CHƯƠNG 14

CÁC CẤU TRÚC DỮ LIỆU ĐA CHIỀU

Từ trước tới nay chúng ta mới chỉ nghiên cứu các CTDL để biểu diễn tập dữ liệu, trong đó dữ liệu được hoàn toàn xác định bởi một thuộc tính được gọi là khó của dữ liệu, và khó của dữ liệu được sử dụng trong các phép toán tìm kiếm, xen, loại. Chúng ta sẽ nói đến các dữ liệu được xác định chỉ bởi một thuộc tính khó như là các dữ liệu một chiều. Tuy nhiên trong rất nhiều lĩnh vực áp dụng, chẳng hạn như đồ hoạ máy tính, xử lý ảnh, các hệ thông tin địa lý, các hệ cơ sở dữ liệu đa phương tiện, các áp dụng của hình học tính toán …, chúng ta cần phải làm việc với các dữ liệu không phải một chiều. Đó là các dữ liệu hình ảnh (image data), dữ liệu video, dữ liệu audio, dữ liệu văn bản (document data), dữ liệu viết tay (handwritten data)… Các dữ liệu này có thể được biểu diễn bởi, chẳng hạn, các thuộc tính không gian, thời gian. Nói chung, các dữ liệu này được biểu diễn bởi vectơ các giá trị thuộc tính $(x_1,\ldots, x_k)$, tức là mỗi dữ liệu được mô tả bởi một điểm trong không gian k - chiều. Các dữ liệu này được gọi là các **dữ liệu điểm k chiều (k – dimensional point data)**. Trong chương này, chúng ta sẽ nghiên cứu các CTDL để biểu diễn tập dữ liệu điểm k - chiều. Các CTDL này được gọi là các **CTDL đa chiều (multidimensional data structures**) hay còn được gọi là các **CTDL không gian (spatial data structures**). Kỹ thuật chung được sử dụng là biểu diễn tập dữ liệu điểm k - chiều bởi các loại cây khác nhau. Cây biểu diễn sự phân hoạch không gian thành các miền con. Gốc cây biểu diễn toàn bộ miền chứa dữ liệu. Mỗi đỉnh của cây biểu diễn mụt miền nào đó, các con của nó biểu diễn sự phân hoạch miền này thành các miền con. Nhiều CTDL đã được đề xuất thực hiện ý tưởng phân hoạch miền chứa dữ liệu thành các miền con sao cho việc tìm kiếm dữ liệu và cập nhật tập dữ liệu được thực hiện hiệu quả. Chúng ta sẽ lần lượt nghiên cứu các CTDL: **cây k - chiều (k – dimensional tree), cây tứ phân (quadtree), cây tứ phân MX (MX – Quadtree**). Các loại cây này khác nhau ở chiến lược phân hoạch một miền thành các miền con.

14.1 CÁC PHÉP TOÁN TRÊN CÁC DỮ LIỆU ĐA CHIỀU

Dữ liệu điểm k - chiều là dữ liệu được hoàn toàn xác định bởi vectơ các giá trị thuộc tính ($x_1, \ldots, x_k$) trong không gian k - chiều, hay nói cách khác dữ liệu được hoàn toàn xác định bởi k giá trị khó $x_1, \ldots, x_k$. Trong các ứng dụng, khi có một tập dữ liệu điểm k - chiều, trong quá trình xử lý dữ liệu chúng ta thường xuyên phải sử dụng **các phép toán từ điển: tìm kiếm, xen, loại,** giống như đối với các dữ liệu một chiều (một khó). Chỉ có điều khác là ở đây chúng ta cần tiến hành tìm kiếm, xen, loại dựa vào k giá trị khó đã cho. Chẳng hạn, phép tìm kiếm ở đây có nghĩa là: tìm trong tập dữ liệu điểm k - chiều đã cho một dữ liệu khi biết vectơ các giá trị khó của nó là ($x_1, \ldots, x_k$).

Ngoài các phép toán từ điển, trên các dữ liệu đa chiều, chúng ta còn cần đến một phép toán đặc biệt: **tìm kiếm phạm vi (Range Search**). Phép toán tìm kiếm phạm vi được phát biểu như sau: cho trước một điểm dữ liệu ($x_1, \ldots, x_k$) và một số thực dương r, cần tìm trong tập dữ liệu điểm k - chiều đã cho tất cả các điểm dữ liệu cách ($x_1, \ldots, x_k$) một khoảng cách không lớn hơn r. Trong không gian 2 - chiều, phép toán tìm kiếm phạm vi, nói theo ngôn ngữ hình học có nghĩa là, cho trước hình tròn tâm (x, y) bán kính r, cần tìm tất cả các điểm dữ liệu nằm trong hình tròn này. Sau đây chúng ta đưa ra một ví dụ minh hoạ tầm quan trọng của phép toán tìm kiếm phạm vi.

Giả sử D là một tập các văn bản. Trước hết chúng ta cần đưa ra một cách biểu diễn văn bản. Giả sử T là danh sách các từ "quan trọng" xuất hiện trong các văn bản của tập D, T = ($t_1, \ldots, t_k$) .Với mỗi văn bản d thuộc D, ta biểu diễn d bởi vectơ các số thực không âm, d = ($x_1, \ldots, x_k$) trong đó x_i (i = 1, …, k) là tỷ số giữa số lần xuất hiện từ t_i trong văn bản d trên số từ trong văn bản d. Chúng ta cần xác định độ đo "sự liên quan" hay độ đo "sự tương tự", hay còn gọi là "khoảng cách" giữa hai văn bản. Có nhiều cách xác định độ đo đó, chúng ta không nêu ra ở đây. Trong các hệ tìm kiếm thông tin, người sử dụng mong muốn tìm ra một danh sách p văn bản "liên quan" nhiều nhất tới một câu hỏi Q từ hệ cơ sở dữ liệu văn bản D. Chúng ta quan niệm câu hỏi Q như một văn bản và biểu diễn nó bởi một vectơ Q = ($q_1, \ldots, q_k$) theo cách biểu diễn đã nêu. Vấn đề tìm câu trả lời cho câu hỏi Q bây giờ được quy về tìm p láng giềng gần nhất với Q (theo độ đo tương tự đã xác định), tức là được quy về thực hiện phép toán tìm kiếm phạm vi.

14.2 CÂY K - CHIỀU

Cây 2 - chiều được sử dụng để lưu giữ các dữ liệu điểm 2 - chiều, cây 3 - chiều để lưu giữ các dữ liệu điểm 3 - chiều, … Tổng quát, cây k - chiều để biểu diễn tập dữ liệu điểm k - chiều. Trong mục này, trước hết chúng ta sẽ trình bày CTDL cây 2 - chiều và chỉ ra các phép toán từ điển và phép toán tìm kiếm phạm vị được thực hiện như thế nào trên cây 2 - chiều. Sau đó chúng ta sẽ trình bày cách tổng quát hoá cây 2 - chiều để có cây k - chiều.

