Cấu trúc lưu trữ dữ liệu đa chiều

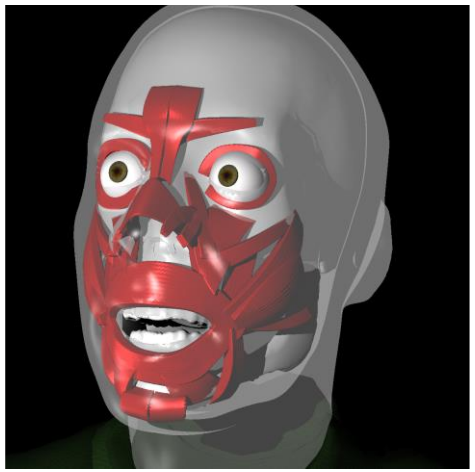
Nguyễn Đình Hóa
hoand@ptit.edu.vn 0942807711

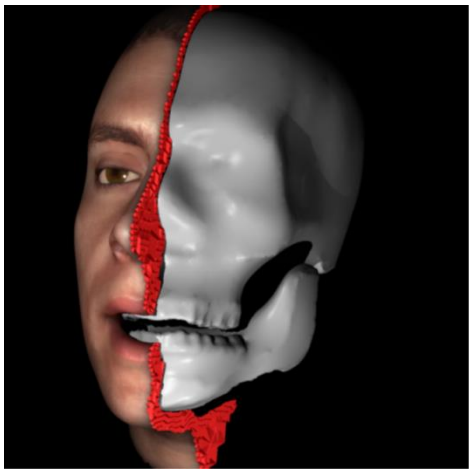
Cấu trúc lưu trữ dữ liệu đa chiều

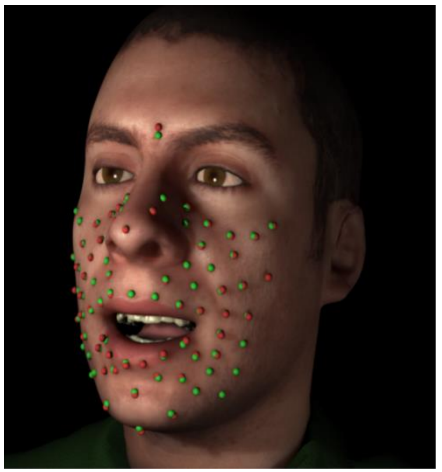
- Lưu trữ và tra cứu dữ liệu theo không gian đa chiều.
  - Cây k-d (k-d trees)
  - Cây tứ phân
  - Cây tứ phân MX
  - Cây tứ phân vùng điểm PR
  - Cây R

Cấu trúc lưu trữ dữ liệu đa chiều

- Ứng dụng:
  - Lưu trữ và khôi phục dữ liệu (về mặt không gian)
  - Tái tạo bề mặt
  - Phục vụ quan sát, học máy (machine learning)
  - Các công nghệ tương tác thông minh
  - Tối ưu hóa

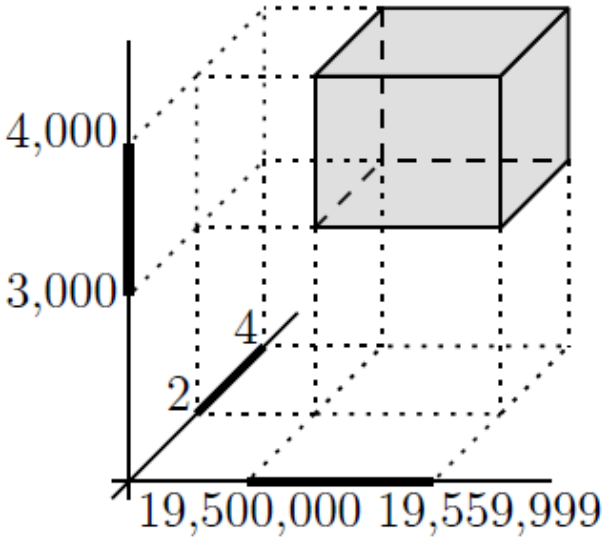


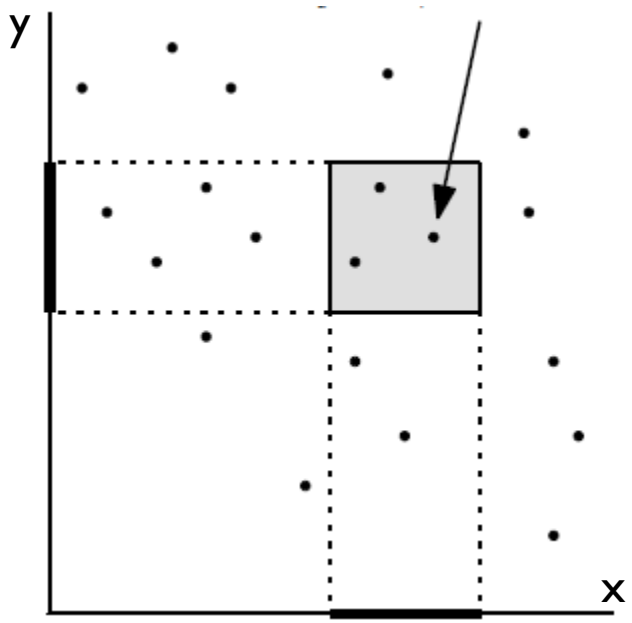




Cấu trúc lưu trữ dữ liệu đa chiều

- Truy vấn dữ liệu
  - Chính xác tuyệt đối: tìm kiếm dữ liệu chính xác theo tất cả các thuộc tính
  - Chính xác tương đối: dữ liệu chính xác nhưng không cụ thể thuộc tính nào
  - Chính xác theo các khoảng



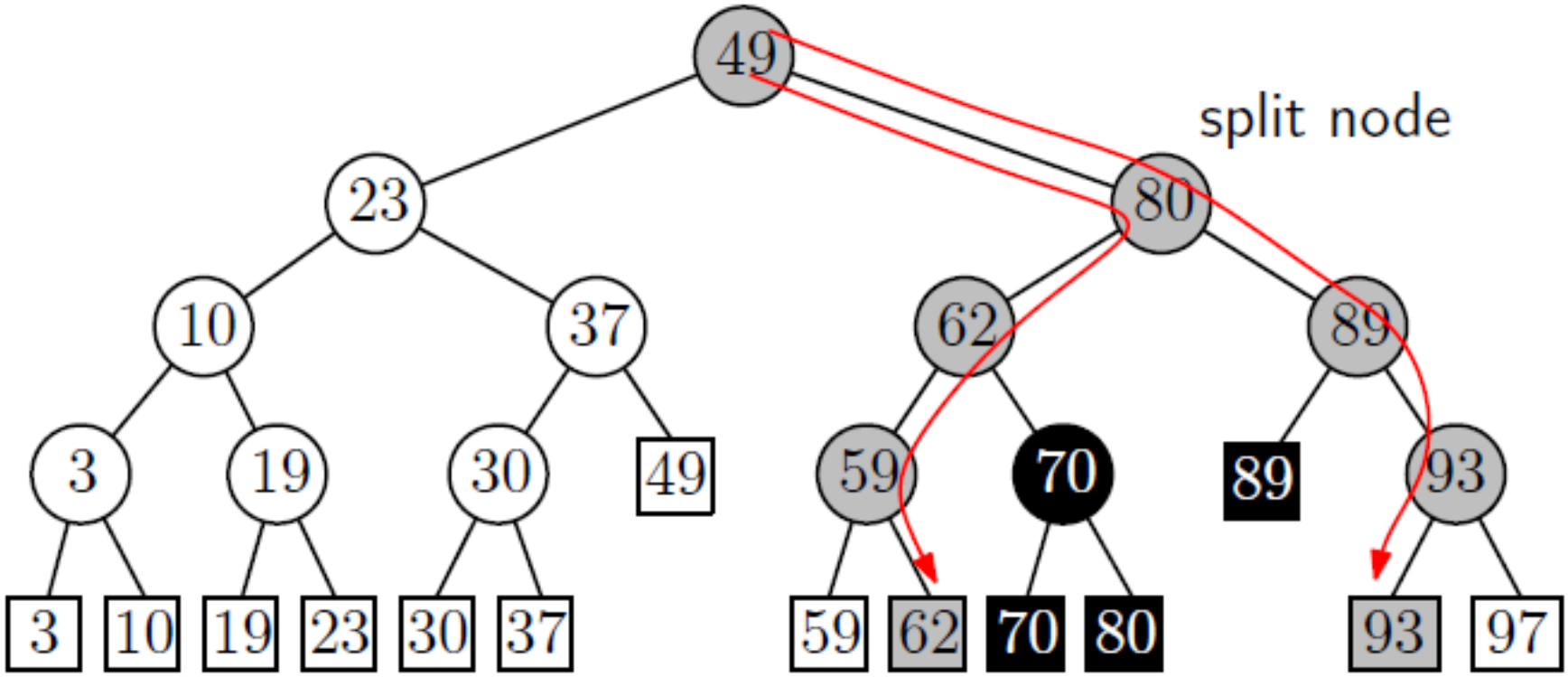


Cấu trúc lưu trữ dữ liệu một chiều

- Ký hiệu của 3 loại nút dữ liệu khi tra cứu
  - Nút trắng: không tìm đến trong khi tra cứu
  - Nút xám: nằm trên đường tìm kiếm nhưng chưa ra kết quả
  - Nút đen: là kết quả tìm kiếm, các nút phụ của chúng cũng là kết quả.

Cấu trúc lưu trữ dữ liệu một chiều

- Tìm các phần tử ngọn nằm trong khoảng [61, 90]



Cấu trúc lưu trữ dữ liệu đa chiều

- Ý tưởng:
  - Xây dựng các cây nhị phân theo từng thuộc tính (chiều) của dữ liệu
  - Thực hiện tra cứu dữ liệu theo từng thuộc tính (chiều) mà ta thấy có hiệu quả nhất (tìm kết quả nhanh nhất).

Cây k-d

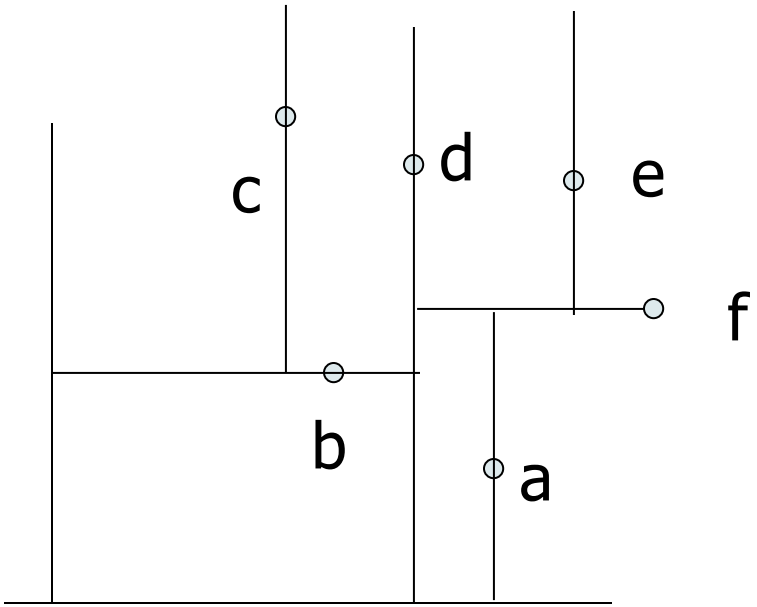
- Khái niệm
  - Là một cây phân nhóm dữ liệu theo không gian bằng phương pháp đệ quy
  - Mỗi nút là một bản ghi dữ liệu (có cấu trúc)
  - Phân chia lần lượt theo các trục thuộc tính của dữ liệu
  - Mỗi nút dữ liệu sẽ lưu trữ ít nhất một nút tiếp theo trên từng trục thuộc tính.

