HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG MÔN

TOÁN RỜI RẠC 2

Giảng viên: TH.S. Phan Thị Hà

Điện thoại/E-mail: hathiphan@yahoo.com

Bộ môn: Công nghệ phần mềm

Học kỳ/Năm biên soạn: I/2009-2010

CHƯƠNG 6.CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ

- Mở đầu
- Các ứng dụng của cây
- Các phương pháp duyệt cây
- Cây khung của đồ thị
- Cây khung nhỏ nhất

MỞ ĐẦU

- **Định nghĩa 1**. Cây là một đồ thị vô hướng, liên thông và không có chu trình đơn.
- Vì cây không thể có chu trình đơn, nên cây không thể có cạnh bội và khuyên. Vậy mọi cây đều là ***đồ thị đơn***.

Định nghĩa 2. Một đồ thị không có chu trình đơn nhưng không liên thông được gọi là ***rừng***.

MỞ ĐẦU

- Cây thường được định nghĩa như một đồ thị vô hướng, trong đó ***giữa mọi cặp đỉnh*** của nó luôn tồn tại một đường đi đơn duy nhất. Định lý sau cho thấy trong đồ thị vô hướng thì tính ***liên thông và không có chu trình đơn*** tương đương với tính ***tồn tại đường đi đơn duy nhất*** giữa mọi cặp đỉnh

MỞ ĐẦU

- **Định lý 1**. Một đồ thị vô hướng là một cây nếu giữa mọi cặp đỉnh của nó luôn tồn tại đường đi đơn duy nhất.
- **Định nghĩa 3**. Cây có gốc được gọi là m-phân nếu tất cả các đỉnh trong của nó không có hơn m con. Cây được gọi là ***m-phân đúng (strict)*** nếu mọi đỉnh trong của nó có đúng m con. Cây m-phân với m = 2 được gọi là cây nhị phân.

  Cây có gốc được sắp (hay ***có thứ tự***) là cây có gốc trong đó các con của mỗi đỉnh được sắp xếp theo một thứ tự nhất định. Cây có gốc được sắp được vẽ sao cho các ***con của mỗi đỉnh trong được sắp thứ tự từ trái qua phải***. Trong cây nhị phân có thứ tự, các đỉnh trong có hai con, con thứ nhất gọi là ***con bên trái*** và con thứ hai là ***con bên phải***. Cây có gốc là con bên trái của một đỉnh được gọi là ***cây con bên trái***, tương tự ta có định nghĩa về ***cây con bên phải***.

MỞ ĐẦU
Những tính chất của cây

- **Định lý 2**. Cây với n đỉnh có đúng n-1 cạnh.
- **Định lý 3**. Cây m-phân đúng với i đỉnh trong sẽ có tất cả n = m*i + 1 đỉnh.
- **Định lý 3**. Cây m-phân đúng với
- a) Nếu cây có ***n đỉnh*** thì số đỉnh trong là i = (n-1)/m và số lá là l = ((m-1)n+1)/m
- b) Nếu cây có ***i đỉnh trong*** thì khi đó số đỉnh là n = m.i + 1 và số lá là l=(m-1)i+1.
- c) $n = \frac{ml-1}{m-1}$ và số đỉnh trong là $i = \frac{l-1}{m-1}$ đỉnh là :

MỞ ĐẦU

- **Định nghĩa**. ***Mức*** của đỉnh v trong cây có gốc là độ dài của đường đi duy nhất từ gốc tới nó. *Mức của gốc được định nghĩa bằng không*.
- ***Độ cao*** của cây là mức cao nhất của tất cả các đỉnh. Nói cách khác độ cao của cây có gốc là chiều dài của *đường đi dài nhất* từ gốc tới một đỉnh bất kỳ.
- Cây m-phân có gốc và độ cao h được gọi là ***cân đối*** nếu *tất cả các lá đều ở mức h hoặc h-1*.

- Cây AVL là cây nhị phân sao cho tại mỗi đỉnh chiều cao của cây con trái và cây con phải khác nhau không quá 1.
- Vậy cây AVL là trường hợp đặc biệt của cây nhị phân cân đối: cây AVL là cây nhị phân cân đối, nhưng ngược lại nói chung không đúng.
- ***Cây m-phân hoàn toàn(complete)*** là cây m-phân đúng trong đó mọi lá ở cùng một mức.

MỞ ĐẦU

❖**Định lý 5**. Có nhiều nhất m^h lá trong cây m-phân, độ cao h.

Hệ quả 1. Nếu cây m-phân có chiều cao h và có l lá, khi đó $h \geq \lceil \log_m l \rceil$. Nếu cây m-phân đúng và cân đối, khi đó h =

$\lceil \log_m l \rceil$. (Nhớ lại là $\lceil x \rceil$ là số nguyên nhỏ nhất không bé hơn x).

CÁC ỨNG DỤNG CỦA CÂY

Chúng ta sẽ nghiên cứu ba bài toán bằng mô hình cây.

- Các phần tử trong một danh sách được lưu trữ như thế nào để có thể dễ dàng định vị được chúng?
- Hãy xác định dãy các quyết định để tìm một đối tượng có tính chất nào đó trong tập hợp các đối tượng thuộc một loại nào đó.
- Cần phải mã hóa tập các chữ cái bằng các dãy nhị phân như thế nào để có hiệu quả nhất?