14.2.1 Cây 2 - chiều

Cây 2 - chiều là cây nhị phân. Mỗi dữ liệu điểm 2 - chiều gồm các giá trị toạ độ của điểm và các thông tin khác gắn với điểm này mà ta quan tâm. Vì vậy, mỗi đỉnh của cây 2 - chiều là một cấu trúc có dạng sau:

```
struct   Node
{
    infoType  info ;
    double  Xval ;
    double  Yval ;
    Node*  left ;
    Node*  right ;
} ;
```

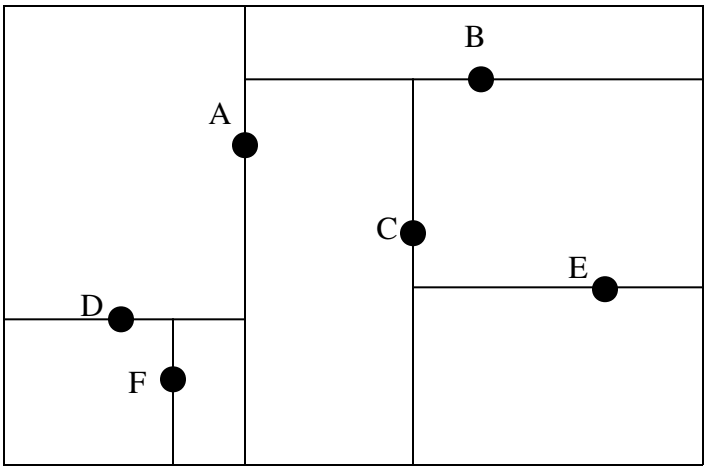
Trong đó, các trường Xval và Yval ký hiệu hoành độ và tung độ của điểm, các con trỏ left và right trỏ tới đỉnh con trái và phải, còn trường info lưu các thông tin khác về điểm. Nội dung của trường info được xác định cụ thể tuỳ từng ứng dụng. Chẳng hạn trong các hệ thông tin địa lý (geographic information system), các dữ liệu điểm được lưu giữ có thể là các điểm biểu diễn các vị trí mà chúng ta quan tâm trên bản đồ một vùng lãnh thổ nào đó. Nếu các vị trí là các thành phố thì thông tin về vị trí có thể là tên thành phố, số dân của thành phố, …

Cây 2 - chiều là sự tổng quát hoá của cây tìm kiếm nhị phân. Mỗi đỉnh trong của cây biểu diễn một miền và phân hoạch miền dữ liệu này theo một chiều (theo một toạ độ). Các đỉnh trên cùng một mức sẽ phân hoạch các miền dữ liệu tương ứng theo đường thẳng đứng (theo hoành độ) hoặc theo đường nằm ngang (theo tung độ). Nếu các đỉnh trên một

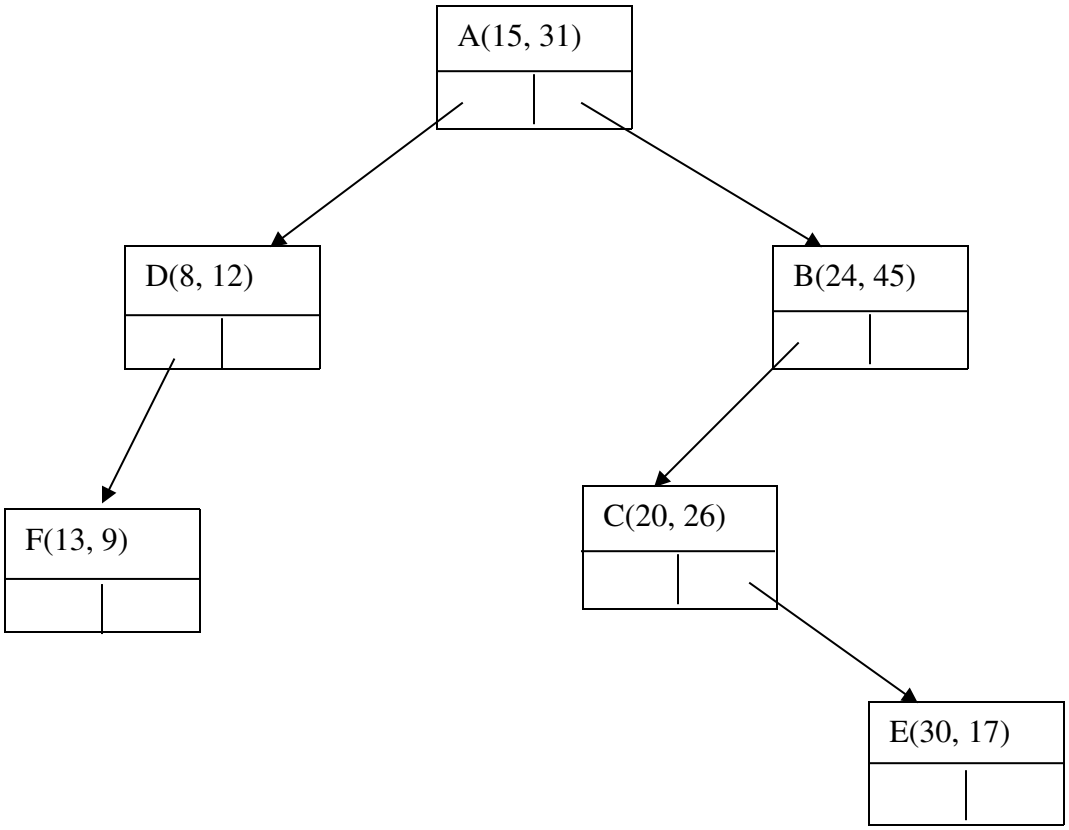
mức phân hoạch dữ liệu theo đường thẳng đứng, thì ở mức tiếp theo các đỉnh sẽ phân hoạch dữ liệu theo đường nằm ngang, các đỉnh ở mức tiếp theo nữa lại phân hoạch dữ liệu theo đường thẳng đứng, và cứ thế tiếp tục. Chẳng hạn, giả sử chúng ta có tập các điểm như sau:

Điểm	Toạ độ
A	(15, 31)
B	(24, 45)
C	(20, 26)
D	(8, 12)
E	(30, 17)
F	(13, 9)

Giả sử ta chọn điểm A đại biểu cho toàn bộ miền chứa các điểm, khi đó A là gốc cây và phân hoạch miền thành hai miền con theo đường thẳng đứng. Chọn điểm B đại biểu cho miền bên phải đường thẳng đứng đi qua A, chọn điểm D đại biểu cho miền bên trái đường thẳng đứng đi qua A, khi đó B là đỉnh con phải, còn D là đỉnh con trái của A, đồng thời đỉnh B và D phân hoạch các miền con tương ứng theo đường nằm ngang. Tiếp theo chọn C đại biểu cho miền con nằm dưới đường nằm ngang đi qua B, ta có C sẽ là đỉnh con trái của B và đỉnh C phân hoạch miền con đĩ theo đường thẳng đứng … Cuối cùng, chúng ta nhận được sự phân hoạch miền chứa các điểm thành các miền con như trong hình 14.1a và cây biểu diễn tập điểm tương ứng với phân hoạch này được cho trong hình 14.1b.



(a)



(b)

Hình 14.1. (a) Một cách phân hoạch tập điểm
(b) Cây 2- chiều tương ứng

Cần lưu ý rằng, có nhiều cách phân hoạch miền chứa dữ liệu và do đó chúng ta có thể biểu diễn tập dữ liệu điểm 2 - chiều bởi các cây 2 chiều khác nhau.

Mức của các đỉnh trong cây được xác định như sau. Gốc ở mức 0, nếu một đỉnh ở mức l, thì các con (nếu có) của nó ở mức l + 1. Bây giờ chúng ta có thể đưa ra định nghĩa cây 2-chiều.

Cây 2-chiều là cây nhị phân thoả mãn các điều kiện sau:

1. Nếu P là đỉnh ở mức chẵn và Q là đỉnh bất kỳ thuộc cây con trái của đỉnh P thì Q → Xval < P → Xval, còn nếu Q là đỉnh bất kỳ thuộc cây con phải của đỉnh P thì Q → Xval P → Xval.
2. Nếu P là đỉnh ở mức lẻ và Q là đỉnh bất kỳ thuộc cây con trái của P thì Q → Yval < P → Yval, và nếu Q là đỉnh bất kỳ thuộc cây con phải của P thì Q → Yval P → Yval.

Chú ý rằng, trong định nghĩa trên, các điều kiện đã nêu có nghĩa là các đỉnh ở mức chẵn (lẻ) sẽ phân hoạch miền mà chúng đại diện thành hai miền con theo đường thẳng đứng (theo đường nằm ngang).

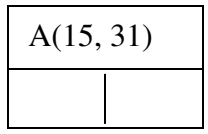
Sau đây chúng ta sẽ xét các phép toán từ điển: tìm kiếm, xen, loại và phép toán tìm kiếm phạm vi trên cây 2-chiều.