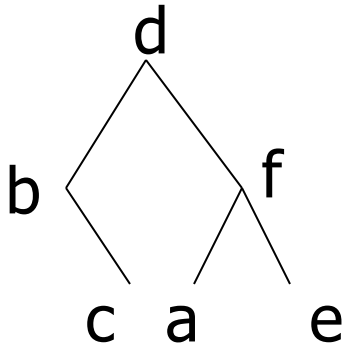
Cấu trúc cây k-d

- Dữ liệu đa chiều với k thuộc tính được lưu vào cây k-d
  - VD: Dữ liệu 2 chiều được lưu vào cây 2-d
- Biểu diễn và tìm kiếm dữ liệu lần lượt theo các thuộc tính (chiều)

Cấu trúc cây k-d

- Ví dụ: dữ liệu 2 chiều (cây 2-d)





Cấu trúc cây k-d

- Cách xây dựng cây k-d
  - Mỗi nút biểu diễn một miền phẳng,
  - Tại mỗi nút, mặt phẳng vuông góc với từng trục tọa độ chia miền dữ liệu thành 2 phần (trái và phải), mỗi phần đại diện cho một nút con của nút đó.
  - Thứ tự chia trục tọa độ thay đổi lần lượt theo từng cấp của cây dữ liệu.
    - VD: cây 2-d, cấp lẻ chia theo x, cấp chẵn chia theo y.
    - Cây 3-d: x -> y -> z -> x -> y …

Cấu trúc cây k-d

- Thêm nút ở cây 2-d:
  - Bắt đầu từ nút gốc (mức 0): thêm nút con vào bên trái hay bên phải theo trục x
  - Đến nút cấp 1: thêm nút con vào bên trái hay bên phải theo trục y
  - Đến nút cấp 2: thêm nút con vào bên trái hay bên phải theo trục x
  - ….
- Mở rộng sang cây 3-d, k-d

Cấu trúc cây k-d

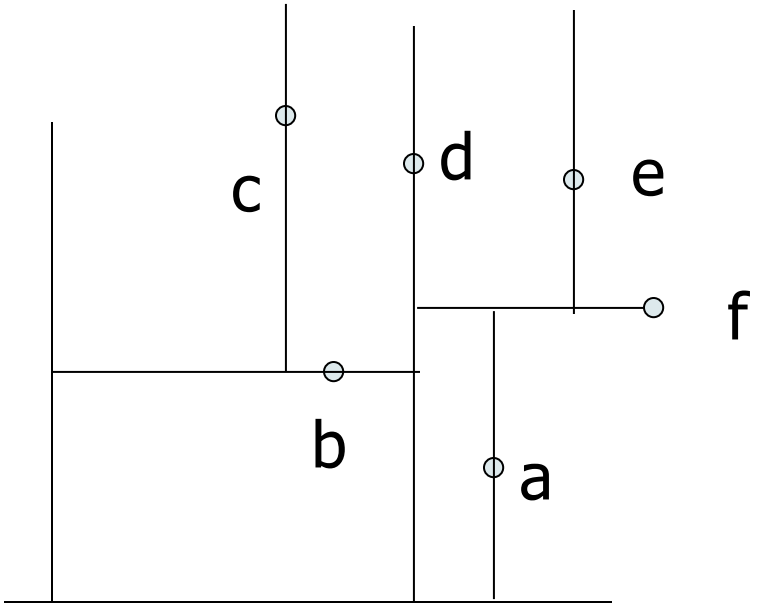
- Xóa nút ở cây 2-d
  - Tìm nút R chứa dữ liệu cần xóa
  - Nếu R là nút ngọn: không cần tìm nút thay thế.
  - Nếu R không là nút ngọn: tìm một nút thay thế P nằm trong các nhánh mà R là gốc (P có thể nằm ở nhánh trái hoặc nhánh phải của R)
  - Thay R bằng P,
  - Xóa P bằng cách lặp lại các bước trên.

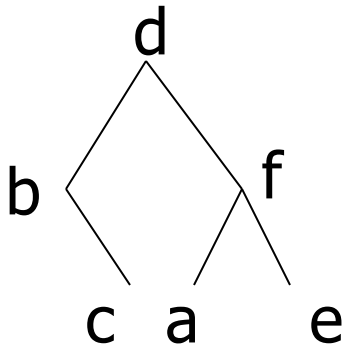
Cấu trúc cây k-d

- Tiêu chí đề chọn nút thay thế P:
  - Nếu P thuộc nhánh phải
    - Nút R ở cấp lẻ: chọn P có P.y là nhỏ nhất
    - R ở cấp chẵn: chọn P có P.x là nhỏ nhất
  - Nếu P thuộc nhánh trái
    - Nút R ở cấp lẻ: chọn P có P.y là lớn nhất
    - R ở cấp chẵn: chọn P có P.x là lớn nhất.

Cấu trúc cây k-d

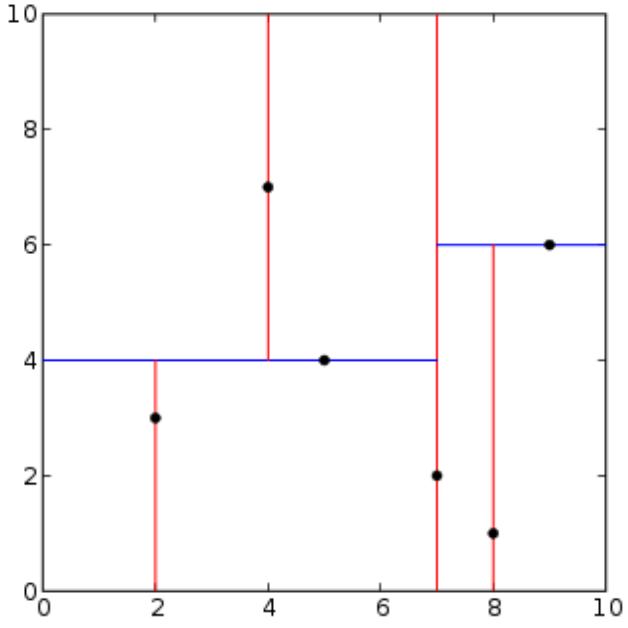
- Ví dụ: cây 2-d

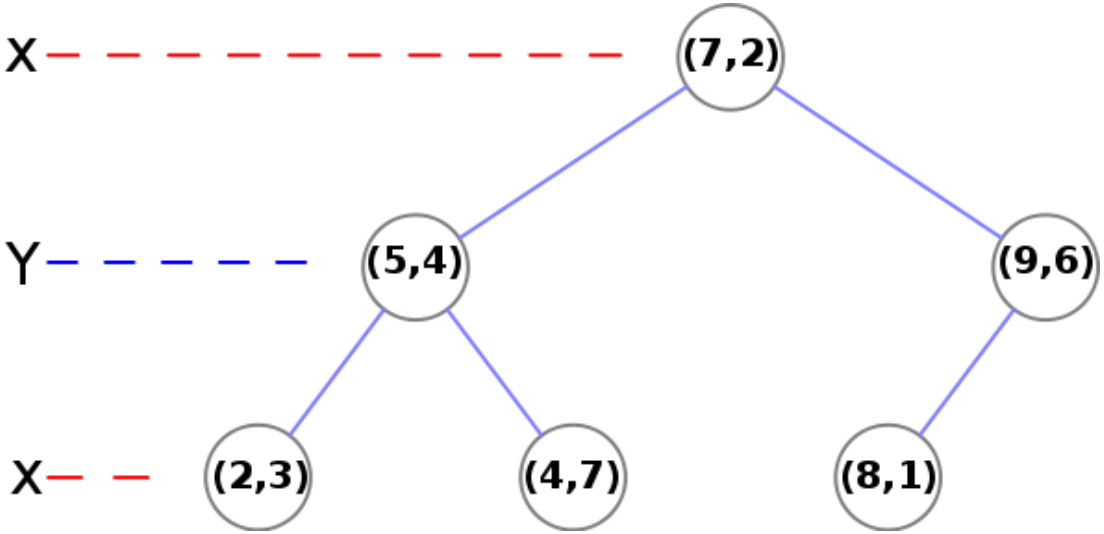




Cấu trúc cây k-d

- Ví dụ: cây 2-d





Cấu trúc cây k-d

- Những biến tấu của cấu trúc cây k-d:
  - Phân chia các nút con (trái, phải) dựa trên giá trị trung bình của từng thuộc tính
  - Phân chia các nút con (trái, phải) dựa trên điểm bình quân của từng thuộc tính.
- Mở rộng từ 2-d sang k-d (k>2): mỗi cấp phân chia theo từng thuộc tính, tuần tự lựa chọn các thuộc tính theo từng cấp.

Cấu trúc cây k-d

- Tra cứu dữ liệu
  - Tìm kiếm dữ liệu theo đường chứa các điểm sát nhất với câu truy vấn
  - 2 kiểu tra cứu thường gặp:
    - Nearest neighbor search (NN)
    - k-nearest neighbor seach (k-NN)

Cấu trúc cây k-d

- NN search
  - Bắt đầu từ nút gốc, đi dần xuống các nút con (trái hay phải) sao cho gần với câu truy vấn
  - Khi đã đi đến nút ngọn, chọn đó là “kết quả tốt nhất”
  - Thực hiện lại thuật toán, tìm kết quả mới.
  - Nếu kết quả mới tốt hơn “kết quả tốt nhất” thì chọn đó là “kết quả tốt nhất”
  - Thuật toán có thể thực hiện theo cả hai chiều (trái và phải) của cây k-d nếu như mặt phẳng phân chia tại nút gốc giao với khoảng giá trí cần tìm.

Cấu trúc cây k-d (kd-trees)

- k-NN search:
  - Giống NN search
  - Chọn và giữ k điểm “kết quả tốt nhất”
- Range search:
  - Tìm kiếm các điểm có từng thuộc tính nằm trong từng khoảng xác định.

Ví dụ

- Cho các dữ liệu theo cấu trúc vector trọng số như sau

  D_1 = [0.4, 0.3, 0.4]

  D_2 = [0.1, 0.1, 0.3]

  D_3 = [0.3, 0.2, 0.2]

  D_4 = [0.2, 0.2, 0.3]

  D_5 = [0.2, 0.1, 0.4]

  D_6 = [0.2, 0.2, 0.2]

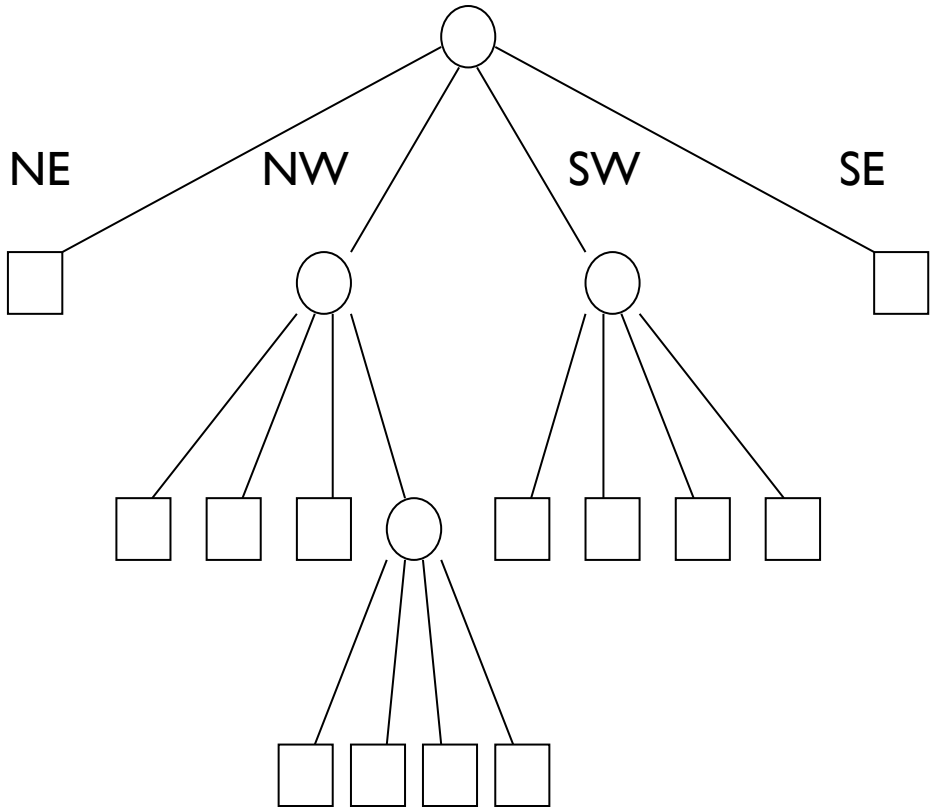
  D_7 = [0.1, 0.2, 0.3]

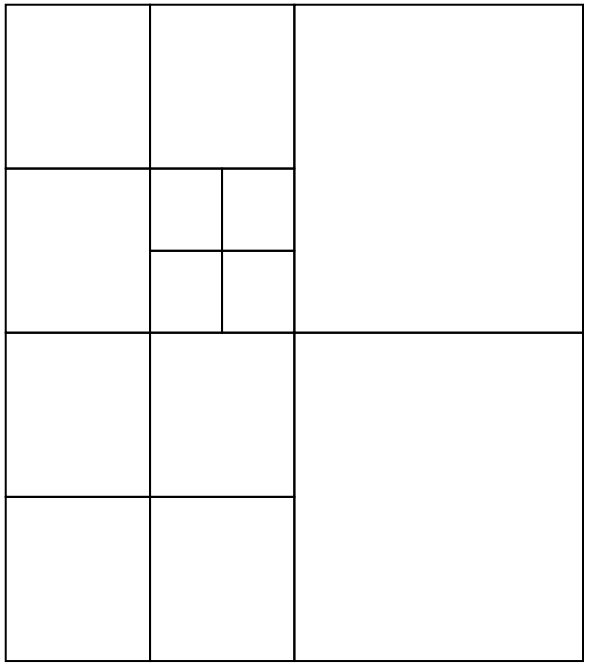
- Hãy sử dụng cây k-d để biểu diễn các dữ liệu trên trên bao gồm các thông tin về tên dữ liệu và vector tương ứng của dữ liệu đó
- Giả sử muốn xóa nút gốc thì làm thế nào?

