây, mỗi cây chỉ gồm một đỉnh tương ứng với một ký hiệu x trong S và được gán nhãn $t(x)$.
- Bước 2: Chọn hai cây trong T có gốc với nhãn nhỏ nhất. Thêm một đỉnh mới với nhãn là tổng các nhãn của hai gốc cây vừa chọn. Nối đỉnh mới với hai gốc này bằng hai cạnh có nhãn 0, 1 để tạo thành một cây nhị phân. Lặp lại bước này chừng nào T chưa phải là cây. Nếu T là một cây thì kết thúc: T là cây mã Huffman cần tìm.

Tính đúng đắn của thuật toán Huffman được chứng minh bằng quy nạp theo số ký hiệu của tập A. Chúng tôi dành cho bạn đọc xem như bài tập.

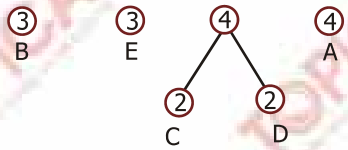
Chú ý:

Cây Huffman nhận được từ thuật toán trên cũng không duy nhất vì trong bước 2, có nhiều khả năng chọn hai cây con với gốc có nhãn nhỏ nhất.

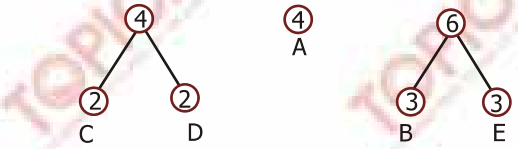
Để ví dụ ta lập cây mã Huffman của tập S gồm 5 chữ cái {A, B, C, D, E} với số lần xuất hiện của các chữ cái này trong bản tin là: A 4 lần, B 3 lần, C 2 lần, D 2 lần, E 3 lần. Các bước hình thành cây mã Huffman theo thuật toán đã nêu như sau (để thuận tiện cho việc vẽ, tại mỗi bước, ta luôn xếp lại gốc của các cây con từ trái sang phải theo chiều tăng dần của nhãn):

Bước 1:

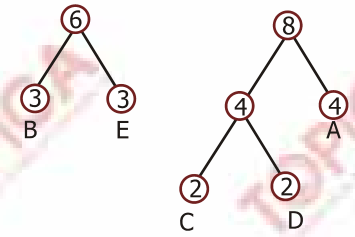
Bước 2 (lần 1):



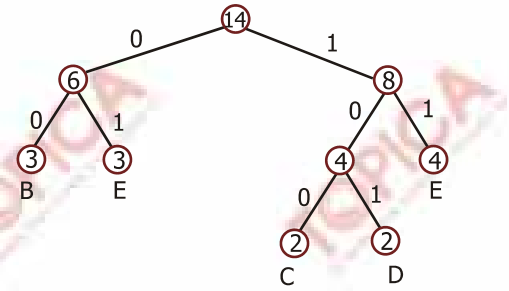
Bước 2 (lần 2):



Bước 2 (lần 3):



Bước 2 (lần 4, kết thúc, nhận được cây mã Huffman):



và ta nhận được bộ mã Huffman tương ứng:

x	mã	t(x)	d(x)	t(x) d(x)
A	11	4	2	8
B	00	3	2	6
C	100	2	3	6
D	101	2	3	6
E	01	3	2	6

với độ dài mã bản tin $\delta = \sum_{x \in S} t(x)d(x) = 32$ (bit) đạt được tối ưu (ngắn nhất).

Để so sánh, ta thử với mã ASCII của bản tin này (trong mã ASCII, mỗi ký hiệu được mã bằng 1 byte = 8 bit). Độ dài (số ký hiệu) của bản tin là 4 + 3 + 2 + 2 + 3 = 14, do đó độ dài mã ASCII của bản tin là 14 × 8 = 112 (bit). Bạn đọc có thể xây dựng thêm một số cây mã tiền tố khác của bản tin để so sánh.

Để cài đặt một chương trình minh họa thuật toán Huffman, bạn đọc có thể tham khảo [4], mục 12.2.

TÓM LƯỢC CUỐI BÀI

Sau khi học xong bài học này, các bạn cần ghi nhớ các vấn đề sau :

- Cây là một đồ thị vô hướng, liên thông, không có chu trình, mọi đồ thị vô hướng liên thông đều tìm được cây khung của nó.
- Cây thư mục, cây gia phả, cây hành chính là những ví dụ của cây.
- Phương pháp xây dựng cây khung của một đồ thị.
- Một số ứng dụng cụ thể của cây nhị phân như: cây biểu thức; cây mã tiền tố và mã Huffman.