Phép toán tìm kiếm. Giả sử T là con trỏ trỏ tới gốc cây 2-chiều, chúng ta cần tìm xem điểm (x, y) có được lưu trong một đỉnh của cây T hay không. Thuật toán tìm kiếm trên cây 2-chiều cũng tương tự như thuật toán tìm kiếm trên cây tìm kiếm nhị phân. Chúng ta cho con trỏ P chạy trên các đỉnh của cây T, ban đầu P trỏ tới gốc cây. Nếu P NULL, ta kiểm tra xem (P → Xval, P → Yval) có trùng với (x, y) không. Nếu (P → Xval, P →Yval) = (x, y) thì sự tìm kiếm đã thành công và dừng lại. Giả sử chúng khác nhau và P ở mức chẵn, khi đó nếu P → Xval > x thì ta cho con trỏ P trỏ tới đỉnh con trái của nó, còn nếu P → Xval x thì cho P trỏ tới đỉnh con phải của nó. Tương tự, nếu P ở mức lẻ và P → Yval > y, thì cho P trỏ tới đỉnh con trái của nó còn nếu P → Yval y thì cho P trỏ tới đỉnh con phải của nó. Quá trình trên được lặp lại. Nếu tới một thời điểm nào đó P = NULL thì có nghĩa là sự tìm kiếm thất bại và dừng lại.

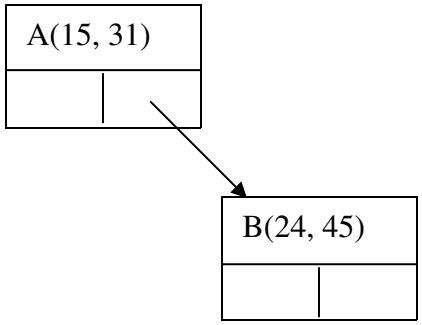
Phép toán xen. Giả sử chúng ta cần xen vào cây 2-chiều T một đỉnh mới chứa điểm (x, y). Nếu T là cây rỗng, ta tạo ra một đỉnh chứa điểm (x, y), các trường left và right chứa hằng NULL, và cho con trỏ T trỏ tới đỉnh này. Nếu cây T không rỗng, ta tiến hành giống như khi tìm kiếm, cho con trỏ P chạy trên các đỉnh của cây, bắt đầu từ gốc cây. Nếu P NULL và (P → Xval, P → Yval) = (x, y) thì điều đó có nghĩa là điểm (x, y) đã có sẵn trong cây T, ta dừng lại. Nếu P NULL và ta cần cho P trỏ tới đỉnh con trái của nó, nhưng P → left là NULL, thì ta tạo ra một đỉnh mới chứa điểm (x, y) và cho con trỏ P → left trỏ tới đỉnh mới này. Còn nếu P NULL và

ta cần cho P trỏ tới đỉnh con phải của P, nhưng P không có đỉnh con phải, thì đỉnh mới cần xen vào là đỉnh con phải của P.

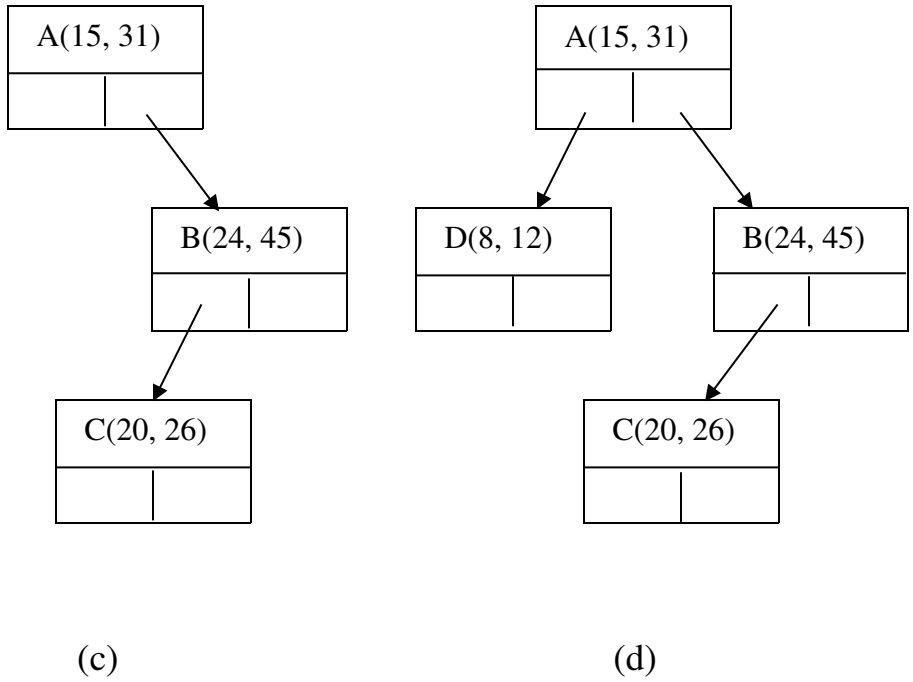
Ví dụ, giả sử chúng ta cần tạo ra cây 2-chiều biểu diễn tập điểm A(15, 31), B(24, 45), C(20, 26), D(8, 12), E(30, 17) và F(13, 9) bằng cách xen vào cây ban đầu rỗng lần lượt các điểm theo thứ tự đã liệt kê. Đầu tiên, xen đỉnh A(15, 31) vào cây rỗng, ta có cây chỉ có một đỉnh gốc A, như trong hình 14.2a. Xen vào cây này đỉnh B(24, 25), vì hoành độ của điểm B là 24, lớn hơn hoành độ 15 của điểm A, nên đỉnh B(24, 45) được thêm vào là đỉnh con phải của đỉnh A, ta có cây trong hình 14.2b. Xen đỉnh C(20, 26) vào cây 14.2b. Ta có đỉnh A ở mức chẵn và hoành độ 20 của điểm C lớn hơn hoành độ 15 của điểm A, nên đỉnh C(20, 26) phải được xen vào cây con phải của đỉnh A. Đỉnh con phải của A là đỉnh B ở mức lẻ, và tung độ 26 của C nhỏ hơn tung độ 45 của B, do đó đỉnh C(20, 26) được xen vào như là đỉnh con trái của B và ta có cây hình 14.2c. Tiếp tục, xen đỉnh D(8, 12) vào cây hình 14.2c, vì 8 <15 nên đỉnh D(8, 12) trở thành đỉnh con trái của A, như trong hình 14.2d. Tương tự, xen nốt các đỉnh E(30, 17) và F(13, 9), ta nhận được cây kết quả như trong hình 14.1b.



(a)



(b)



Hình 14.2. Xen vào cây 2-chiều lần lượt các điểm A(15, 31), B(24, 45), C(20, 26), D(8, 12)

Phép toán loại. Giả sử T là cây 2-chiều, và chúng ta cần loại khỏi cây này đỉnh chứa điểm (x, y). Phép toán loại là phép toán phức tạp nhất trong các phép toán trên cây 2-chiều. Tư tưởng của thuật toán loại trên cây 2-chiều cũng tương tự như trên cây tìm kiếm nhị phân.

Trước hết ta cần tìm đỉnh cần loại, giả sử đó là đỉnh P. Nếu P là đỉnh lá, việc loại P rất đơn giản, ta chỉ cần đặt con trỏ liên kết từ đỉnh cha của P tới P bằng NULL và thu hồi bộ nhớ đã cấp phát cho P. Giả sử P không phải là đỉnh lá, tức là ít nhất một trong hai cây con của P không rỗng. Ta ký hiệu T_l là cây con trái của P, T_r là cây con phải của P. Các hành động tiếp theo phụ thuộc vào cây con phải T_r là rỗng hay không rỗng. Ta xét từng khả năng:

Cây con phải T_r không rỗng. Trong trường hợp này ta thực hiện các bước cơ bản sau:

Bước 1. Tìm đỉnh Q thuộc T_r có thể thay thế cho đỉnh P. Đỉnh Q cần phải phân hoạch miền dữ liệu giống như P.