Cấu trúc cây tứ phân (Quadtrees)

- Cách chia điểm giống cây k-d
- Tại mỗi nút, không gian được chia làm 4 phần theo 2 mặt phẳng vuông góc với từng cặp trục tọa độ (NW, SW, NE, SE)
- Mỗi phần tư tương ứng với một nút con của nút đó
- Mỗi nút có tối đa 4 nút con
- Trong dữ liệu nhiều chiều, việc phân chia miền dữ liệu được thực hiện dựa trên tuần tự từng cặp trục tọa độ
  - VD: dữ liệu 3 chiều (x, y, z): dữ liệu được chia theo từng cặp trục tọa độ (x, y) -> (y, z) -> (z, x) -> (x, y) ...

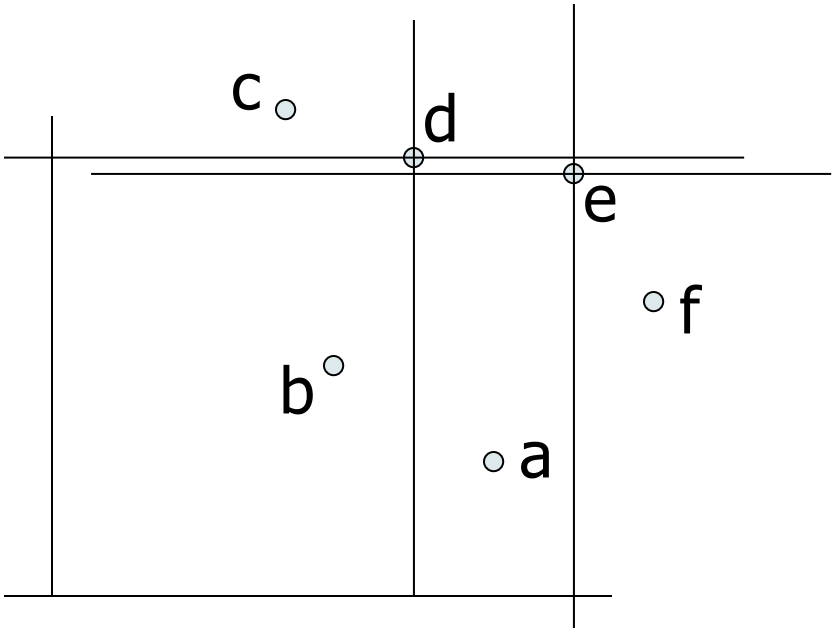
Cấu trúc cây tứ phân

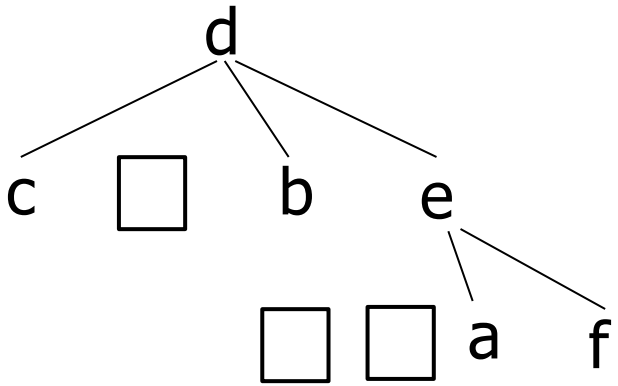




Cấu trúc cây tứ phân

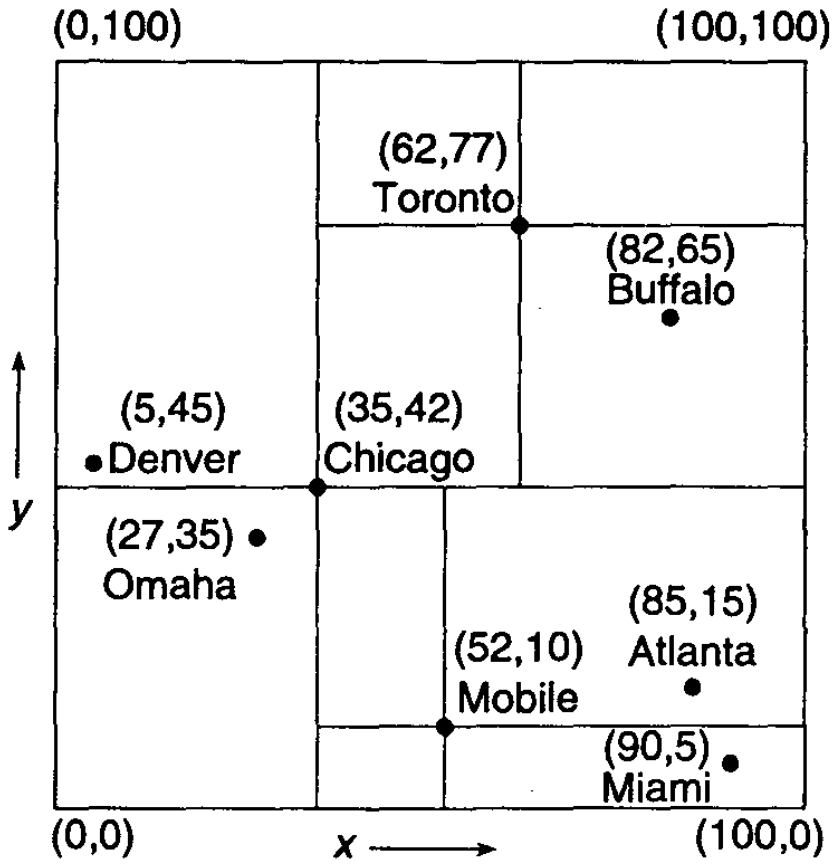
- Ví dụ:

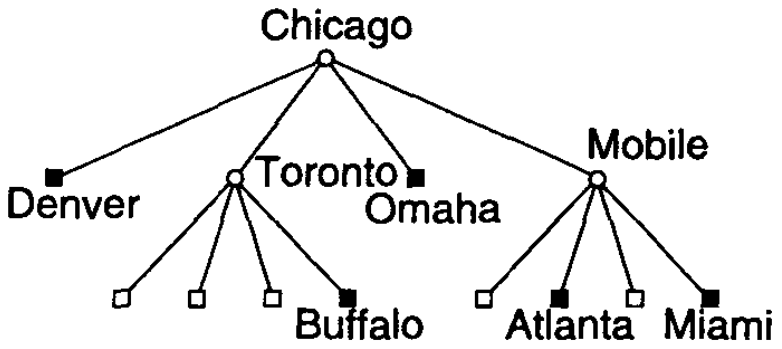




Cấu trúc cây tứ phân

- Ví dụ:





Cấu trúc cây tứ phân

- Xóa một nút trong cây tứ phân
  - Tìm nút R cần xóa
  - Tìm nút thay thế P thuộc một trong các miền NW, NE, SW, SE của R sao cho:
    - Nếu P thuộc R.NW thì tất cả các nút còn lại trong miền đó phải thuộc NW của P
    - Nếu P thuộc R.NE thì tất cả các nút còn lại trong miền đó phải thuộc NE của P,
    - Nếu P thuộc R.SW thì tất cả các nút còn lại trong miền đó phải thuộc SW của P.
    - Nếu P thuộc R.SE thì tất cả các nút còn lại trong miền đó phải thuộc SE của P.
  - Đôi khi khó tìm được nút thay thế P thỏa mãn yêu cầu.
  - Việc chọn P có thể dẫn đến sự phân bố lại các nút tại các miền

Cấu trúc cây tứ phân

- Tra cứu dữ liệu trong cây tứ phân
  - Giống như trong cây k-d
  - Tra cứu chính xác
  - Tra cứu gần đúng
  - Tra cứu theo khoảng cách.

Đặc điểm của cây k-d và cây tứ phân

- Hình dạng và cấu trúc của cây phụ thuộc vào thứ tự lựa chọn của các nút trong cây
- Mỗi sự lựa chọn ảnh hưởng đến quá trình nhập và tra cứu thông tin.
- Tại mỗi nút, không gian dữ liệu được chia thành 2 phần (cây k-d) hoặc 4 phần (cây tứ phân).
- Các nút dữ liệu được phân chia đồng đều hoặc không đồng đều tại các nhánh.

Ví dụ

- Cho các dữ liệu theo cấu trúc vector trọng số như sau

  $D_1 = [0.3, 0.2]$

  $D_2 = [0.2, 0.4]$

  $D_3 = [0.4, 0.4]$

  $D_4 = [0.3, 0.3]$

  $D_5 = [0.2, 0.3]$

  $D_6 = [0.3, 0.1]$

- Hãy sử dụng cây tứ phân để biểu diễn các dữ liệu trên bao gồm các thông tin về tên dữ liệu và vector tương ứng của dữ liệu đó

Ví dụ 2

- Cho các dữ liệu theo cấu trúc vector trọng số như sau

$D_1 = [0.3, 0.2, 0.1]$

$D_2 = [0.2, 0.4, 0.2]$

$D_3 = [0.4, 0.4, 0.1]$

$D_4 = [0.3, 0.3, 0.2]$

$D_5 = [0.2, 0.3, 0.3]$

$D_6 = [0.3, 0.1, 0.2]$

$D_7 = [0.2, 0.4, 0.1]$

- Hãy sử dụng cây tứ phân để biểu diễn các dữ liệu trên bao gồm các thông tin về tên dữ liệu và vector tương ứng của dữ liệu đó

Cây tứ phân MX

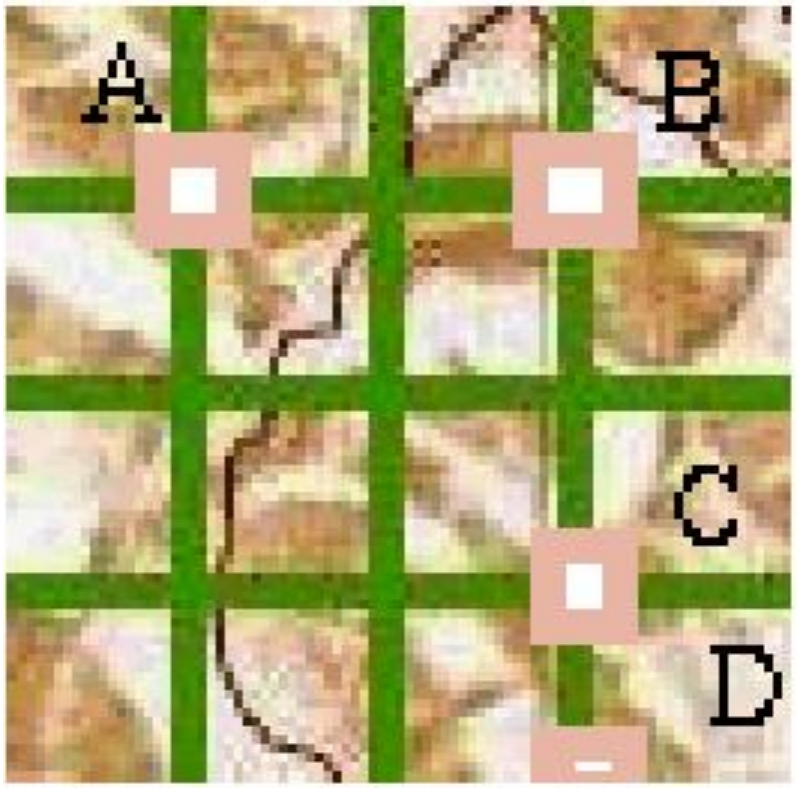
- Đặc điểm:
  - Hình dạng của cây tứ phân MX không phụ thuộc vào số lượng nút dữ liệu, cũng như thứ tự lựa chọn các nút.
  - Cho phép thêm hoặc xóa các nút, cũng như tìm kiếm dữ liệu một cách dễ dàng.
  - Tất cả các nút dữ liệu đều nằm ở ngọn và thuộc cùng một cấp (level)
  - Nếu CSDL có quá nhiều bản ghi, hoặc các bản ghi quá sát nhau, số cấp của cây có thể quá lớn, khó biểu diễn và truy vấn dữ liệu hiệu quả.