CÁC ỨNG DỤNG CỦA CÂY

Để giải quyết: tìm hiểu 3 loại cây sau

- ***Cây tìm kiếm nhị phân***
- ***Cây quyết định***
- **Các mã tiền tố**

CÁC ỨNG DỤNG CỦA CÂY

Cây tìm kiếm nhị phân

- Tìm kiếm một phần tử trong một danh sách là một trong những công việc quan trọng nhất trong tin học. Mục đích hàng đầu của chúng ta là đưa ra một thuật toán tìm kiếm có hiệu quả nhất là tìm một phần tử khi các phần tử được sắp xếp theo một thứ tự nào đó. Điều đó có thể thực hiện được bằng ***cây tìm kiếm nhị phân***. Đó là cây nhị phân trong đó mỗi đỉnh chỉ có nhiều nhất là 2 con: một con trái và một con phải. Mỗi đỉnh được gán một khóa. Các đỉnh được gán khóa sao cho khóa của đỉnh lớn hơn khóa của tất cả các đỉnh thuộc cây con bên trái, và nhỏ hơn khóa của tất cả các đỉnh thuộc cây con bên phải của nó.

CÁC ỨNG DỤNG CỦA CÂY

- Một cây nhị phân tìm kiếm chỉ có các đỉnh con bên trái sẽ tạo thành một cây lệch trái hay sắp xếp theo thứ tự giảm dần của khóa.
- Một cây nhị phân tìm kiếm chỉ có các đỉnh con bên phải sẽ tạo nên một cây lệch phải hay sắp xếp theo thứ tự tăng dần của khóa.

CÁC ỨNG DỤNG CỦA CÂY

- Nếu giá trị khóa của đỉnh x trùng với giá trị khóa tại đỉnh gốc thì không thể thêm node.
- Nếu giá trị khóa của đỉnh x nhỏ hơn giá trị khóa tại đỉnh gốc và chưa có lá con bên trái thì thực hiện thêm node vào nhánh bên trái.
- Nếu giá trị khóa của đỉnh x lớn hơn giá trị khóa tại đỉnh gốc và chưa có lá con bên phải thì thực hiện thêm node vào nhánh bên phải.

Thao tác tìm kiếm đỉnh trên cây nhị phân tìm kiếm:
Giả sử ta cần tìm kiếm khóa có giá trị x trên cây nhị phân tìm kiếm, trước hết ta bắt đầu từ gốc:

CÁC ỨNG DỤNG CỦA CÂY

- Nếu cây rỗng: phép tìm kiếm không thoả mãn;
- Nếu x trùng với khoá gốc: phép tìm kiếm thoả mãn;
- Nếu x nhỏ hơn khoá gốc thì tìm sang cây bên trái;
- Nếu x lớn hơn khoá gốc thì tìm sang cây bên phải;

CÁC ỨNG DỤNG CỦA CÂY

- **Thao tác loại bỏ đỉnh (Remove):** Việc loại bỏ đỉnh trên cây nhị phân tìm kiếm khá phức tạp. Vì sau khi loại bỏ ta phải điều chỉnh lại cây để nó vẫn là cây nhị phân tìm kiếm. Khi loại bỏ đỉnh trên cây nhị phân tìm kiếm thì đỉnh cần loại bỏ có thể ở một trong 3 trường hợp sau:

CÁC ỨNG DỤNG CỦA CÂY

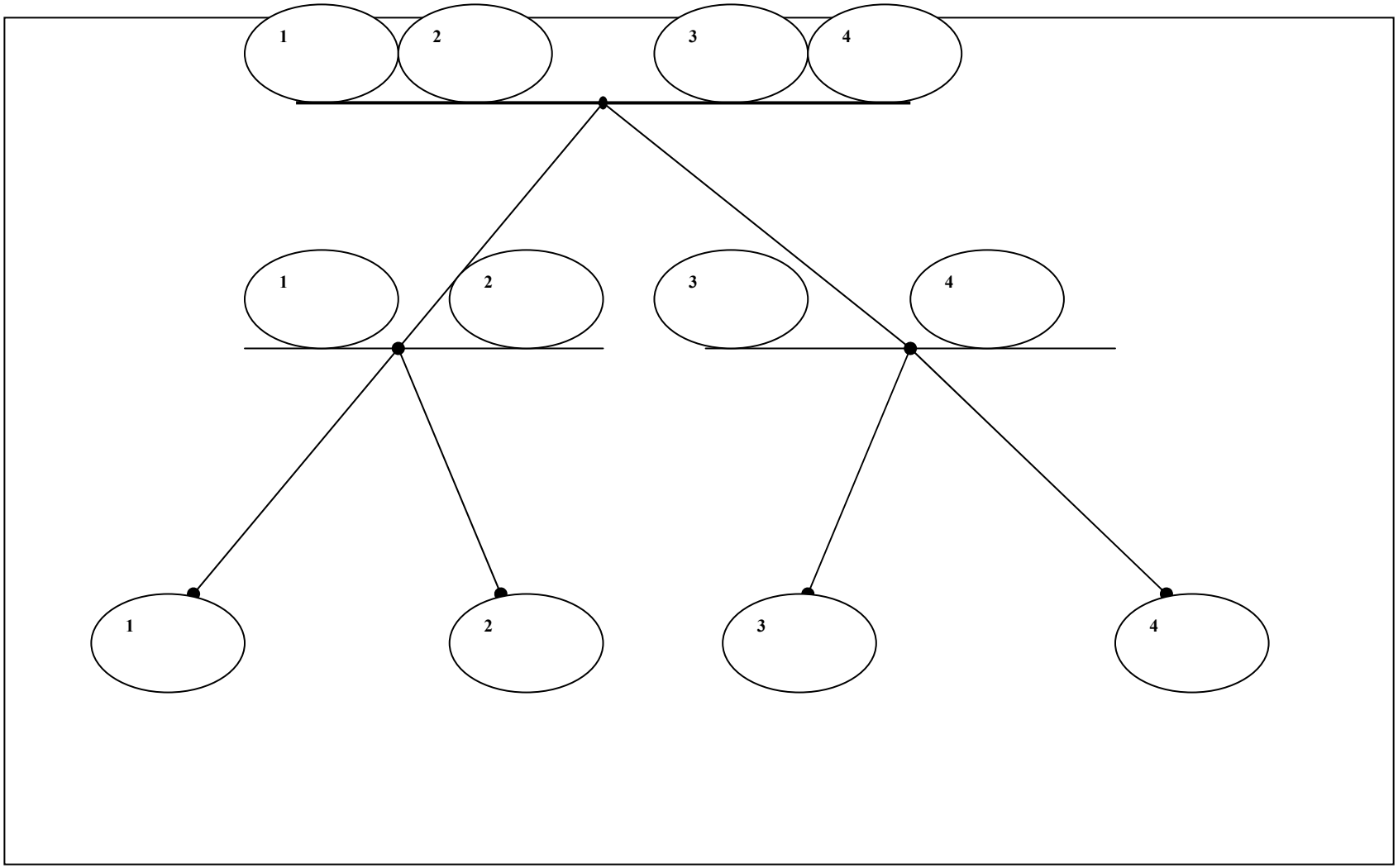
- Nếu đỉnh p cần loại là đỉnh treo thì việc loại bỏ được thực hiện ngay.
- Nếu node p cần xoá có một cây con thì ta phải lấy node con của node p thay thế cho p.
- Nếu đỉnh p cần xoá có nhieu cây con thì ta xét: Nếu đỉnh cần xoá ở phía cây con bên trái thì đỉnh bên trái nhất sẽ được chọn làm đỉnh thế mạng, nếu đỉnh cần xoá ở phía cây con bên phải thì đỉnh bên phải nhất sẽ được chọn làm node thế mạng.

CÁC ỨNG DỤNG CỦA CÂY

Cây quyết định

- Các cây có gốc có thể dùng để mô hình các bài toán trong đó có một dãy các quyết định dẫn đến lời giải. Chẳng hạn, cây tìm kiếm nhị phân có thể dùng để định vị các phần tử dựa trên một loạt các so sánh, trong đó mỗi so sánh cho ta biết ta có định vị được phần tử đó hay chưa, hoặc ta sẽ đi theo cây con trái hoặc cây con phải.
- Cây có gốc trong đó mỗi đỉnh trong ứng với một quyết định và mỗi cây con tại các đỉnh này ứng với mỗi một kết cục có thể của quyết định được gọi là ***cây quyết định***. Những lời giải có thể của bài toán tương ứng với các đường đi tới các lá của cây có gốc này. Ví dụ sau sẽ minh họa một ứng dụng của cây quyết định.

- **Ví dụ 1.** Có 4 đồng xu trong đó có 1 đồng xu giả nhẹ hơn đồng xu thật. Xác định số lần cân (thăng bằng) cần thiết để xác định đồng xu giả.
- **Giải.** Rõ ràng ta chỉ cần hai lần cân để xác định đồng xu giả vì khi ta đặt bốn đồng xu lên bàn cân thì chỉ có thể xảy ra hai kết cục: đồng số 1,2 nhẹ hơn hoặc nặng hơn đồng số 3, 4. Thực hiện quyết định cân lại giống như trên cho hai đồng xu nhẹ hơn ta xác định được đồng xu nào là giả. Hình 7.3 dưới đây sẽ mô tả cây quyết định giải quyết bài toán.


1
2
3
4
1
2
3
4
1
2
3
4

PHƯƠNG PHÁP DUYỆT CÂY

❖ Hệ địa chỉ thông dụng

1. Gán nhãn cho gốc bằng số 0. Sau đó k đỉnh con của nó (ở mức 1) từ trái sang phải được gán nhãn là 1, 2, 3, ..., k.

2. Với mọi đỉnh v ở mức n có nhãn là A, thì r đỉnh con của nó tính từ trái qua phải được gán nhãn là A.1, A.2,..., A.r.

Theo thủ tục này, đỉnh v ở mức n với $n \geq 1$, có nhãn là x1.x2.....xn, trong đó đường đi duy nhất từ gốc tới v sẽ đi qua đỉnh thứ x1 ở mức 1, đỉnh thứ x2 ở mức 2, ...