Bước 2. Thay thế đỉnh P bởi đỉnh Q. Điều này được thực hiện bằng cách chuyển dữ liệu chứa trong đỉnh Q lên đỉnh P, tức là đặt:

P → info = Q → info

P → Xval = Q → Xval

P → Yval = Q → Yval

Bước 3. Loại đệ quy đỉnh Q khỏi cây con T_r.

Chú ý rằng, vì cây con T_r có độ cao thấp hơn cây T nên các lời gọi đệ quy sẽ dẫn tới việc loại đỉnh lá, một việc đơn giản đã nói.

Bây giờ chúng ta làm sáng tỏ bước 1. Chúng ta cần tìm trong cây con T_r một đỉnh Q cho ta sự phân hoạch miền dữ liệu giống như đỉnh P. Điều này có nghĩa là, nếu P ở mức chẵn thì với mọi đỉnh L thuộc cây con trái của P, ta có L → Xval < Q → Xval, và với mọi đỉnh R thuộc cây con phải của P, ta có R → Xval Q → Xval. Còn nếu P ở mức lẻ, thì L → Yval < Q → Yval và R → Yval Q → Yval. Việc tìm đỉnh Q thuộc cây con phải T_r thoả mãn các tính chất trên là đơn giản. Nếu P ở mức chẵn, chúng ta chỉ cần tìm đỉnh Q là đỉnh có giá trị Xval là nhỏ nhất trong cây T_r; còn nếu P ở mức lẻ thì Q là đỉnh có giá trị Yval nhỏ nhất trong cây T_r. Chẳng hạn, trong cây hình 14.1b, giả sử ta cần loại đỉnh A(15, 31), khi đó đỉnh thay thế là đỉnh C(20, 26) trong cây con phải của đỉnh A. Từ hình 14.1a, ta thấy khi xoá đi điểm A thì điểm C cho ta sự phân hoạch dữ liệu giống như điểm A.

Cây con phải T_r rỗng và cây con trái T_r không rỗng. Trong trường hợp này, ta thực hiện các bước sau:

Bước 1. Tìm đỉnh Q thuộc cây con trái T_r có giá trị Xval nhỏ nhất (nếu P ở mức chẵn) hoặc Q có giá trị Yval nhỏ nhất (nếu P ở mức lẻ).

Bước 2. Thay thế đỉnh P bởi đỉnh Q. Chuyển cây con trái của P thành cây con phải của P, tức là đặt:

P → right = P → left

P → left = NULL

Bước 3. Loại đệ quy đỉnh Q khỏi cây con phải của P.

Phép toán tìm kiếm phạm vi. Vấn đề được đặt ra là, cho trước một điểm (x, y) và một số thực dương r, chúng ta cần tìm tất cả các điểm chứa trong các đỉnh của cây 2-chiều, nằm trong hình tròn tâm (x,y) với bán kính r. Để giải quyết vấn đề này, cần lưu ý rằng, mỗi đỉnh của cây 2-chiều biểu diễn một miền. Miền tương ứng với mỗi đỉnh là miền hình chữ nhật [X_1, X_2, Y_1, Y_2], trong đó (X_1, Y_1) là toạ độ của góc dưới bên trái và (X_2, Y_2) là toạ độ của góc trên bên phải của hình chữ nhật, miền này gồm tất cả các điểm (x, y) mà X_1 x < X_2 và Y_1 y < Y_2. Chẳng hạn, trong cây 2-chiều hình 14.1b, đỉnh A biểu diễn toàn bộ không gian, tức là miền [- , + ; - , +]. Đỉnh B biểu diễn miền [15, + ; - , +], đỉnh D biểu diễn miền [- , 15; - , +]. Miền tương ứng với đỉnh C sẽ là [15, + ; - ,45], … Miền được biểu diễn bởi gốc cây 2-chiều có thể xác định được ngay, khi mà không đánh giá được hình chữ nhật chứa toàn bộ dữ liệu, chúng ta có thể xem miền này là toàn bộ không gian. Nếu biết được miền hình chữ nhật tương ứng với một đỉnh, ta có thể xác định được miền hình chữ nhật tương ứng với các đỉnh con của nó. Và do đó, ta có thể xác định được miền hình chữ nhật tương ứng với mọi đỉnh của cây 2-chiều. (Bạn đọc hãy đưa ra thuật toán). Chúng ta sẽ ký hiệu hình tròn tâm (x,y), bán kính r là H, ký hiệu miền hình chữ nhật tương ứng với đỉnh P là R_P. Trong quá trình tìm kiếm, nếu chúng ta thấy miền được biểu diễn bởi đỉnh P không cắt hình tròn H, thì ta bỏ qua toàn bộ cây con gốc P, không cần xem xét cây con đó. Thuật toán tìm các điểm trên cây 2-chiều T (T là con trỏ trỏ tới gốc cây) nằm trong hình tròn H là thuật toán đệ quy như sau:

```
        RangeSearch (T, H)
{
    if (R_T    H =   )
        return ;
    else {
        if (( T → Xval , T → Yval)    H )
            In ra điểm (T → Xval, T → Yval) ;
        RangeSearch  (T → left, H );
        RangeSearch  (T → right, H );
        }
}
```

Ví dụ. Giả sử chúng ta cần tìm trên cây 2-chiều hình 14.1b các điểm nằm trong phạm vi hình tròn H với tâm (25, 22), bán kính 8. Đương nhiên là hình tròn H giao với miền được biểu diễn bởi đỉnh A. Kiểm tra ta thấy điểm (15, 31) không nằm trong hình tròn H. Xét đỉnh D(8, 12) là gốc của cây con trái của đỉnh A, miền được biểu diễn bởi đỉnh D không cắt hình tròn H, do đó ta không cần xem xét cây con trái của đỉnh A. Xét đỉnh B(24, 45) là đỉnh con phải của A, miền biểu diễn bởi đỉnh B giao với hình tròn H. Kiểm tra, ta thấy điểm (24, 45) không nằm trong hình tròn H. Lại xét đỉnh C(20, 26), miền được biểu diễn bởi đỉnh C giao với hình tròn H, và kiểm tra ta thấy điểm (20, 26) nằm trong hình tròn H. Tiếp tục xét đỉnh E(30, 17), và ta cũng thấy điểm (30, 17) nằm trong hình tròn H. Như vậy, ta tìm được hai điểm (20, 26) và (30, 17) nằm trong hình tròn tâm (25, 22), bán kính 8.

14.2.2 Cây k-chiều

Cây 2-chiều để biểu diễn các dữ liệu điểm 2 chiều. Chúng ta cần cây 3-chiều để biểu diễn các điểm (x, y, z) trong không gian 3 chiều, cây 4-chiều để biểu diễn các điểm (x, y, z, t), với t là chiều thời gian chẳng hạn. Tổng quát, các dữ liệu điểm k-chiều (x_o, x_1, …, x_{k-1}) (k > 2) có thể được biểu diễn bởi CTDL cây k-chiều. Cây k-chiều là sự tổng quát hoá tự nhiên của cây 2-chiều. Cây k-chiều cũng là cây nhị phân. Mỗi đỉnh của cây k-chiều là một cấu trúc giống như cấu trúc biểu diễn đỉnh cây 2-chiều, chỉ khác là thay cho các trường Xval và Yval chúng ta sử dụng mảng Xarray[k] để lưu điểm (x_o, x_1, …, x_{k-1}).

```
struct   Node
{
   infoType   info ;
   double   Xarray[k] ;
   Node*   left ;
   Node*   right ;
} ;
```

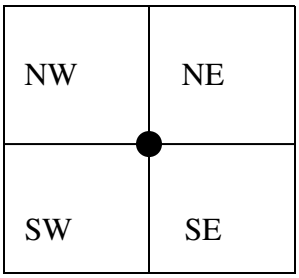
Giống như trong cây 2-chiều, các đỉnh nằm trên cùng một mức của cây k-chiều sẽ phân hoạch các miền mà chúng biểu diễn thành hai nửa theo một chiều không gian nào đó. Cụ thể là các đỉnh nằm ở mức m sẽ phân hoạch các miền tương ứng thành hai phần theo chiều không gian thứ

i với i = m mod k. Các thuật toán tìm kiếm, xen, loại, tìm kiếm phạm vi trên cây k-chiều là sự tổng quát hoá tự nhiên của các thuật toán tương ứng trên cây 2-chiều.