Cây tứ phân MX

- Cách xây dựng cây tứ phân MX: (VD: 2 chiều)
  - Mặt phẳng tọa độ được chia làm $2^k.2^k$ phần đều nhau (k cố định)
  - Nút gốc đại diện cho toàn bộ miền dữ liệu
  - Các miền dữ liệu được chia tại điểm trung tâm.
  - Tất cả các nút dữ liệu đều nằm ở mức k
- Mở rộng sang trường hợp nhiều chiều: chia miền không gian dữ liệu thành $2^k.2^k....2^k$ phần đều nhau, thực hiện giống như trên.

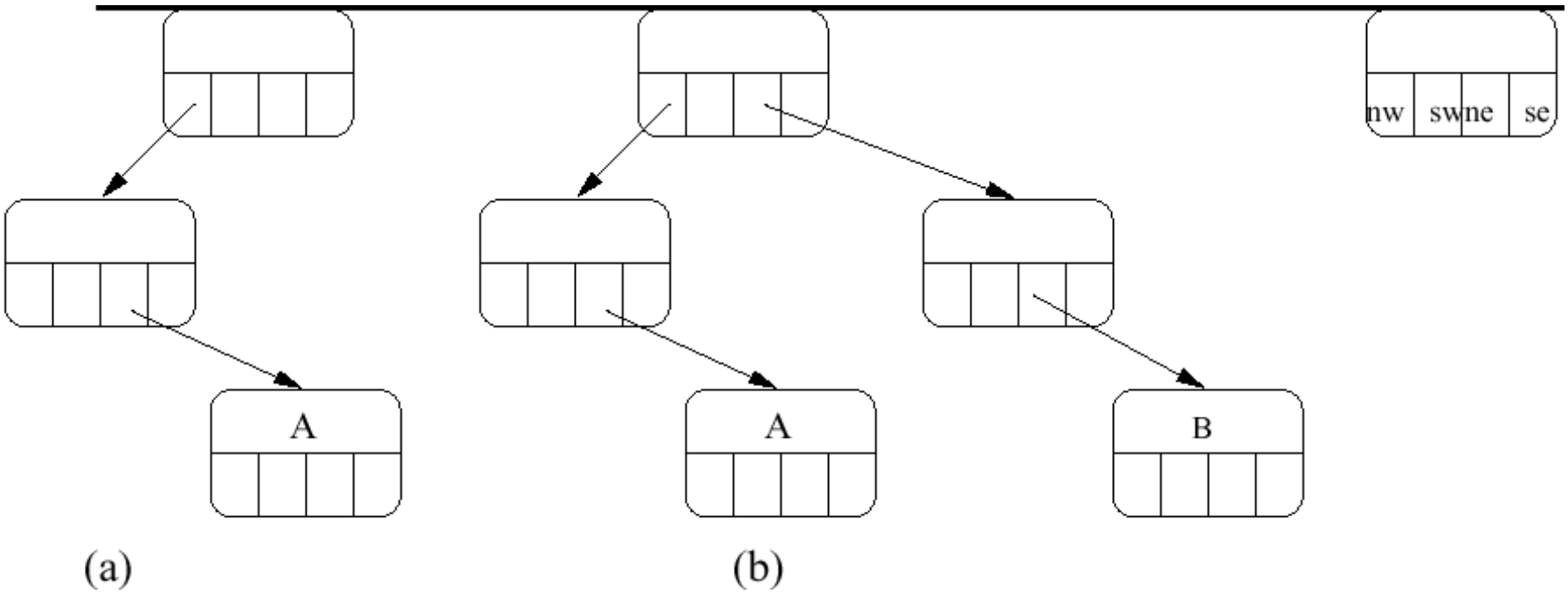
Cấu trúc cây tứ phân MX

- Cho 4 nút dữ liệu vào cây tứ phân MX



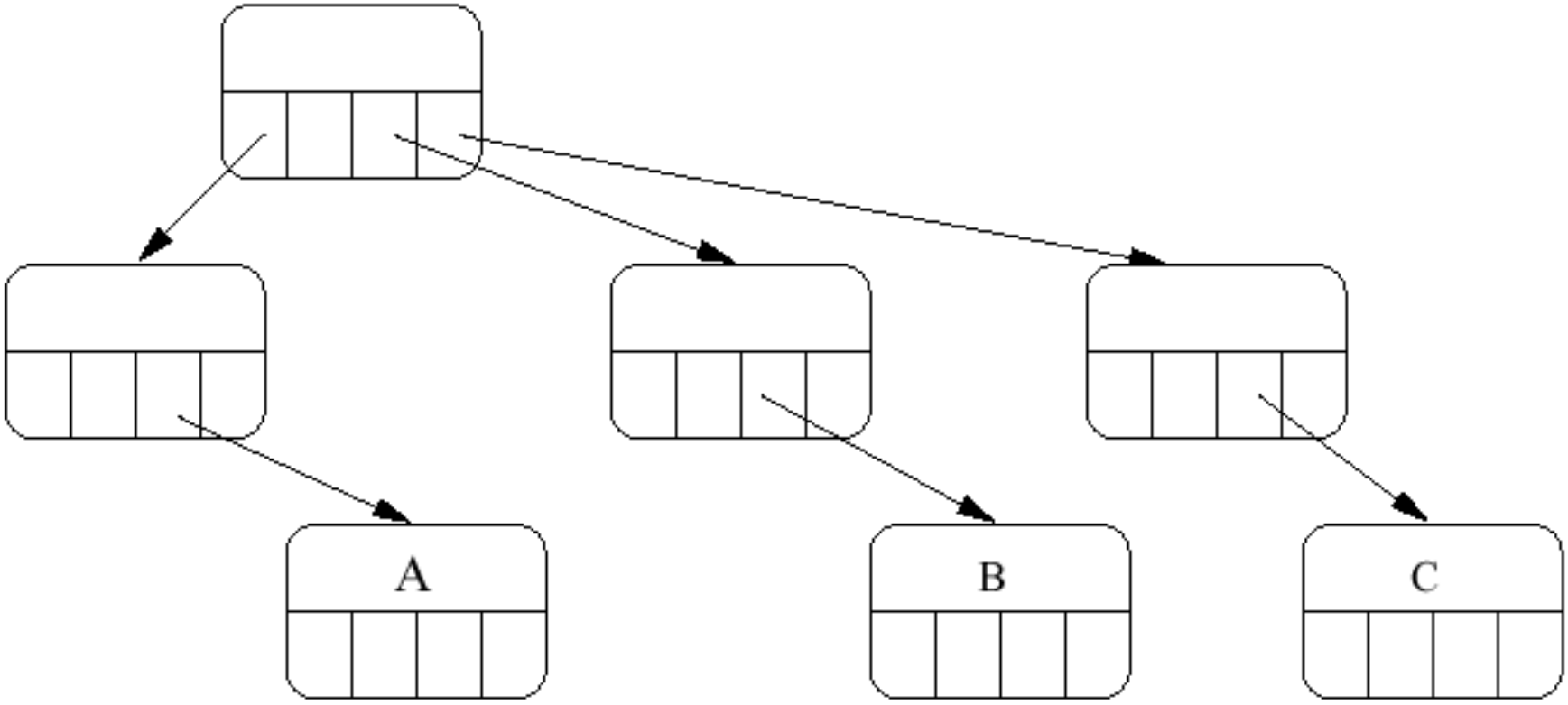
Cấu trúc cây tứ phân MX

- Thêm các nút dữ liệu A, B vào cây tứ phân MX



Cấu trúc cây tứ phân MX

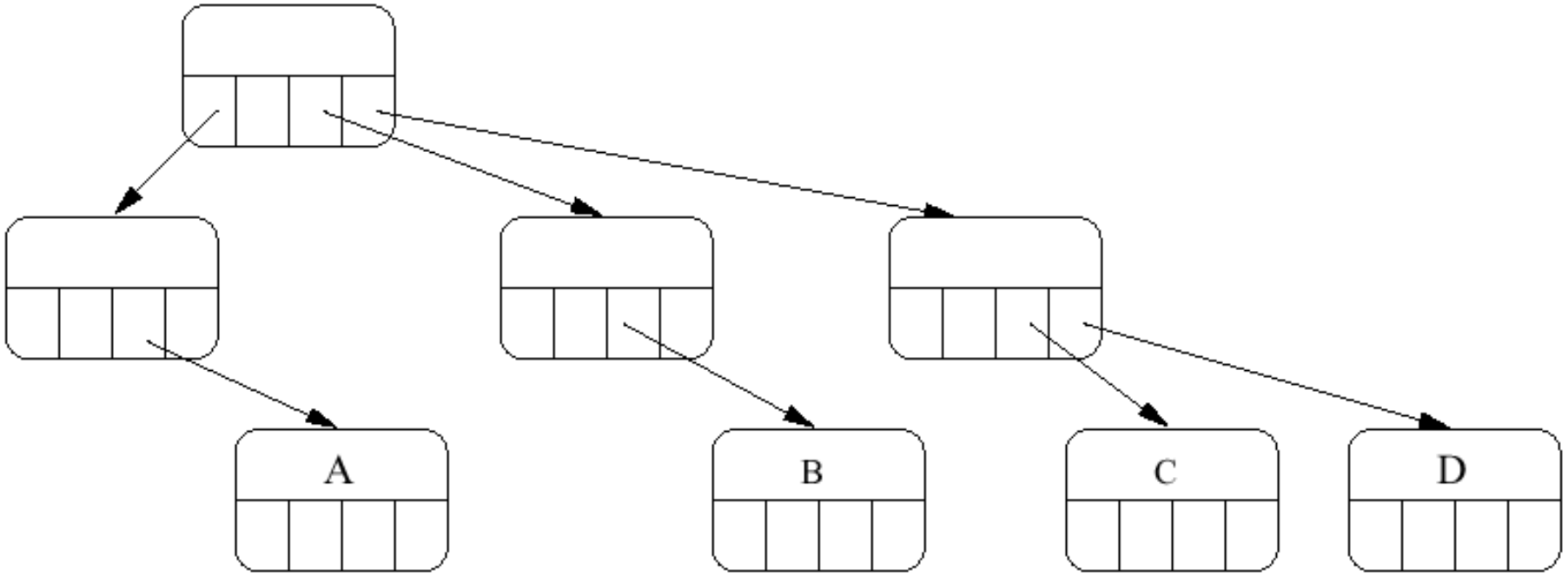
- Thêm nút dữ liệu C vào cây tứ phân MX



(c)

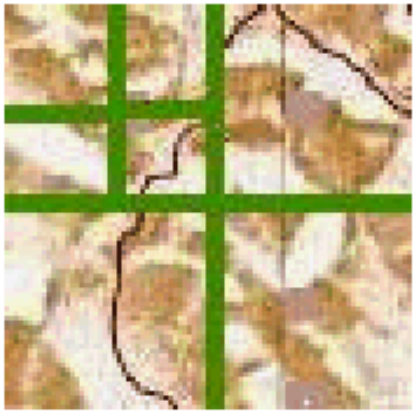
Cấu trúc cây tứ phân MX

- Thêm nút dữ liệu D vào cây tứ phân MX

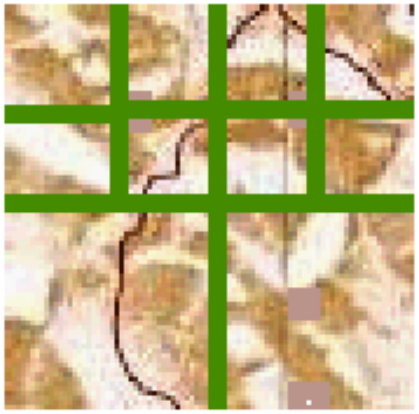


(d)