PHƯƠNG PHÁP DUYỆT CÂY

❖ ***Các phương pháp duyệt cây***

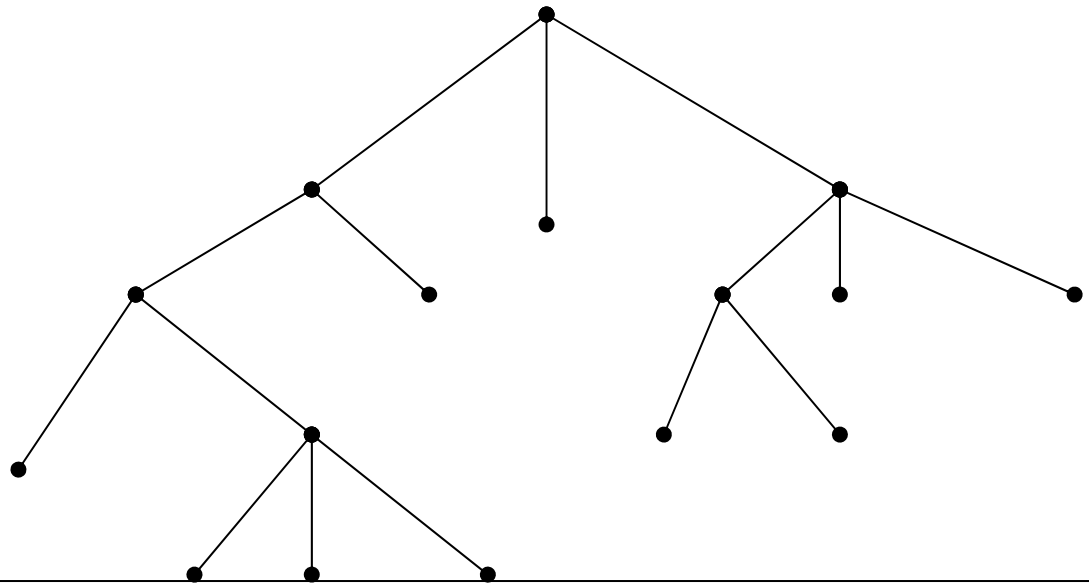
- Duyệt tiền thứ tự,
- Trung thứ tự
- Hậu thứ tự.

PHƯƠNG PHÁP DUYỆT CÂY

❖ **Định nghĩa 1**. Giả sử ta có cây T với gốc r. Nếu T chỉ có gốc thì r là cách duyệt tiền thứ tự của T. Nếu không thì gọi T1, T2, ..., Tn là các cây con tại r từ trái qua phải. Duyệt tiền thứ tự T sẽ viếng thăm r đầu tiên. Sau đó duyệt T1 theo kiểu tiền thứ tự, duyệt T2 theo kiểu tiền thứ tự, ... cứ như vậy cho đến khi Tn được duyệt theo kiểu tiền thứ tự

PHƯƠNG PHÁP DUYỆT CÂY

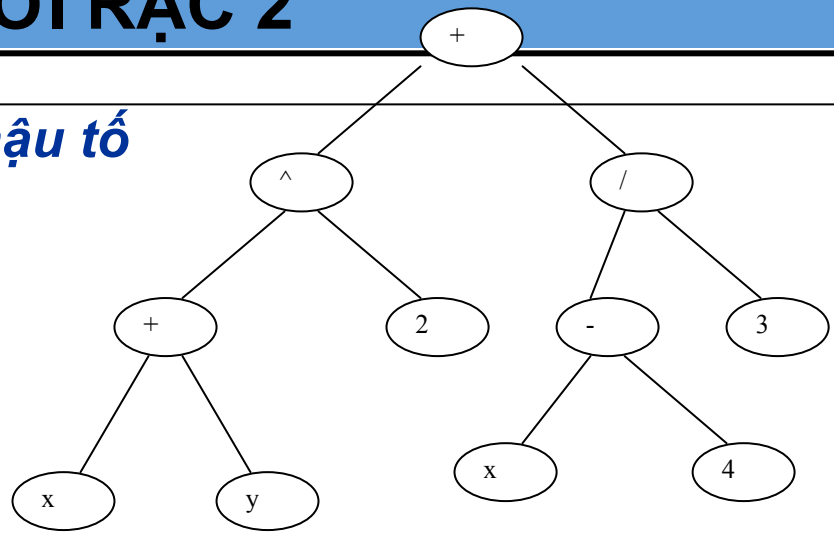
- Duyệt theo kiểu tiền thứ tự cây T là a,b,e,j,h,n,o,p,f,c,d,g,l,m,h,i.



- **Định nghĩa 2**. Giả sử ta có cây T với gốc r. Nếu T chỉ có gốc thì r là cách duyệt trung thứ tự của T. Nếu không thì gọi T1, T2, ..., Tn là các cây con tại r từ trái qua phải. Duyệt trung thứ tự T sẽ viếng thăm T1 đầu tiên, sau đó thăm r, rồi duyệt T2 theo kiểu trung thứ tự, ... cứ như vậy cho đến khi Tn được duyệt theo kiểu trung thứ tự.
- VD:j,e,n,k,o,p,b,f,a,c,l,g,m,d,h,i.

- **Định nghĩa 3**. Giả sử ta có cây T với gốc r. Nếu T chỉ có gốc thì r là cách duyệt hậu thứ tự của T. Nếu không thì gọi T1, T2, ..., Tn là các cây con tại r từ trái qua phải. Duyệt hậu thứ tự T sẽ bắt đầu duyệt T1 theo kiểu hậu thứ tự, rồi duyệt T2 theo kiểu hậu thứ tự, ... cứ như vậy cho đến khi Tn được duyệt theo kiểu hậu thứ tự, và cuối cùng kết thúc bằng việc viếng thăm r.
- j,n,o,p,k,e,f,b,c,l,m,g,h,i,d,a.

❖ ***Các ký pháp trung tố, tiền tố và hậu tố***



❖ Cách duyệt cây nhị phân biểu diễn biểu thức đã cho theo kiểu trung thứ tự sẽ tạo ra biểu thức có các số hạng và các phép toán theo đúng thứ tự như là đã có trong biểu thức ban đầu.