BÀI TẬP

1. Có bao nhiêu cây không đẳng cấu với n đỉnh nếu

a) n = 3 b) n = 4 c) n = 5

2. Trong các đồ thị hai phía đầy đủ $K_{m, n}$ với m, n nguyên dương, đồ thị nào là cây?

3. Cho rừng G gồm p cây và n đỉnh. Tìm số cạnh của G.

4. Cho đồ thị vô hướng dưới dạng ma trận kề. Chỉ ra một thuật toán xác định xem đồ thị đã cho có phải là cây không? Minh họa thuật toán vừa tìm trong các thí dụ dưới đây (các ma trận chỉ cho một nửa vì đối xứng):

a)

0	1	1	1	0	0
	0	0	0	0	0
		0	0	0	0
			0	1	1
				0	0
					0

b)

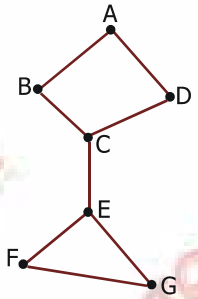
0	1	0	0	0	0
	0	1	0	1	0
		0	0	1	0
			0	0	1
				0	0
					0

c)

0	1	0	0	0	0
	0	1	0	1	0
		0	0	1	0
			0	0	1
				0	0
					0

5. Viết một chương trình (Pascal hoặc C) thực hiện bài 4, sau đó chạy lại các thí dụ đã nêu.

6. Một mạng máy tính bao gồm một số đường nối hai máy với nhau. Hai máy được xem như trao đổi được thông tin với nhau nếu chúng có một đường nối trực tiếp hoặc một đường truyền đi qua một số máy của mạng. Thời gian truyền tin là một đơn vị khi đi qua một cạnh nối. Tìm một thuật toán chọn một máy làm máy chủ và lược bỏ một số cạnh nối của mạng, sao cho từ máy chủ có duy nhất một đường truyền tin đến mỗi máy trong mạng và mọi máy của mạng nhận được tin với thời gian ngắn nhất. Minh họa thuật toán trên thí dụ sau:



(Một kết quả: máy chủ là C, các cạnh nối được lược bỏ là AD và FG, thời gian đạt được là 2).

7. Viết chương trình (trên Pascal hoặc C) thực hiện bài **6**. Thông tin của mạng được cho bằng ma trận kề.

8. Cho cây nhị phân đầy đủ với 1990 đỉnh trong. Hỏi nó có bao nhiêu cạnh?

9. Giả sử có N người tham gia vào một cuộc đấu cờ. Một kỳ thủ sẽ bị loại sau một trận thua và các trận đấu được tiến hành cho đến khi chỉ có một người thắng (người giữ chức vô địch).

Giả sử không có trận hòa (nếu hòa, người đi trước xem như thua). Hãy dùng cây lập mô hình cuộc thi đấu và xác định xem có bao nhiêu trận đấu xảy ra.

10. *Cây m-phân* là một cây phân cấp mà mọi đỉnh trong của nó có ít nhất m con. Nếu mọi đỉnh trong của cây m-phân đều có đúng m con thì nó được gọi là *cây m-phân đầy đủ*. Cây nhị phân là trường hợp riêng của cây m-phân với m = 2.

Hãy phát biểu và chứng minh các định lý tương tự như **định lý 1** và **định lý 2** (mục 7.6.2.) trong bài giảng.

11. Cây m-phân đầy đủ được gọi là một *cây m-phân hoàn chỉnh* nếu mọi lá của nó ở cùng một tầng. Hãy tìm số đỉnh và số lá của cây m phân hoàn chỉnh biết chiều cao của nó là *h*.

12. Xây dựng cây nhị phân biểu diễn các biểu thức dưới đây sau đó viết nó dưới dạng các ký pháp tiền tố, hậu tố, trung tố:

a) Biểu thức số ((x + 2) ^ 3) * (y − (3 + x)) − 5

b) Biểu thức tập hợp $(A \cap B) - (A \cup (B - A))$

c) Biểu thức lôgic $(\neg p \wedge (q \leftrightarrow \neg p)) \vee \neg q$

13. Viết lại biểu thức và tính giá trị của nó nếu biết ký pháp:

a) dạng tiền tố * / 9 3 + * 2 4 − 7 6

b) dạng hậu tố 3 2 * 2 ^ 5 3 − 8 4 / * −

14. Xây dựng cây và bộ mã Huffman của bản tin *this is an example of a huffman free* (kể cả dấu trắng) và xác định độ dài chuỗi mã của nó.

CÂU HỎI THƯỜNG GẶP

1. Cây là gì?

2. Đặc trưng của chu số?

3. Trong đồ thị vô hướng liên thông, khái niệm "chu số" là gì?

4. Có mấy cách duyệt cây nhị phân?