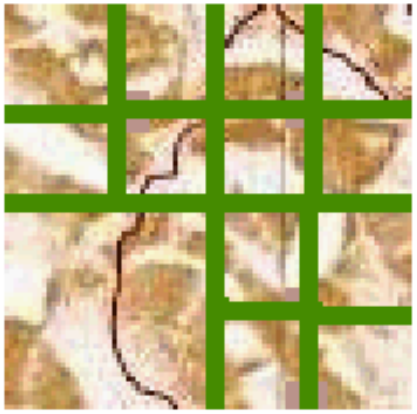
Cấu trúc cây tứ phân MX



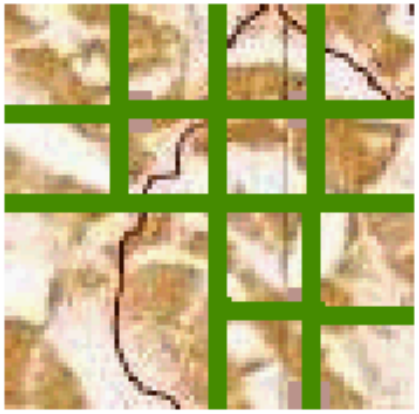
After insertion of A



After Insertion of B



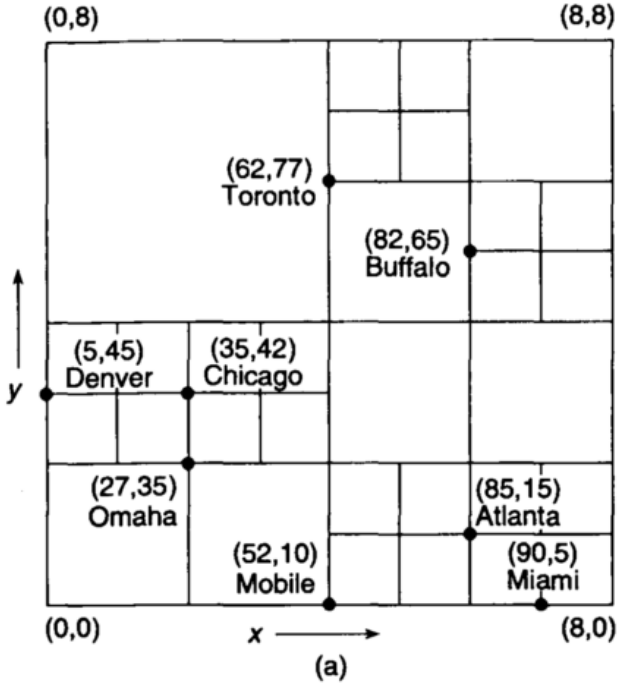
After insertion of C



After Insertion of D

Cấu trúc cây tứ phân MX (MX Quadtrees)

- Ví dụ:



Cấu trúc cây tứ phân MX

- Xóa một nút:
  - Thực hiện dễ dàng vì tất cả các nút đều cùng một cấp
  - Trong trường hợp nút cần xóa R không phải là nút ngọn, miền dữ liệu do R đại diện chắc chắn có thêm một nút thuộc cây tứ phân MX. Thuộc tính này cần được bảo tồn sau khi xóa nút R.
- Tra cứu dữ liệu
  - Giống như cây tứ phân

Ví dụ

- Cho các dữ liệu theo cấu trúc vector trọng số như sau

  $D_1 = [0.3, 0.2]$

  $D_2 = [0.2, 0.3]$

  $D_3 = [0.3, 0.3]$

  $D_4 = [0.4, 0.5]$

  $D_5 = [0.2, 0.2]$

  $D_6 = [0.5, 0.6]$

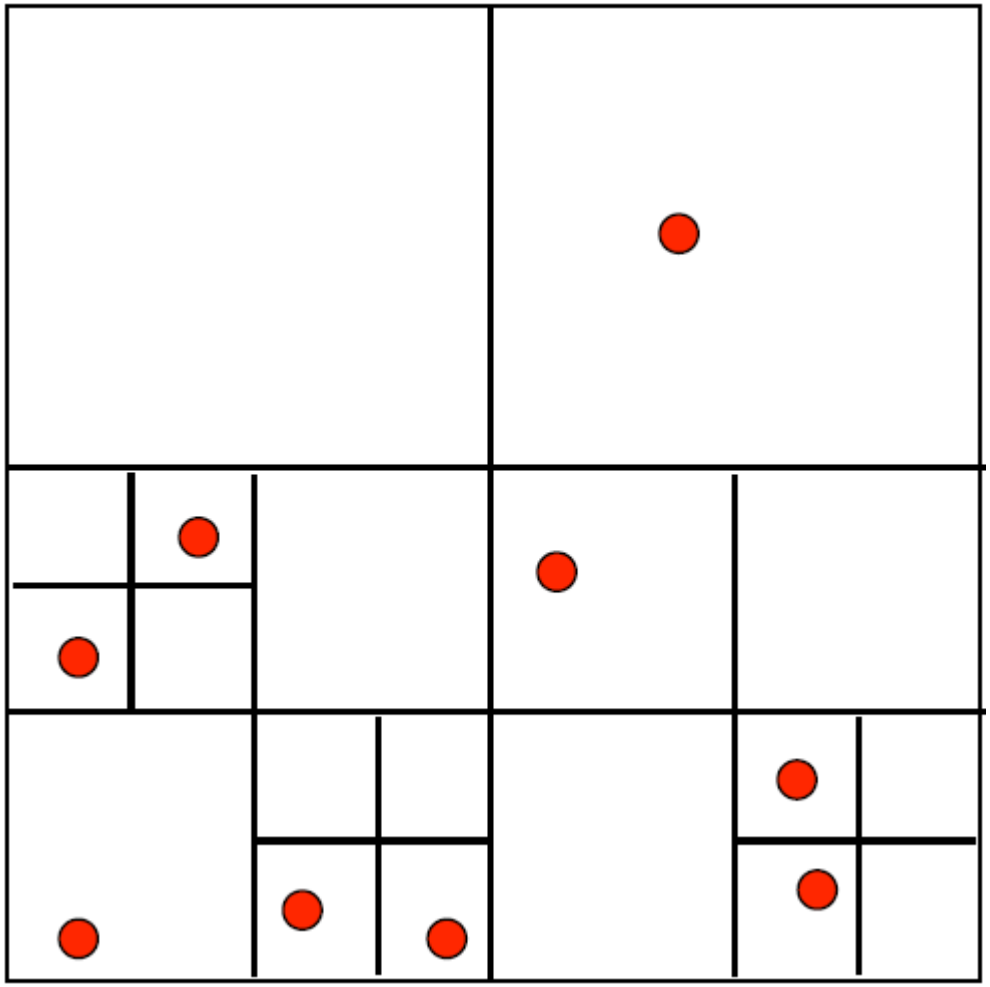
  $D_7 = [0.6, 0.3]$

- Hãy sử dụng cây tứ phân MX để biểu diễn các dữ liệu trên bao gồm các thông tin về tên dữ liệu và vector thuộc tính tương ứng của dữ liệu đó
- Hãy bổ sung dữ liệu $D_8 = [0.3, 0.4]$

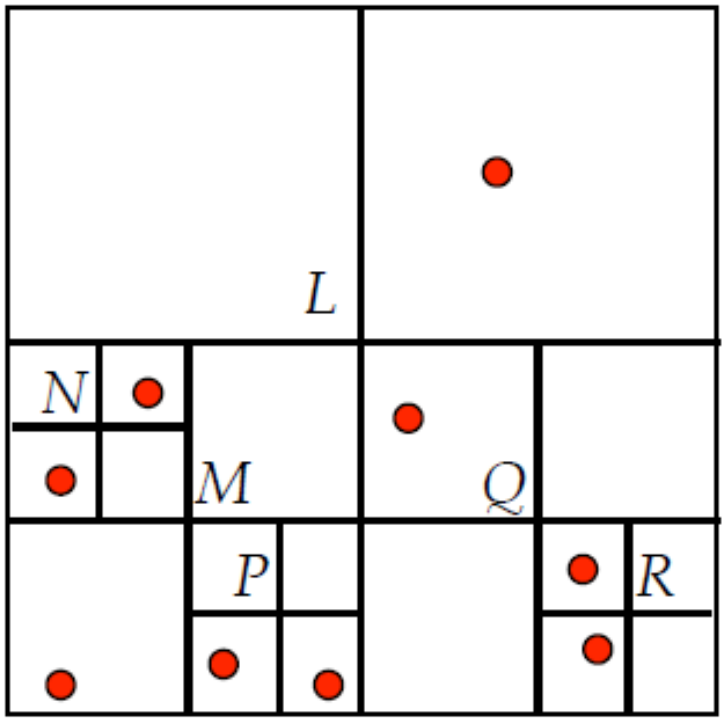
Cấu trúc cây tứ phân vùng điểm PR

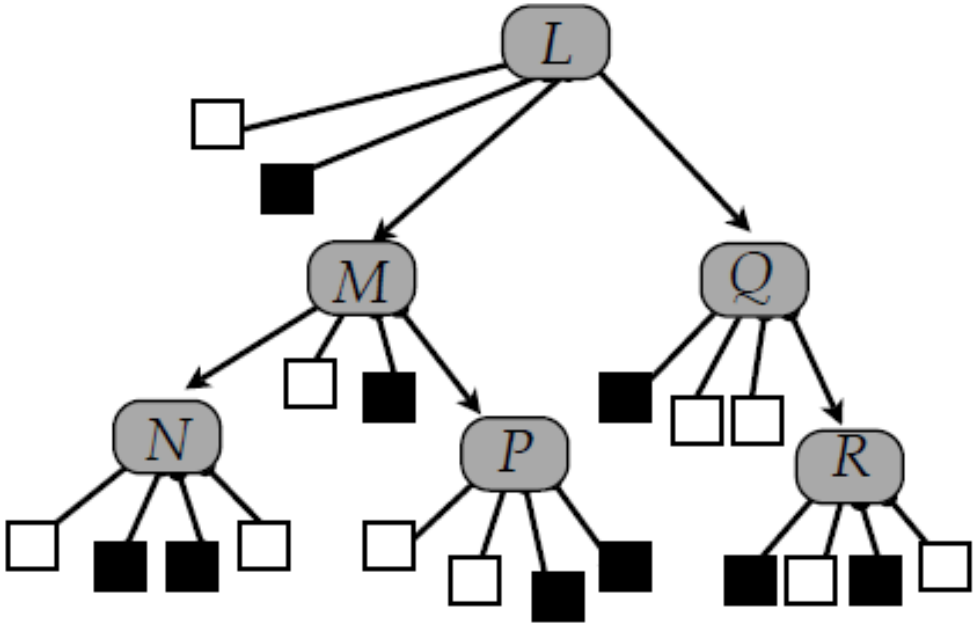
- Giống cây tứ phân MX
- Các miền dữ liệu được chia sao cho cuối cùng chỉ có duy nhất một nút dữ liệu đại diện cho một miền con.
- Không có nút dữ liệu nào là con của nút khác.