- Biểu thức có đầy đủ dấu ngoặc như vậy được gọi là ***dạng trung tố***.
- x y + 2 ^ x 4 - 3 / +.

  được ***dạng tiền tố*** của biểu thức khi ta duyệt cây theo kiểu tiền thứ tự. Các biểu thức được viết dưới dạng tiền tố được gọi là ***ký pháp Ba lan***
- + ^ + x y 2 / - x 4 3

PHƯƠNG PHÁP DUYỆT CÂY(tiếp)

- ***dạng hậu tố*** của một biểu thức bằng cách duyệt cây nhị phân theo kiểu hậu thứ tự. Biểu thức viết dưới dạng hậu tố được gọi là ***ký pháp Ba lan ngược***.
- x y + 2 ^ x 4 - 3 / +.

CÂY BAO TRÙM

- *Khái niệm*
- *Những thuật toán xây dựng cây bao trùm*

CÂY BAO TRÙM

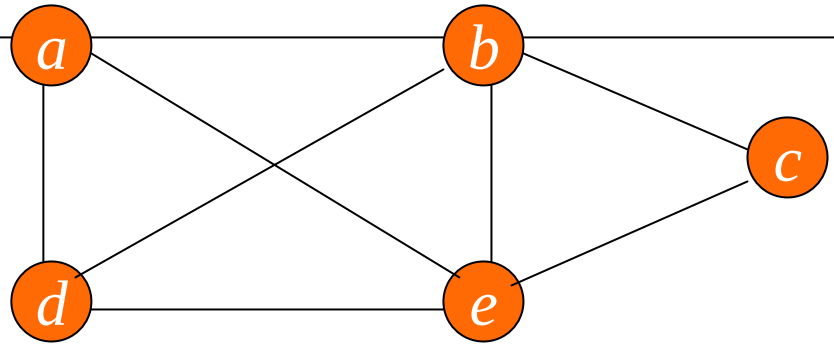
❖ ***Định nghĩa 11.2***

Giả sử G là một đồ thị vô hướng.

Cây T được gọi là cây bao trùm của đồ thị G nếu T là một đồ thị riêng của G.

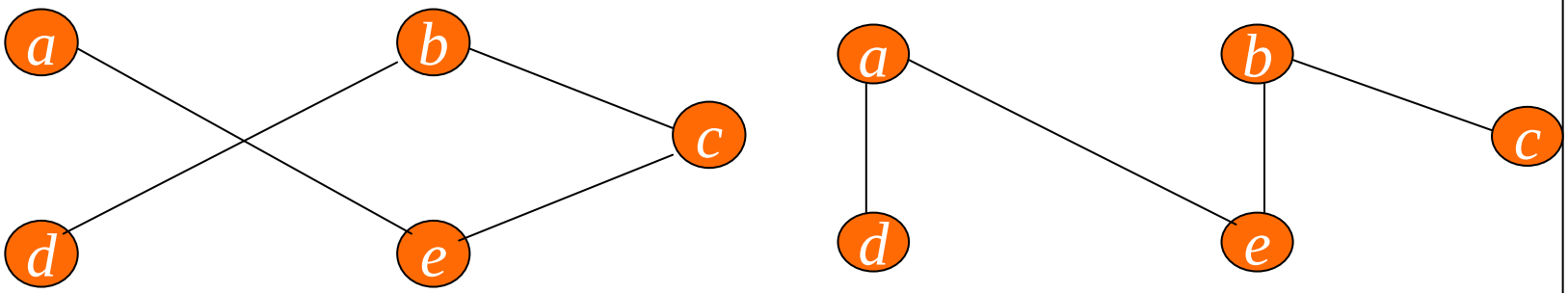
VÍ DỤ

Đồ thị G:



Một số cây bao tr

Hình 11.2. Đồ thị có cây bao trùm



Hình 11.3. Hai cây bao trùm của đồ thị G

CÂY BAO TRÙM (tiếp)

❖ ***Định lý 11.2***: Đồ thị vô hướng G có cây bao trùm $\Leftrightarrow$ G liên thông.

Chứng minh:

$\Rightarrow$: Hiển nhiên

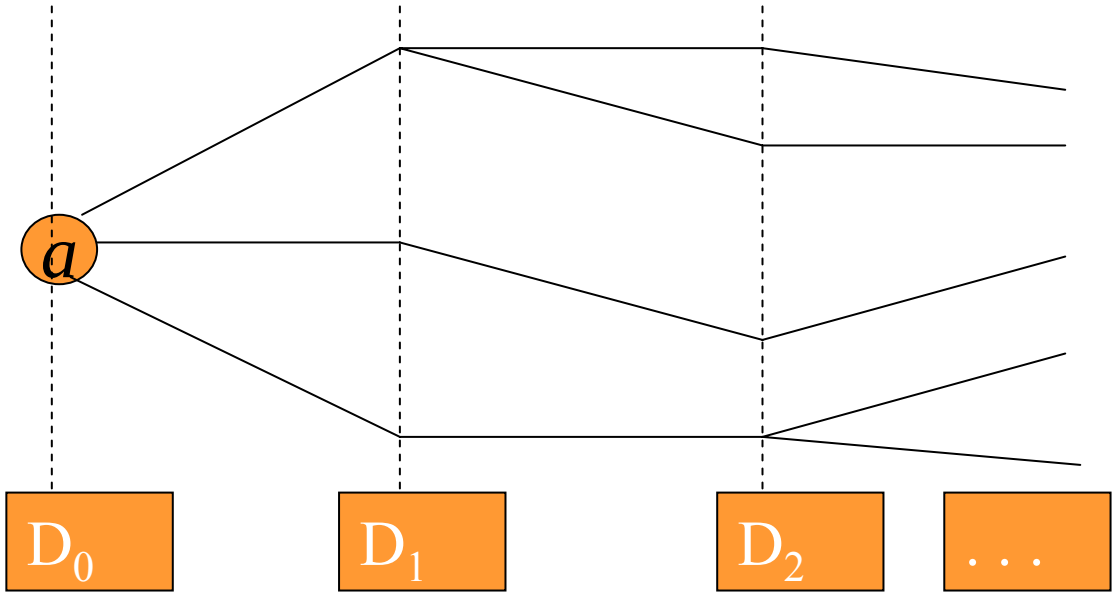
$\Leftarrow$: Chọn a là một đỉnh bất kỳ trong đồ thị G.