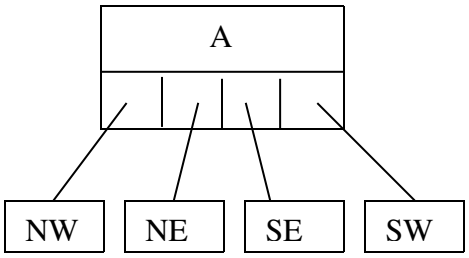
Nhận xét. Ưu điểm của cây k-chiều là dễ cài đặt. Cũng như trên cây tìm kiếm nhị phân, dễ dàng thấy rằng, thời gian thực hiện các phép toán tìm kiếm, xen, loại trên cây k-chiều là O(h), trong đó h là độ cao của cây. Trong trường hợp xấu nhất, cây k-chiều với n đỉnh có thể có độ cao n, và do đó thời gian thực hiện các phép toán tìm kiếm, xen, loại là O(n). Người ta đã chứng minh được rằng, thời gian thực hiện phép toán tìm kiếm phạm vị trên cây k-chiều với n đỉnh là O($kn^{1-1/k}$); nói riêng với k = 2, tìm kiếm phạm vi trên cây 2-chiều đòi hỏi thời gian O(2 $\sqrt{\ }$)

14.3 CÂY TỨ PHÂN

CTDL cây tứ phân (quadtree) được sử dụng để biểu diễncác điểm 2-chiều. Trong cây 2-chiều, mỗi đỉnh của cây phân hoạch miền mà nó biểu diễn thành hai phần hoặc là theo đường thẳng đứng (theo chiều x), hoặc theo đường nằm ngang (theo chiều y), và do đó cây 2-chiều là cây nhị phân. Trong cây tứ phân, mỗi điểm sẽ phân hoạch miền mà nó đại diện thành bốn phần theo cả hai chiều: phần tây-bắc (NW), phần đông-bắc (NE), phần đông- nam (SE) và phần tây-nam (SW), như trong hình 14.3a. Do đó, mỗi đỉnh trong cây tứ phân sẽ có bốn con tương ứng với một trong bốn phần đó như trong hình 14.3b.



(a)



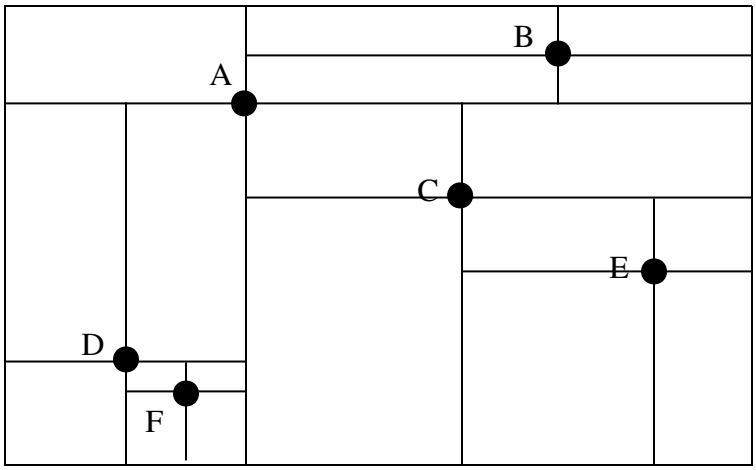
(b)

Hình 14.3. (a) Điểm A phân hoạch một miền thành bốn phần
(b) Điểm A được biểu diễn bởi một đỉnh có bốn con.

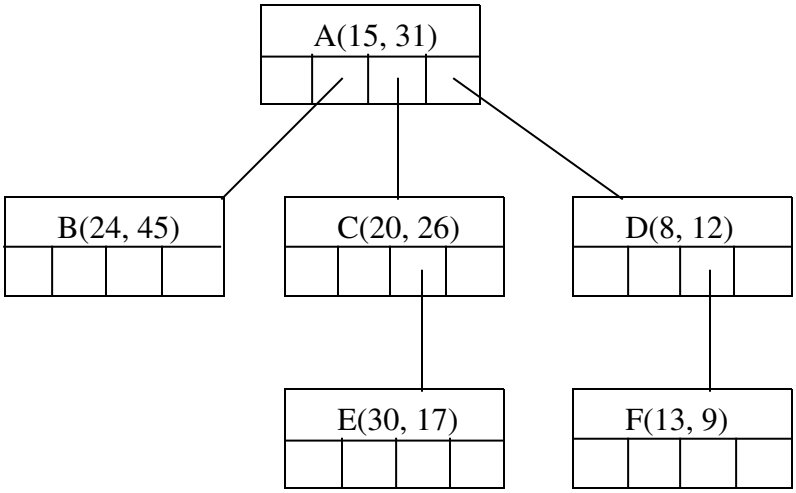
Cấu trúc đỉnh của cây tứ phân tương tự như cấu trúc đỉnh của cây 2-chiều, chỉ khác là ta cần đưa vào bốn con trỏ trỏ tới bốn đỉnh con.

```
struct   Node
{
  infoType   info ;
  double     Xval ;
  double     Yval ;
  Node*    NW-pointer ;
  Node*    NE-pointer ;
  Node*    SE-pointer ;
  Node*    SW-pointer ;
} ;
```

Ví dụ. Giả sử chúng ta cần biểu diễn tập điểm A(15, 31), B(24, 45), C(20, 26), D(8, 12), E(30, 17), F(13, 9), (tập điểm này đã được biểu diễn bởi cây 2-chiều hình 14.1b). Giả sử chúng ta sử dụng các điểm này để phân hoạch mặt phẳng thành các miền con như trong hình 14.4a. Tương ứng với phân hoạch này, tập điểm A, B, C, D, E, F được biểu diễn dưới dạng cây tứ phân hình 14.4b.



(a)



(b)

Hình 14.4. (a) Một cách phân hoạch mặt phẳng bởi các điểm A, B, C, D, E, F
(b) Cây tứ phân tương ứng với phân hoạch

Chú ý rằng, cũng như trong trường hợp xây dựng cây 2-chiều, bởi vì có nhiều cách chọn điểm đại biểu cho một miền, và do đó có nhiều cách phân hoạch một miền thành các miền con, nên có nhiều cây tứ phân khác nhau biểu diễn cùng một tập điểm.

Bây giờ chúng ta xét xem các phép toán tìm kiếm, xen, loại và phép toán tìm kiếm phạm vi được thực hiện như thế nào trên cây tứ phân.

Phép toán tìm kiếm và phép toán xen. Hai phép toán này được tiến hành theo cùng một phương pháp như trên cây 2-chiều. Để tìm kiếm (hoặc để xen vào) một điểm, chúng ta cần tìm miền chứa điểm đó. Mỗi điểm đại diện cho một miền hình chữ nhật. Nếu điểm P có toạ độ (x_P, y_P) biểu diễn miền hình chữ nhật $[X_1, X_2; Y_1, Y_2]$ thì điểm P sẽ chia miền này thành bốn miền con như sau:

Miền con NW = $[X_1, x_P; y_P, Y_2]$

Miền con NE = $[x_P, X_2; y_P, Y_2]$

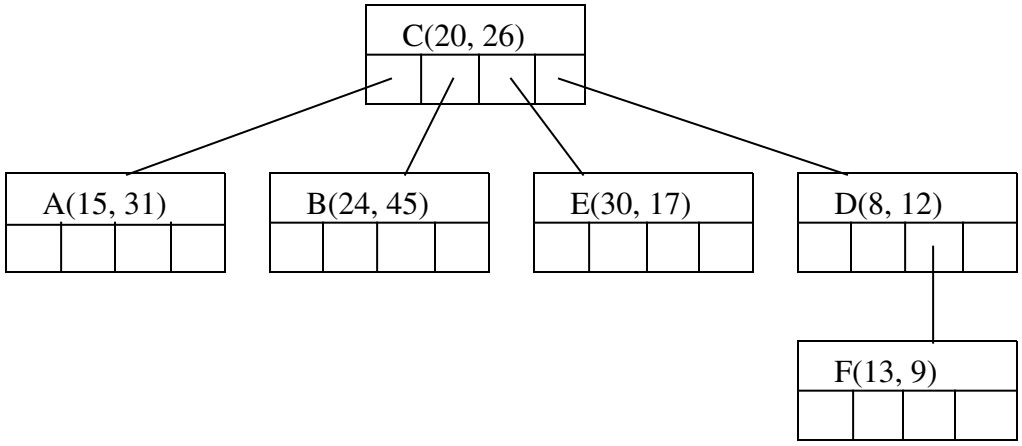
Miền con SW = $[X_1, x_P; Y_1, y_P]$

Miền con SE = $[x_P, X_2; Y_1, y_P]$

Muốn biết cây tứ phân có chứa điểm (x, y) hay không, chúng ta xem xét các đỉnh của cây, kể từ gốc. Nếu đỉnh đang xem xét là đỉnh P và điểm (x_P, y_P) là (x, y) thì có nghĩa là ta đã tìm kiếm thành công. Nếu không, tuỳ

thuộc điểm (x, y) nằm trong miền con nào trong bốn miền con NW, NE, SE, SW mà ta xem xét đỉnh con của P tương ứng với miền con đĩ. Nếu đỉnh con chúng ta cần xem xét không có, thì ta kết luận điểm (x, y) không có trong cây tứ phân. Chẳng hạn, giả sử ta cần tìm xem điểm (x, y) = (18, 23) có chứa trong cây tứ phân hình 14.4b hay không. Điểm ở gốc cây là (15, 31) (18, 23). Điểm (18, 23) nằm trong miền con SE đối với điểm (15, 31). Đi theo con trỏ SE tới đỉnh C(20, 26). Điểm (20, 26) (18, 23) và (18, 23) nằm trong miền con SW đối với (20, 26). Nhưng SW-pointer của đỉnh C(20, 26) là NULL, do đó ta kết luận điểm (18, 23) không có trong cây tứ phân hình 14.4b.

Bây giờ chúng ta xét phép toán xen. Giả sử ta xen tập điểm trong hình 14.4a vào cây ban đầu rỗng. Ta xen các điểm đó vào cây theo thứ tự C, B, A, D, E, F. Xen điểm C(20, 26) vào cây rỗng ta có cây chỉ có một đỉnh gốc là đỉnh C(20, 26). Các điểm B(24, 45), A(15, 31), D(8, 12), E(30, 17) lần lượt nằm ở miền NE, NW, SW, SE đối với điểm C, và do đó chúng lần lượt được thêm vào cây như các đỉnh con NE, NW, SW, SE của đỉnh C. Đến đây, ta có cây tứ phân gốc là đỉnh C, đỉnh C có bốn đỉnh con theo thứ tự NW, NE, SE SW là A, B, E, D. Bây giờ xen vào cây này điểm F(13, 9). Điểm F nằm trong miền SW đối với đỉnh C. Đi theo con trỏ SW tới đỉnh D. Điểm F nằm trong miền SE đối với đỉnh D, và do đó nó được thêm vào cây như là đỉnh con SE của đỉnh D. Ta thu được cây hình 14.5, cây này khác với cây hình 14.4b, mặc dầu chúng biểu diễn cùng một tập điểm.



Hình 14.5. Cây tứ phân được tạo thành bằng cách xen vào cây rỗng lần lượt các điểm C, B, A, D, E, F

Phép toán loại. Loại một đỉnh P(x, y) khỏi cây tứ phân là rất phức tạp. Nhớ lại rằng, để loại đỉnh P khỏi cây 2-chiều khi P không phải là đỉnh lá, ta tìm đỉnh Q trong cây con phải T_r của P sao cho Q cho phân hoạch miền dữ liệu giống như P, rồi thay P bởi Q và loại đệ quy Q khỏi cây con T_r. Đối với cây tứ phân ta có thể áp dụng kỹ thuật đó được không? Tức là, để loại đỉnh P khỏi cây tứ phân khi P không phải là đỉnh lá (nếu P là đỉnh lá thì việc loại nó là tầm thường), ta có thể tìm một đỉnh Q thuộc một trong các cây con NW, NE, SE, SW của đỉnh P, có thể thay thế cho đỉnh P, tức là Q phân hoạch miền dữ liệu giống như P? Nếu tìm được đỉnh Q như thế, ta thay P bởi Q và loại đệ quy Q khỏi cây con chứa nó. Song đáng tiếc là không phải lúc nào ta cũng tìm được đỉnh Q thay thế cho đỉnh P. Chẳng hạn, xét cây tứ phân hình 14.4b, nếu ta loại đỉnh A thì có thể tìm được đỉnh thay thế là đỉnh C. Nhưng nếu ta thêm vào cây hình 14.4b một điểm mới G(18, 23), thì trên cây này muốn loại đỉnh A ta không thể tìm được đỉnh thay thế.

Chúng ta có thể loại đỉnh P (không phải là lá) khỏi cây tứ phân theo thuật toán đơn giản sau. Cắt khỏi cây tứ phân cây con gốc P, rồi xen vào cây tứ phân còn lại các đỉnh nằm trong cây con gốc P, trừ đỉnh P. Thuật toán này đương nhiên là kém hiệu quả. Người ta đã đưa ra các thuật toán loại hiệu quả hơn, nhưng phức tạp nên chúng ta không trình bày.

Phép toán tìm kiếm phạm vi trên cây tứ phân được thực hiện theo cùng phương pháp như trên cây 2-chiều.

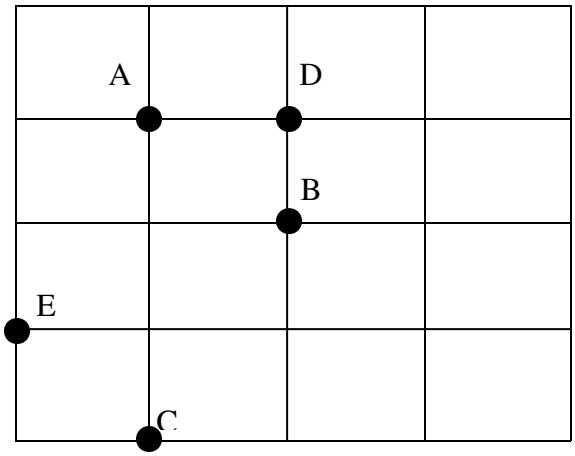
Cây tứ phân cũng rất dễ cài đặt. Thời gian thực hiện các phép toán tìm kiếm và xen vào là O(h), trong đó h là độ cao của cây, tức là cũng như trên cây 2-chiều. Tuy nhiên, cùng biểu diễn một tập điểm thì nói chung cây tứ phân ngắn hơn cây 2-chiều, bởi vì mỗi đỉnh trong cây tứ phân có bốn con, trong cây 2-chiều mỗi đỉnh chỉ có hai con. Phép toán tìm kiếm phạm vi trên cây tứ phân cần thời gian O(2$\sqrt{n}$), với n là số đỉnh trong cây, thời gian này cũng giống như trên cây 2-chiều. Nhược điểm của cây tứ phân là phép toán loại phức tạp. Trong mục sau đây, chúng ta sẽ nghiên cứu một loại cây tứ phân khác được gọi là cây tứ phân MX, trong đó phép loại được thực hiện dễ dàng hơn nhiều.

14.4 CÂY TỨ PHÂN MX

Trong cây 2-chiều cũng như trong cây tứ phân, "hình dạng" của cây phụ thuộc vào thứ tự các điểm được xen vào cây. Chẳng hạn, cây tứ phân hình 14.4b và cây tứ phân hình 14.5 có hình dạng khác nhau, cây hình 14.4b được tạo thành khi ta xen vào cây rỗng các điểm theo thứ tự A, B,C, D, E, F, còn cây hình 14.5 được tạo thành khi ta xen các điểm đó theo thứ tự C, B, A, D, E, F. Do đó trật tự các điểm được xen vào cây ảnh hưởng đến độ cao của cây. Hiệu quả của các phép toán trên cây lại phụ thuộc vào độ cao của cây.