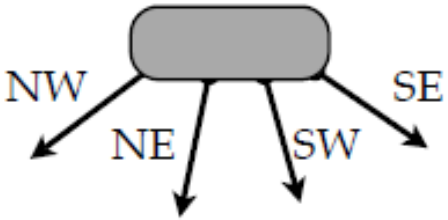
Cấu trúc cây tứ phân vùng điểm PR



Cấu trúc cây tứ phân vùng điểm PR

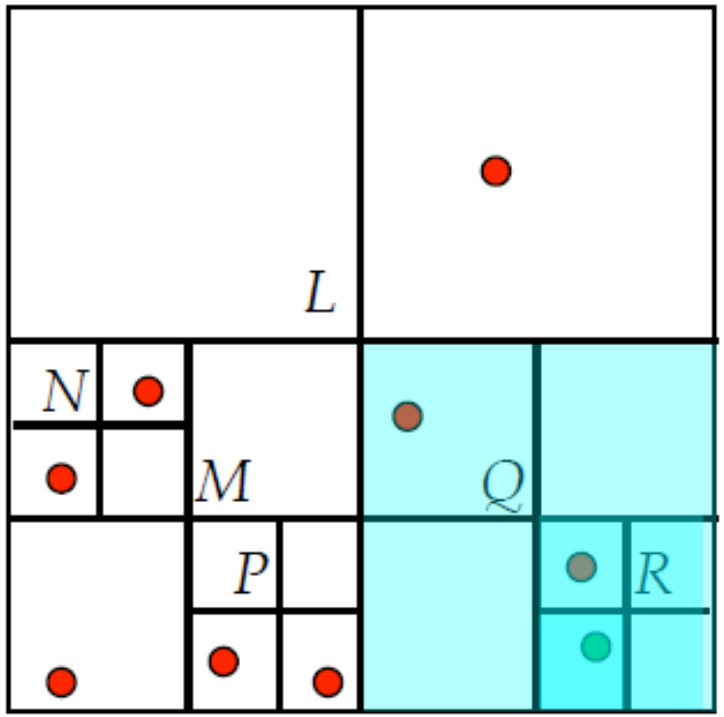


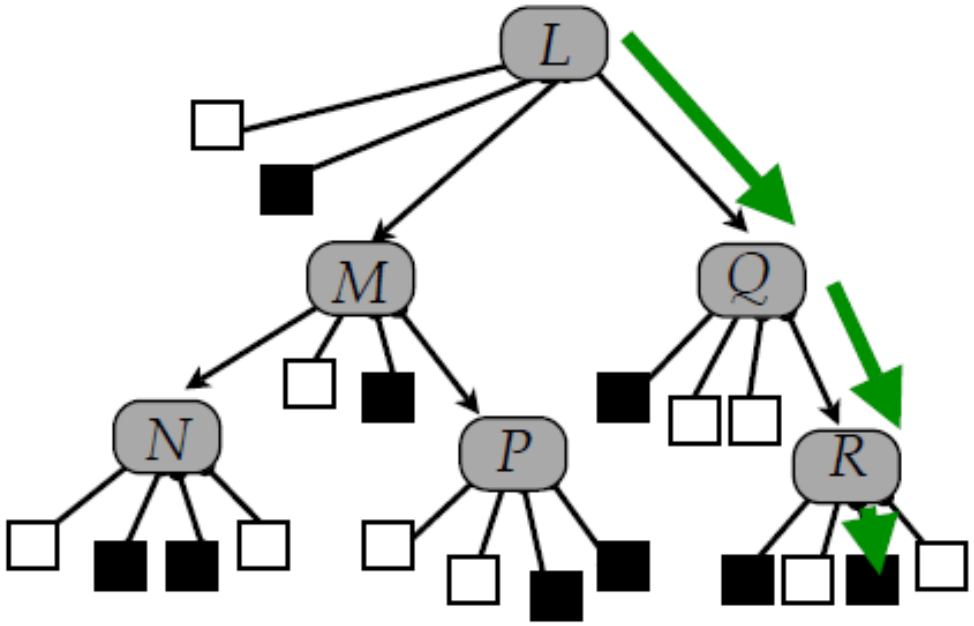




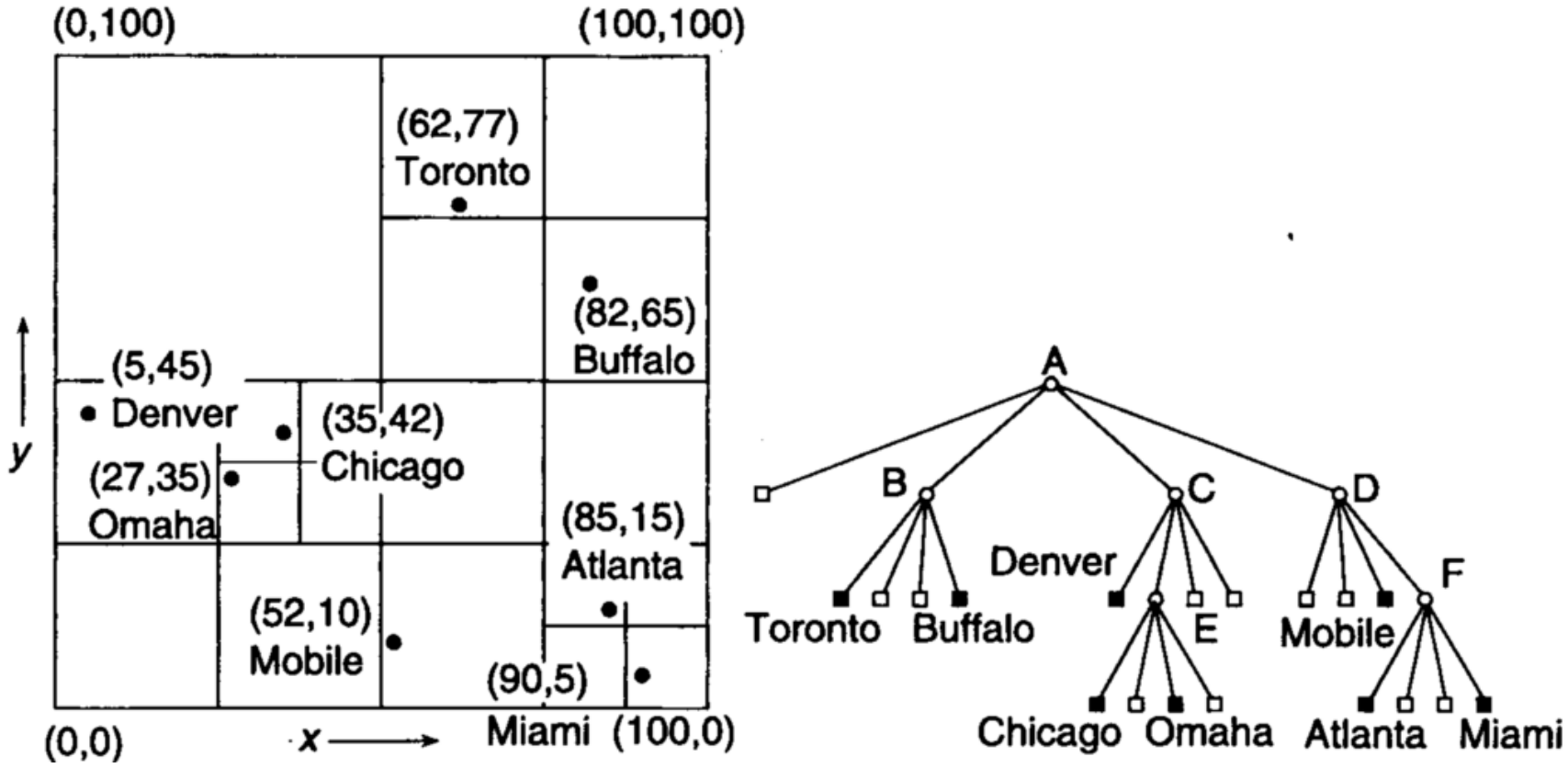
Cấu trúc cây tứ phân vùng điểm PR

- Tra cứu dữ liệu:





Cấu trúc cây tứ phân vùng điểm PR



Cấu trúc cây tứ phân vùng điểm PR (Point Region Quadtrees)

- Đặc điểm:
  - Miền phân chia không phụ thuộc vào điểm dữ liệu
  - Tất cả các ngọn đều chứa dữ liệu
  - Việc phân chia miền dữ liệu được thực hiện một cách đệ quy theo kích thước bộ dữ liệu
  - Mở rộng: cho phép có b>1 điểm dữ liệu trong một đơn miền.

Cấu trúc cây tứ phân vùng điểm PR

- Nhược điểm:
  - Đôi khi không hiệu quả: nếu bộ dữ liệu có nhiều điểm nằm sát nhau thì miền dữ liệu phải chia quá nhỏ, dẫn đến việc có quá nhiều cấp dữ liệu
  - Biểu diễn trên miền dữ liệu đa chiều tốn nhiều dung lượng lưu trữ.

Ví dụ

- Cho các dữ liệu theo cấu trúc vector trọng số như sau

  $D_1 = [0.3, 0.3]$
  $D_2 = [0.2, 0.2]$
  $D_3 = [0.4, 0.3]$
  $D_4 = [0.4, 0.4]$
  $D_5 = [0.2, 0.4]$
  $D_6 = [0.3, 0.6]$
  $D_7 = [0.5, 0.6]$

- Hãy sử dụng cây tứ phân PR để biểu diễn các dữ liệu trên bao gồm các thông tin về tên dữ liệu và vector tương ứng của dữ liệu đó

Cấu trúc cây R

- Dùng để lưu dữ liệu bằng các vùng hình chữ nhật,
- Hữu ích trong việc lưu trữ dữ liệu lớn, có ít thuộc tính (thường tối đa 10 thuộc tính), VD: GIS.
- Tối thiểu hóa việc truy cập thông tin trong thiết bị lưu trữ.

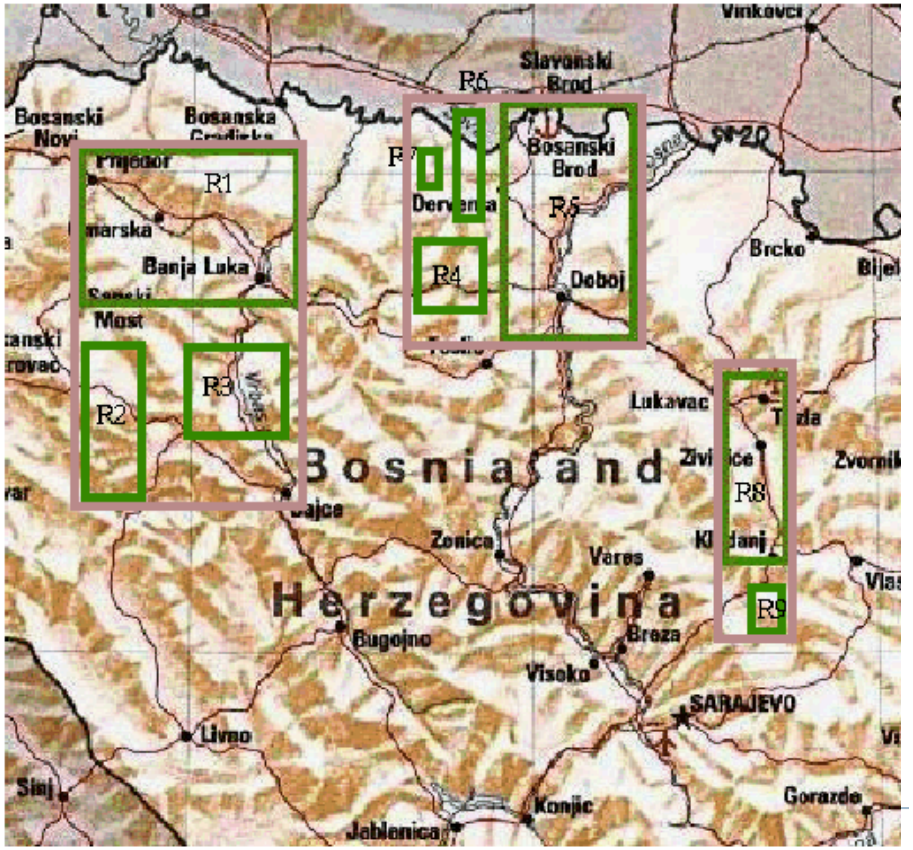
Cấu trúc cây R

- Đặc điểm
  - Mỗi cây R có một thứ tự nhất định các vùng chữ nhật và số cấp là K.
  - Mỗi nút không phải là ngọn (vùng cha) thì chứa tối đa K vùng dữ liệu con và ít nhất K/2 vùng dữ liệu con (hoặc ít nhất là 2 vùng con)
  - Mỗi nút là một hình chữ nhật có diện tích nhỏ nhất chứa các nút con của nó (Minimum Bounding Rectangle – MBR).
  - Tất cả các nút chứa dữ liệu đều là nút ngọn (ở cùng cấp).
  - Mỗi nút ngọn chứa từ K/2 (hoặc là 2) đến K bản ghi.

Cấu trúc cây R

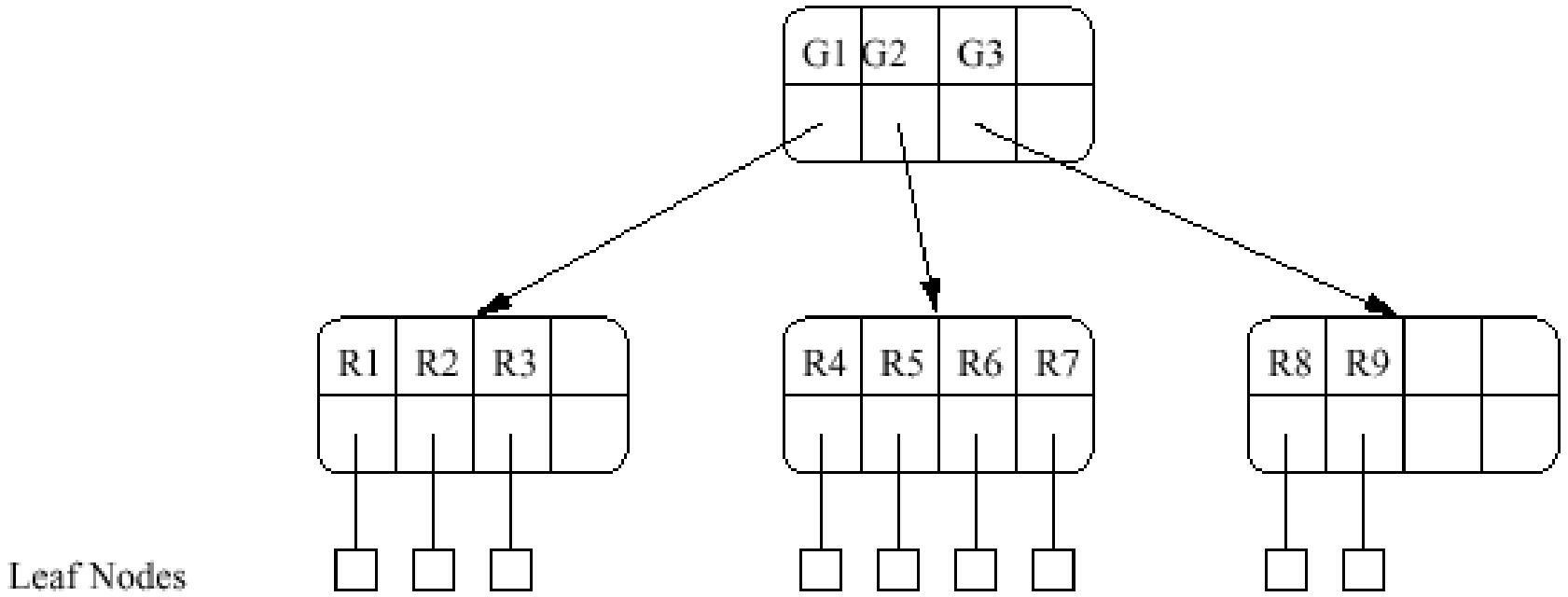
- Cây R chứa 2 loại vùng chữ nhật
  - Vùng "thực" (như trong slide 49): chứa dữ liệu thực, thường là các nút lá của cây R.
  - Vùng nhóm:

là các nút gốc của cây R



Cấu trúc cây R

- Cây R với 4 cấp được xây dựng trên sơ đồ slide 51

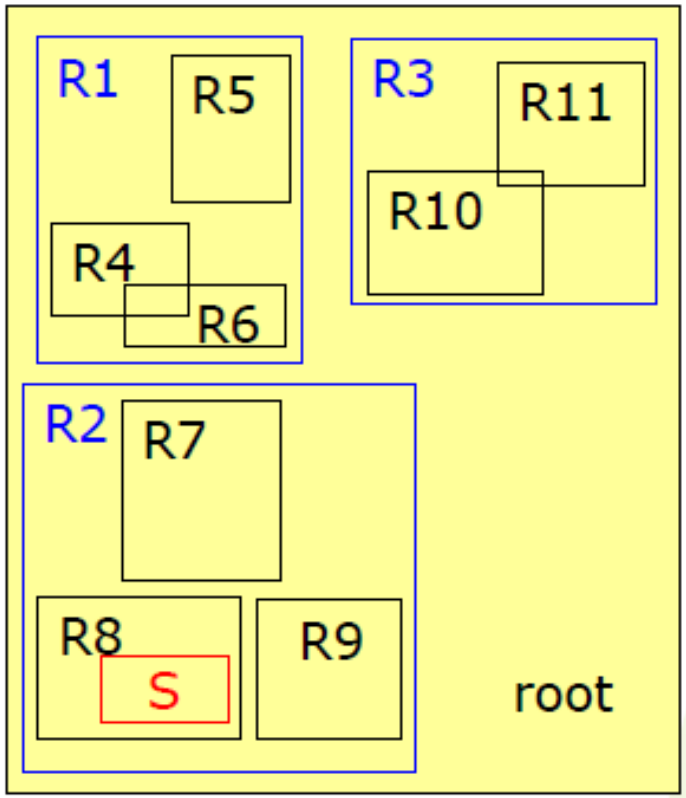


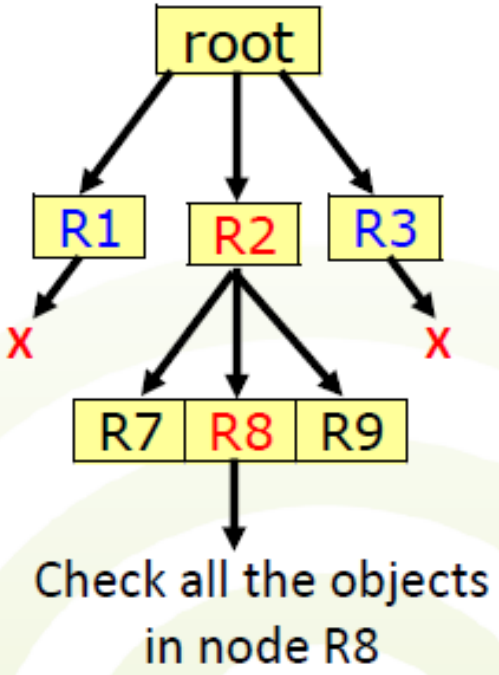
Tìm kiếm dữ liệu trên cây R

- Quá trình tìm kiếm được thực hiện từ gốc
  - Lựa chọn một nhánh để tìm
  - Nếu không thấy dữ liệu trong nhánh này thì chuyển sang nhánh khác.
- Việc lựa chọn nhánh tìm kiếm là ngẫu nhiên

Tìm kiếm trên cây R

- VD: để tìm dữ liệu S, ta cần tìm trong 7 nút.



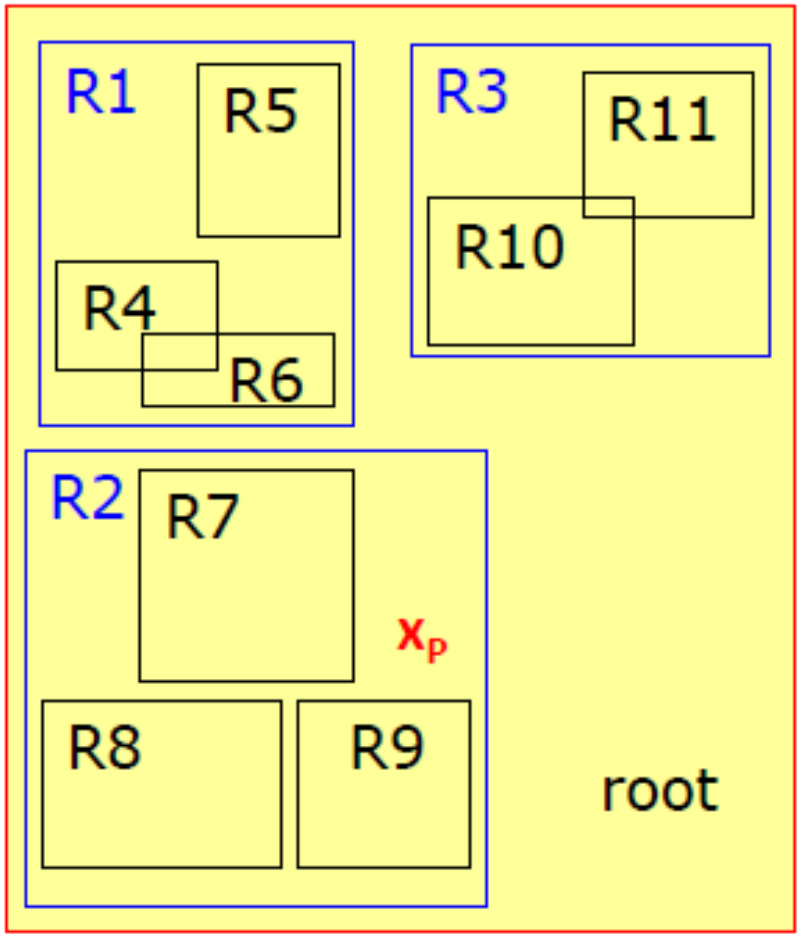


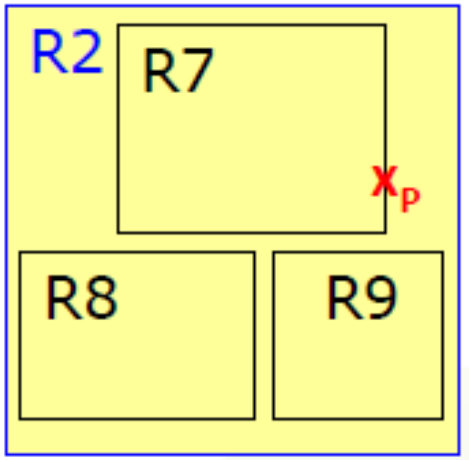
Thêm dữ liệu vào cây R

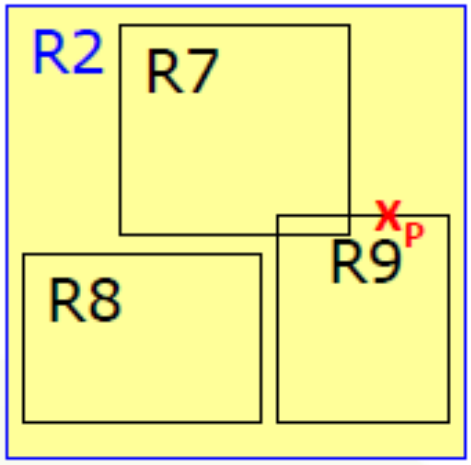
- Lựa chọn nút ngọn tốt nhất để chứa dữ liệu cần thêm.
  - Nút tốt nhất là nút cần phải mở rộng với diện tích nhỏ nhất để chứa dữ liệu đó.
- Dữ liệu sẽ được thêm vào nút tốt nhất đó nếu còn chỗ (số dữ liệu trong nút đó hiện đang < K)
- Nếu số dữ liệu trong nút tốt nhất đó đã đầy, sau khi thêm dữ liệu mới vào ta phải chia đôi nút đó.
- Các nút mới (sau khi chia) sẽ là các nút ngọn.
- Nếu nút gốc đã chứa đủ số nút ngọn thì 2 nút mới chia ra sẽ được ghép vào một nút gốc mới.

Thêm dữ liệu vào cây R

- VD: thêm bản ghi x_P vào cây R.

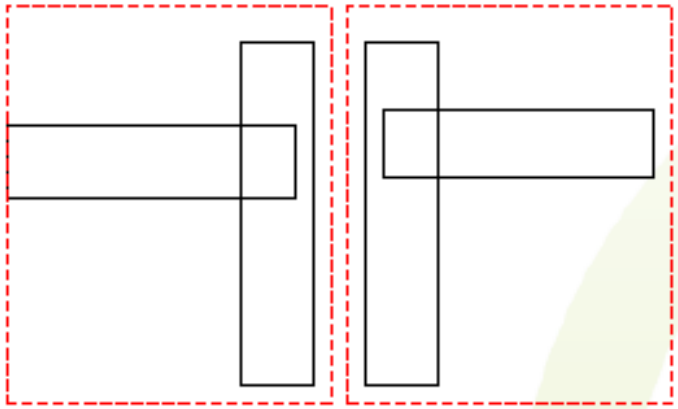




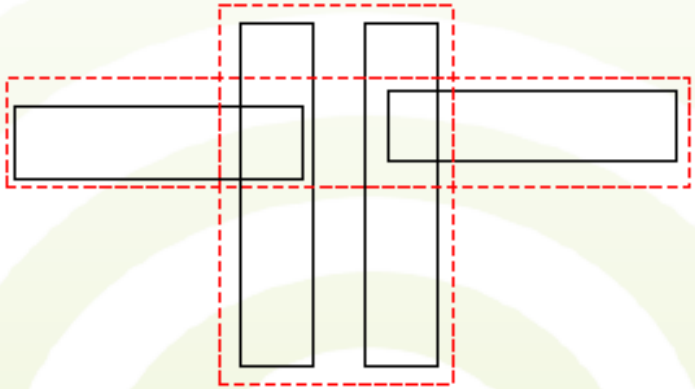


Thêm dữ liệu vào cây R

- Trường hợp phải chia đôi nút, thực hiện nguyên tắc:
  - Tối thiểu hóa diện tích dư thừa
  - Tối thiểu hóa diện tích chồng lấn (overlap)



Bad split

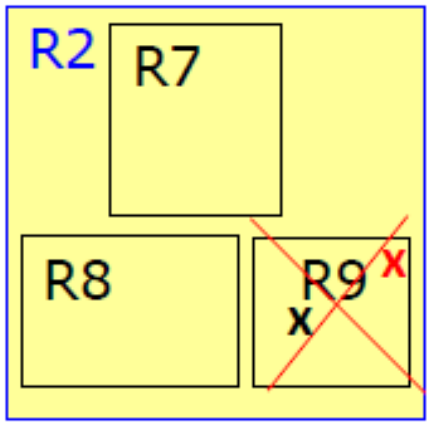


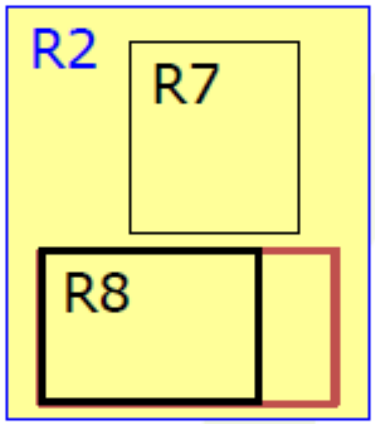
Better Split

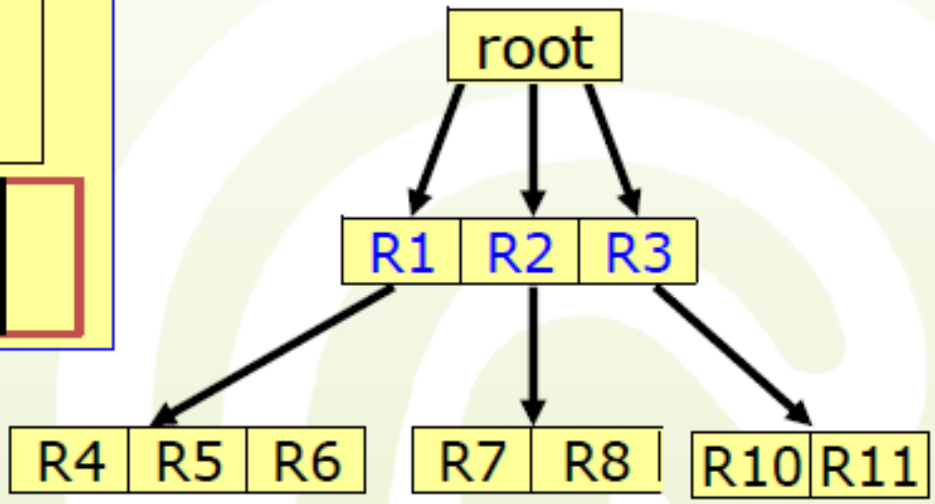
- Quá trình phân chia nút đôi khi rất phức tạp.