Ký hiệu: $d(x)$ là khoảng cách giữa a và đỉnh x.

Xây dựng các tập đỉnh:

$D_0 = \{a\}$, $D_i = \{ x \mid d(x) = i \}$ với $i \geq 1$.

CÂY BAO TRÙM (tiếp)



Hình 11.4. Cách xây dựng cây bao trùm

CÂY BAO TRÙM (tiếp)

Chứng minh: Lập tập cạnh T như sau:

Với mỗi đỉnh x của đồ thị G, $x \in D_i$ với $i \geq 1$, ta lấy cạnh nào đó nối x với một đỉnh trong D_{i-1}.

Tập cạnh này sẽ tạo nên một đồ thị riêng của G với n đỉnh và n - 1 cạnh.

Đồ thị riêng này liên thông vì mỗi đỉnh đều được nối với đỉnh a. Theo tính chất 3) của cây thì T là một cây.

Do vậy, T là cây bao trùm của đồ thị G. □

CÂY BAO TRÙM (tiếp)

Định lý 11.3 (*Borchardt):* Số cây bao trùm của một đồ thị vô hướng đầy đủ n đỉnh là n^{n-2}.

❖ Một số thuật toán tìm cây bao trùm:

- Thuật toán sử dụng phương pháp duyệt theo chiều sâu.
- Thuật toán sử dụng phương pháp duyệt theo chiều rộng.

THUẬT TOÁN TÌM CÂY BAO TRÙM

❖ ***Thuật toán 11.1*** (Tìm cây bao trùm bằng phương pháp duyệt theo chiều sâu)

Dữ liệu: Biểu diễn mảng DK các danh sách kề của đồ thị vô hướng G.

Kết quả: Cây bao trùm (*V, T*) của đồ thị G.

THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

Thuật toán

```
1  void  CBT_S (v) ;
2   {
3     Duyet [v]  :=  true ;
4     for   u ∈ DK[v]  do
5        if   ! (Duyet [u] )
6          {T  :=  T ∪ {(v, u)} ;  CBT_S (u) }
7   } ;
```

THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

```
8   main()          { Chương trình chính }
9{    for   u ∈ V   do   Duyet [u]  :=  false ;
10    T  :=  ∅ ;
11   CBT_S (z) ;              { z là đỉnh tuỳ ý của đồ thị,
                                 sẽ trở thành gốc của cây }
12  }
```

THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

❖ Tính đúng đắn của thuật toán: :

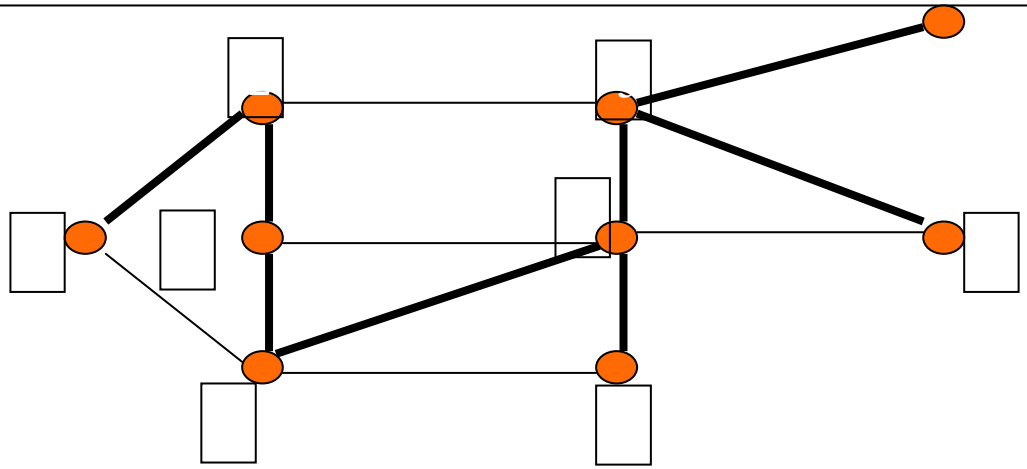
1. Khi ta thêm cạnh (*v, u*) vào tập cạnh T thì trong đồ thị (*V*, T) đã có đường đi từ *z* tới *v*. Vậy thuật toán xây dựng lên đồ thị liên thông.
2. Mỗi cạnh mới (*v, u*) được thêm vào tập T có đỉnh *v* đã được duyệt và đỉnh *u* đang duyệt. Vậy đồ thị đang được xây dựng không có chu trình.
3. Theo tính chất của phép duyệt theo chiều sâu, thủ tục CBT_S thăm tất cả các đỉnh của đồ thị liên thông G.

THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

- Do vậy, đồ thị do thuật toán xây dựng là một cây bao trùm của đồ thị đã cho.
- Độ phức tạp của thật toán là: O(n+m).

VÍ DỤ

Áp dụng thuật toán trên cho đồ thị (*nét mảnh*) ta nhận được cây bao trùm (*nét đậm*) như sau:



Hình 11.5. Cây bao trùm của đồ thị tìm theo phương pháp duyệt theo chiều sâu

THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

- ***Thuật toán 11.2*** (Tìm cây bao trùm bằng phương pháp duyệt theo chiều rộng)

  Dữ liệu: Biểu diễn mảng DK các danh sách kề của đồ thị vô hướng G.

  Kết quả: Cây bao trùm (*V, T*) của đồ thị G.

11.3. THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

Thuật toán

```
1  BEGIN
2    for  u ∈ V  do  Duyet [u]  :=  false ;
3    T  :=  ∅ ;
4    Q  :=  ∅ ;
5     pus_queue  z  into  Q ;     { z là đỉnh tuỳ ý của
                                   đồ thị và là gốc của cây }
6    Duyet [z]  :=  true ;
7      while  Q ≠ ∅  do
8        begin  pop_qưeue  v  from  Q ;
9              for  u ∈ DK[v]  do
10               if  ! Duyet [u]  then
11                 begin in-queue u into Q ;
12                       Duyet [u]  :=  true ;
13                       T  :=  T ∪ {(v, u)}  end
14       end
15     END.
```

TOÁN RỜI RẠC 2

11.3.THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

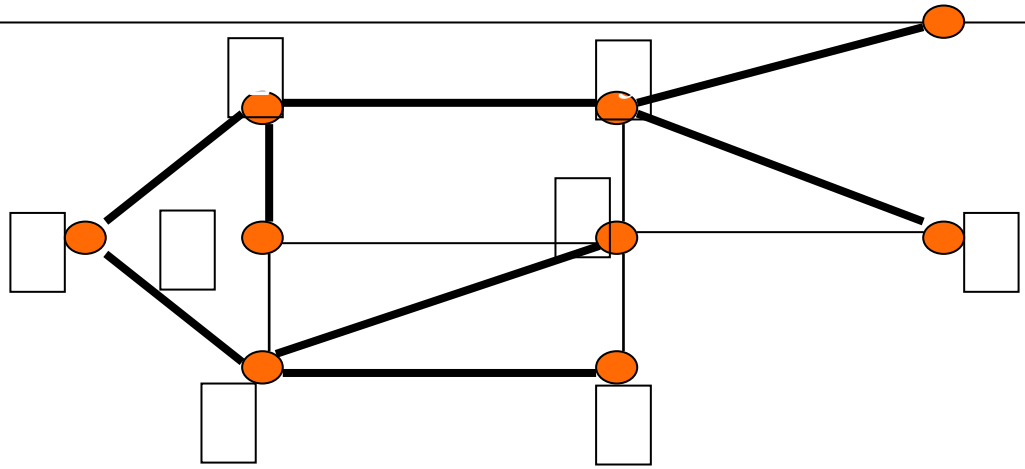
GIẢNG VIÊN: TH.S.PHAN THỊ HÀ

BỘ MÔN: CÔNG NGHỆ PHẦN MỀM

THUẬT TOÁN TÌM CÂY BAO TRÙM (tiếp)

❖ Độ phức tạp của thuật toán: O(*m*+*n*).

Ví dụ: Áp dụng thuật toán trên cho đồ thị (*nét mảnh*) ta nhận được cây bao trùm (*nét đậm*) như sau:



Hình 11.6. Cây bao trùm của đồ thị tìm theo phương pháp duyệt theo chiều rộng

CÂY BAO TRÙM NHỎ NHẤT

KN cây bao trùm

Thuật toán tìm cây bao trùm nhỏ nhất

CÂY BAO TRÙM NHỎ NHẤT

- *Bài toán*: Cho đồ thị vô hướng G liên thông với tập cạnh *E* và hàm trọng số *c* : *E* → N. Tìm cây bao trùm T của G sao cho tổng trọng số của các cạnh của T đạt giá trị nhỏ nhất.

- Một số thuật toán tìm cây bao trùm nhỏ nhất:
  - Thuật toán Kruskal
  - Thuật toán Prim

THUẬT TOÁN KRUSKAL

Thuật toán:

1. Chọn cạnh có trọng số bé nhất, ký hiệu là e_1 và đặt $W := \{e_1\}$.
2. Giả sử đã chọn được $W = \{e_1, e_2, \dots, e_i\}$. Chọn e_{i+1} là cạnh có trọng số bé nhất trong số các cạnh còn lại trong $E \setminus W$ sao cho $\{e_1, e_2, \dots, e_i, e_{i+1}\}$ không chứa chu trình.
3. Bổ sung: $W := W \cup \{e_{i+1}\}$.
4. Lặp lại các bước 2. – 3. chừng nào còn có thể.

THUẬT TOÁN KRUSKAL (tiếp)

- ***Định lý 11.4 :*** Tập các cạnh W tìm được theo thuật toán Kruskal tạo nên cây bao trùm nhỏ nhất của đồ thị G.