Trong mục này chúng ta sẽ nghiên cứu một dạng cây tứ phân được gọi là ***cây tứ phân MX (MX- QuadTree***). Cây tứ phân MX có ưu điểm là hình dạng của cây không phụ thuộc vào thứ tự các điểm được xen vào cây, hay nói cách khác, một tập điểm được biểu diễn bởi duy nhất một cây tứ phân MX. Một ưu điểm khác là, các phép toán trên cây tứ phân MX như tìm kiếm tìm kiếm phạm vi, xen vào cũng dễ dàng như trên cây tứ phân, còn phép toán loại thì đơn giản và hiệu quả hơn.

Ý tưởng biểu diễn tập điểm trên mặt phẳng bởi cây tứ phân MX là như sau. Chúng ta giả sử rằng tập điểm mà chúng ta quan tâm nằm trong một miền hình vuông. Chia hình vuông này thành một lưới 2^k x 2^k ô vuông (với k nào đó), điểm ở góc dưới bên trái được xem là có toạ độ (0, 0), điểm ở góc trên bên phải có toạ độ (2^k, 2^k). Trong các ứng dụng ta cần chọn k thích hợp sao cho các điểm mà ta quan tâm tương ứng với các điểm (i, j) trên lưới với 0 i, j 2^k – 1. Chẳng hạn, hình 14.6 là lưới 2^2 x 2^2 ô vuông (k = 2) với các điểm được quan tâm là A(1, 2), B(2, 2), C(1, 0), D(2, 3) và E(0, 1).



Hình 14.6. Một lưới ô vuông với k = 2

Cấu trúc đỉnh của cây tứ phân MX giống như cấu trúc đỉnh của cây tứ phân ta đã đưa ra trong mục 14.3. Gốc cây tứ phân MX biểu diễn miền hình vuông [0, 2^k; 0, 2^k], hay nói cách khác, hình vuông với điểm dưới bên trái là (0, 0), cạnh là 2^k. Mỗi đỉnh của cây tứ phân MX biểu diễn miền hình vuông với điểm dưới bên trái (i, j) cạnh a, ta ký hiệu hình vuông này là [(i, j), a], và sẽ phân chia hình vuông này thành bốn hình vuông con bằng nhau (xem hình 14.7).

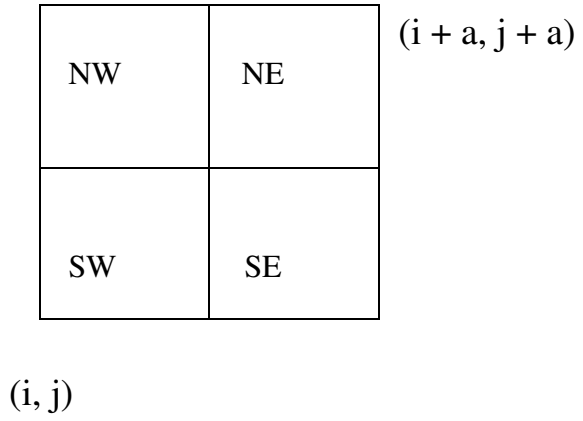
Hình vuông tây-bắc (hình vuông NW): [(i, j + a/2), a/2]

Hình vuông đông-bắc (hình vuông NE): [(i + a/2, j + a/2), a/2]

Hình vuông đông-nam (hình vuông SE): [(i + a/2, j), a/2]

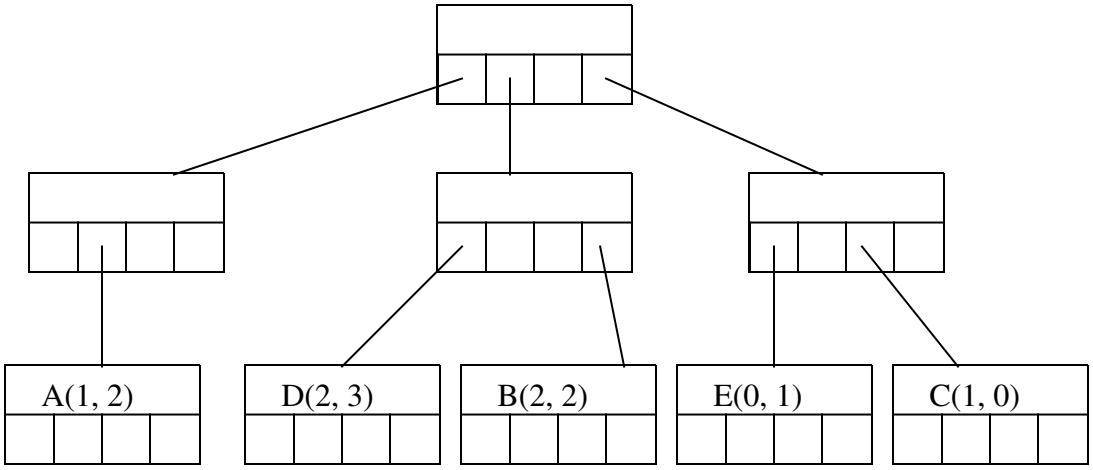
Hình vuông tây-nam (hình vuông SW): [(i, j), a/2]

Mỗi đỉnh chứa con trỏ trỏ tới đỉnh con biểu diễn hình vuông con tương ứng.



Hình 14.7. Đỉnh cây tứ phân MX phân chia hình vuông mà nó biểu diễn thành bốn hình vuông con

Gốc cây (đỉnh ở mức 0) biểu diễn miền hình vuông với cạnh là 2^k. Các đỉnh ở mức 1 biểu diễn các miền hình vuông con với cạnh là 2^{k-1} , … Các đỉnh ở mức k biểu diễn các hình vuông với cạnh là 1. Các điểm dữ liệu (x, y), chẳng hạn các điểm A, B, C, D, E trong hình 14.6, được xem là biểu diễn các ô vuông [(x, y), 1] và do đó được lưu giữ trong các đỉnh ở mức k (các đỉnh lá). Ví dụ, các điểm A(1, 2), B(2, 2), C(1, 0), D(2, 3), E(0, 1) trên lưới ô vuông 2^2 x 2^2 (k = 2) hình 14.6 được biểu diễn bởi cây tứ phân MX hình 14.8, trong đó các con trỏ theo thứ tự từ trái sang phải là NW, NE, SE, SW.