- ***Thuật toán Kruskal chi tiết***

```
1  procedure  Kruskal ;
2    begin
3      W  :=  ∅ ;   Z  :=  E ;
```

THUẬT TOÁN KRUSKAL (tiếp)

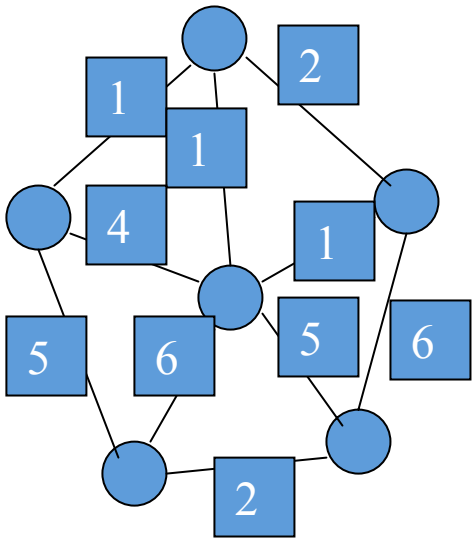
4 ***Thuật toán Kruskal chi tiết***

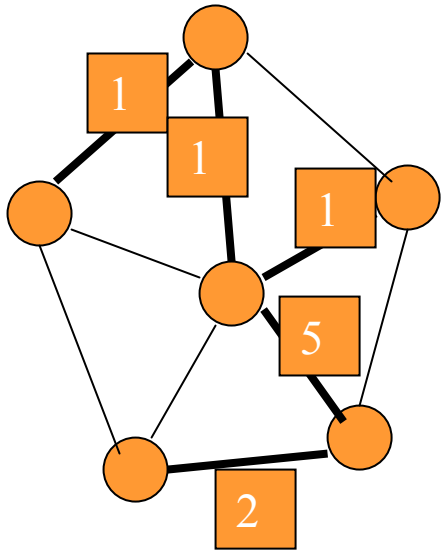
```
1  procedure  Kruskal ;
2    begin
3      W  :=  ∅ ;   Z  :=  E ;


4 while   (|W| < n -1) and  (Z  ≠ ∅)  do
5       begin
6          chọn cạnh  e  có trọng số bé nhất trong  Z ;
7          Z  :=  Z \ {e} ;
8     if   W ∪ {e}  không chứa chu trình   then
        1     W  :=  W ∪ {e}
9       end ;
10  if   |W| < n -1   then  writeln(″Đồ thị không liên
                                                thông″)
11  end ;
```

VÍ DỤ

Đồ thị có trọng số và cây bao trùm nhỏ nhất:





THUẬT TOÁN PRIM

❖ ***Thuật toán Prim***

Prim đã cải tiến thuật toán Kruskal như sau: ở mỗi vòng lặp ta chọn cạnh có trọng số bé nhất trong số các cạnh kề với các cạnh đã chọn mà không tạo nên chu trình.

THUẬT TOÁN PRIM (tiếp)

Thuật toán Prim được gọi là phương pháp lân cận gần nhất: bắt đầu từ một đỉnh nào đó *a* của đồ thị G ta nối nó với đỉnh "gần" nhất, chẳng hạn *b*. Nghĩa là, cạnh (*a, b*) được chọn có trọng số bé nhất. Tiếp theo, trong số các cạnh kề với đỉnh *a* hoặc đỉnh *b* ta chọn cạnh có trọng số bé nhất mà không tạo nên chu trình với cạnh (*a, b*). Cạnh này dẫn đến đỉnh thứ ba *c* ...

Tiếp tục quá trình này cho đến khi nhận được cây gồm *n* đỉnh và *n*-1 cạnh. Đó chính là cây bao trùm nhỏ nhất.

THUẬT TOÁN PRIM (tiếp)

```
1  procedure  Prim ;
2    begin
3      W  :=  {cạnh có trọng số bé nhất };
4       for   i  := 1  to  n - 2  do
5        begin
6          e  :=  cạnh có trọng số bé nhất kề với cạnh
                  trong  W và nếu ghép nó vào  W thì không
                  tạo nên chu trình ;
7          W  :=  W ∪ {e}
8        end
9    end ;
```

CÂY BAO TRÙM NHỎ NHẤT (tiếp)

❖ ***Định lý 11.5***

Trong đồ thị vô hướng có trọng số đôi một khác nhau, cây bao trùm nhỏ nhất tồn tại và duy nhất.

Chứng minh:

Vì trong vòng lặp chỉ có duy nhất một cạnh được chọn. □

CÂY BAO TRÙM LỚN NHẤT

- Trong các thuật toán Kruskal và Prim ta không ràng buộc về dấu của trọng số, nên có thể áp dụng cho đồ thị vô hướng với trọng số trên các cạnh có cùng dấu tuỳ ý.

CÂY BAO TRÙM LỚN NHẤT (tiếp)

❖ Để tìm cây bao trùm lớn nhất ta có hai cách:

1. Đổi thành dấu - cho các trọng số trên các cạnh. áp dụng một trong hai thuật toán đã trình bày ở trên để tìm cây bao trùm nhỏ nhất. Sau đó đổi dấu + trở lại, ta sẽ được cây bao trùm lớn nhất.
2. Sửa đổi trong các thuật toán: bước "*chọn cạnh có trọng số bé nhất ...* " được thay bằng "*chọn cạnh có trọng số lớn nhất ...* " còn các bước khác thì giữ nguyên. Khi thuật toán kết thúc, ta sẽ nhận được cây bao trùm lớn nhất.