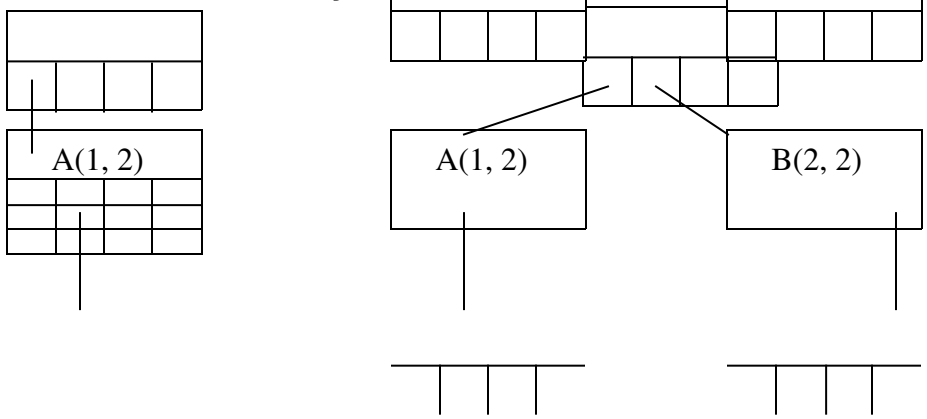


Hình 14.8. Cây tứ phân MX biểu diễn các điểm A, B, C, D, E trên lưới hình 14.6

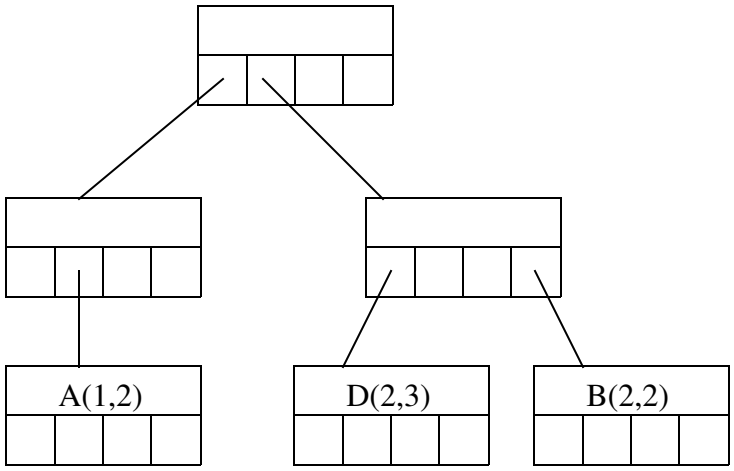
Như vậy, đặc điểm của cây tứ phân MX là: cây tứ phân MX biểu diễn tập điểm dữ liệu trên lưới 2^k x 2^k là cây tứ phân có độ cao k, tất cả các đỉnh lá đều nằm trên cùng một mức (mức k) và các điểm dữ liệu chỉ chứa trong các đỉnh lá.

Phép toán tìm kiếm và xen trên cây tứ phân MX được tiến hành theo cùng một phương pháp. Chúng ta xét phép toán xen. Giả sử ta cần xen vào cây ban đầu rỗng các điểm A, B, C, D, E trên lưới hình 14.6. Trước

hết ta xen vào cây điểm A(1, 2). Tạo ra gốc cây biểu diễn hình vuông [(0, 0), 2^2], nó chia hình vuông này thành bốn hình vuông con, điểm A nằm trong hình vuông NW. Tạo ra đỉnh con NW của gốc biểu diễn hình vuông [(0, 1), 2], đỉnh này lại chia hình vuông thành bống hình vuông con với cạnh là 1. Điểm A nằm ở hình vuông con NE, ta cho con trỏ NE từ đỉnh đó trỏ tới một đỉnh lá chứa điểm A, ta có cây hình 14.9a. Bây giờ ta xen vào cây đó điểm B(2, 2). Điểm B nằm trong hình vuông con NE của hình vuông được biểu diễn bởi gốc. Tạo ra đỉnh con NE của gốc biểu diễn hình vuông [(1, 1), 2], đỉnh này lại cia hình vuông thành bốn hình vuông con với cạnh là 1. Điểm B nằm trong hình vuông con SW, và do đó con trỏ SW của đỉnh đó sẽ trỏ tới một đỉnh lá chứa điểm B, ta nhận được cây hình 14.9b. Tiếp tục, ta xen vào cây hình 14.9b điểm D(2, 3). Điểm D nằm trong hình vuông con NE của hình vuông ứng với gốc. Đi theo con trỏ NE tới đỉnh biểu diễn hình vuông con đĩ, tức hình vuông [(1, 1), 2]. Điểm D lại nằm trong hình vuông NW, tức hình vuông [(2, 3), 1], và do đó ta cho con trỏ NW của đỉnh đó trỏ tới một đỉnh lá chứa điểm D, ta nhận được cây hình 14.9c. Tương tự xen tiếp các đỉnh còn lại C, E ta có cây kết quả như trong hình 14.8. Ta có nhận xét rằng, nếu ta xen các điểm A, B, C, D, E theo một thứ tự khác bất kỳ th n được c



(a) (b)



(c)

Hình 14.9. (a) Xen vào cây rỗng điểm A.
(b) Xen vào cây (a) điểm B.
(c) Xen vào cây (b) điểm D.

Phép toán loại. Phép toán loại trên cây tứ phân MX được thực hiện khá đơn giản. Trước hết ta cần lưu ý đến tính chất của cây tứ phân MX: tất cả các đỉnh lá đều nằm trên cùng một mức (mức k) và chỉ các đỉnh lá mới chứa các điểm dữ liệu. Khi thực hiện phép loại, chúng ta phải đảm bảo cây sau khi loại vẫn còn thoả mãn tính chất đó. Thủ tục loại khỏi cây tứ phân MX một đỉnh chứa điểm (x, y) là như sau. Tìm đỉnh lá chứa điểm (x, y), giả sử đó là đỉnh P. Giả sử cha của đỉnh P là đỉnh Q. Để loại đỉnh P, ta chỉ cần đặt con trỏ trong đỉnh Q trỏ tới P bằng NULL và thu hội vùng nhớ của đỉnh P. Sau đó kiểm tra, nếu đỉnh Q còn chứa con trỏ khác NULL thì không cần làm gì nữa, nếu bốn con trỏ trong Q đều là NULL, tức Q trở thành đỉnh lá, thì ta lại loại đỉnh Q. Lặp lại quá trình trên, trường hợp xấu nhất quá trình trên dẫn ta tới xem xét gốc cây có là đỉnh lá hay không.

Ví dụ, xét cây hình 14.8. Để loại đỉnh B, ta chỉ cần đặt con trỏ SW trong đỉnh cha của B bằng NULL. Nhưng nếu loại đỉnh A thì cha của đỉnh A trở thành lá, và do đó ta phải loại tiếp đỉnh đó bằng cách đặt con trỏ NW trong đỉnh gốc bằng NULL.

Phép toán tìm kiếm phạm vi trên cây tứ phân MX được tiến hành tương tự như trên cây tứ phân. Cần chú ý rằng, cây tứ phân và cây tứ phân

MX khác nhau ở chỗ, trong cây tứ phân tất cả các đỉnh đều chứa dữ liệu, còn trong cây tứ phân MX thì chỉ các đỉnh lá (các đỉnh ở mức k) mới chứa dữ liệu. Vì vậy, việc tìm các điểm dữ liệu trên cây tứ phân MX nằm trong phạm vi hình tròn C được tiến hành như sau. Ta xem xét các đỉnh của cây bắt đầu từ đỉnh gốc, giả sử đỉnh đang xem xét là P. Nếu hình vuông mà P biểu diễn không cắt hình tròn C thì ta không cần xem xét tiếp các đỉnh thuộc cây con gốc P. Còn nếu hình vuông ứng với đỉnh P cắt hình tròn C thì ta xem xét tiếp bốn đỉnh con (nếu có) của P, và trong trường hợp các đỉnh con của P là đỉnh lá thì ta kiểm tra xem các điểm chứa trong các đỉnh lá đó có nằm trong hình tròn C hay không. Thuật toán tìm kiếm phạm vi trên cây tứ pân MX được mô tả dưới dạng hàm đệ quy như sau, trong đó T là con trỏ trỏ tới gốc cây.

```
        RangeSearch (T, C)
    {
      if (Hình vuông được biểu diễn bởi đỉnh T không cắt C)
                    return ;
      else  if (Đỉnh T ở mức < k – 1)
            {
                RangeSearch (T → NW-pointer, C) ;
                RangeSearch (T → NE-pointer, C) ;
                RangeSearch ( T → SE-pointer, C) ;
                RangeSearch (T → SW-pointer, C) ;
            }
      else if (Đỉnh T ở mức k – 1)
            {
                Kiểm tra xem các điểm nằm trong các đỉnh con của T
có
                nằm trong hình tròn C hay không, nếu có thì in ra.
            }
    }
```

Dễ dàng thấy rằng, các phép toán tìm kiếm, xen, loại trên cây tứ phân MX biểu diễn tập điểm trên lưới 2^k x 2^k đòi hỏi thời gian O(k), trong đó k là độ cao của cây, và có thể chỉ ra rằng phép toán tìm kiếm phạm vi cần thời gian $O(2^k + m)$, trong đó m là số điểm nằm trong hình tròn.