Xóa dữ liệu ở cây R

- Tìm nút lá chứa dữ liệu cần xóa
- Xóa dữ liệu tại nút đó
- Nếu sau khi xóa, số bản ghi còn lại trong nút nhỏ hơn yêu cầu tối thiểu, nút này sẽ bị xóa bỏ. Các dữ liệu còn lại sẽ được phân chia lại.



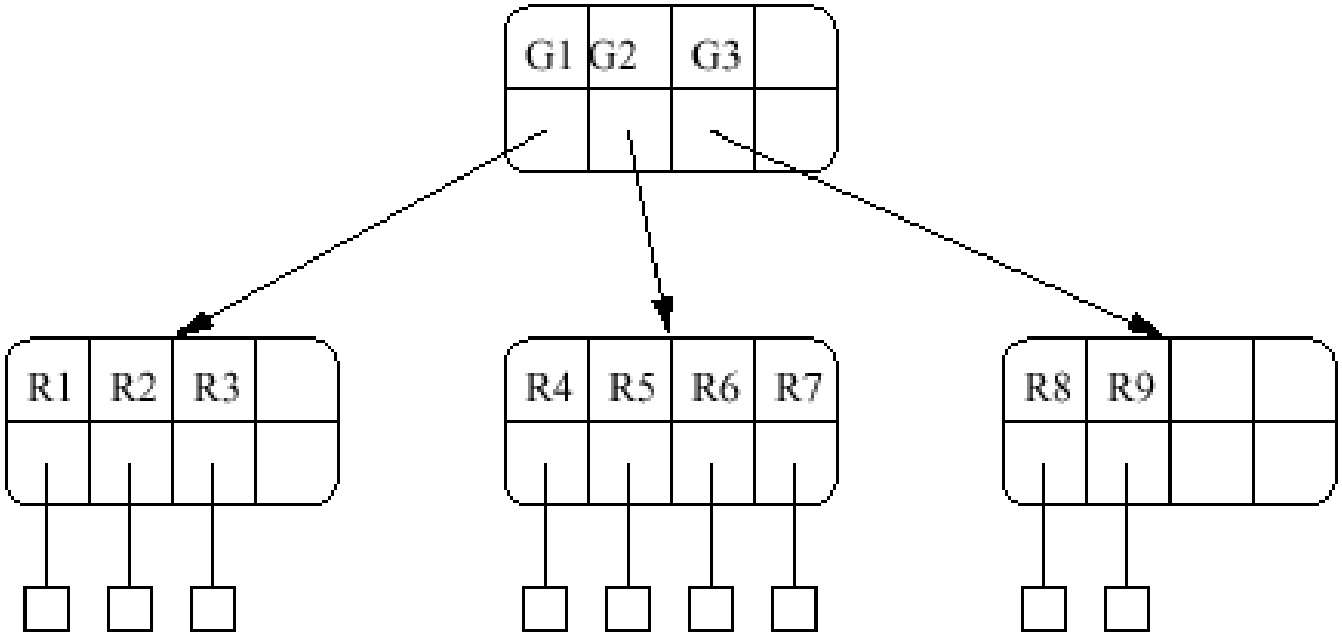




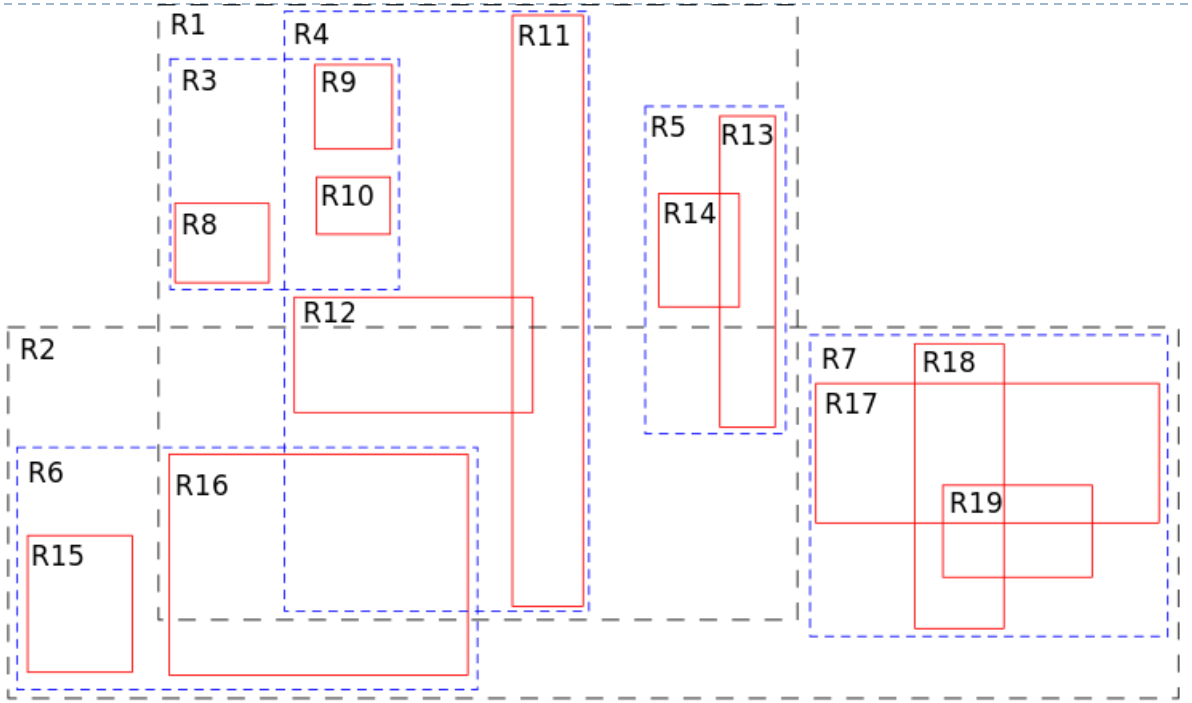
Xóa dữ liệu ở cây R

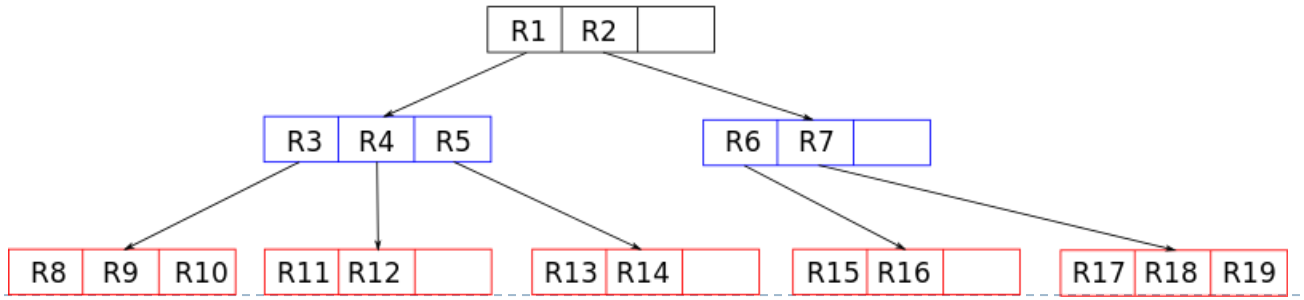
- Để đảm bảo quân số tối thiểu của mỗi nhóm, khi xóa một nút dữ liệu đôi khi dẫn đến việc phân chia lại nhóm.
  - VD: muốn xóa R9.

Group	Rectangles
G1	R1,R2,R3
G2	R4,R6,R7
G3	R5,R8

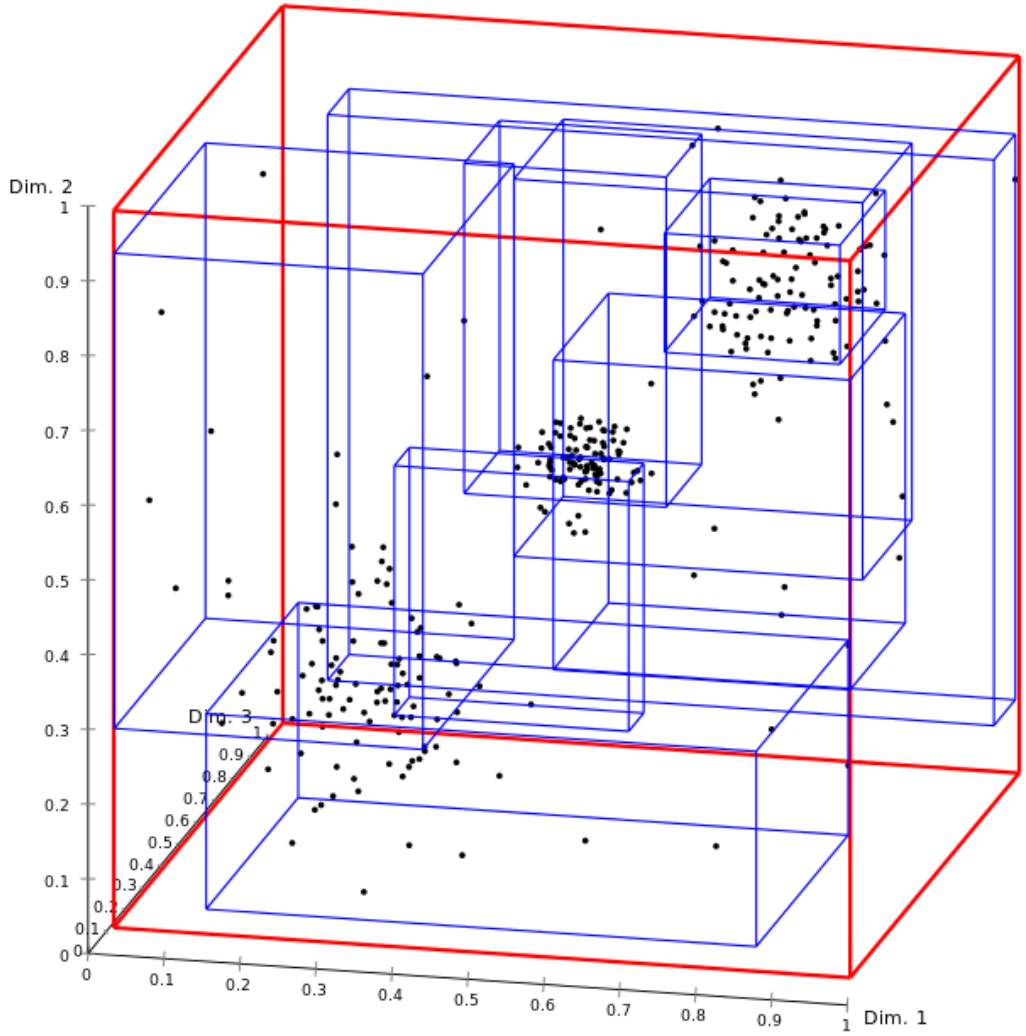


Các ví dụ về cây R





Các ví dụ về cây R



Ví dụ

- Cho các dữ liệu theo cấu trúc vector trọng số như sau

  D_1 = [0.2, 0.1]

  D_2 = [0.6, 0.5]

  D_3 = [0.5, 0.5]

  D_4 = [0.4, 0.4]

  D_5 = [0.5, 0.4]

  D_6 = [0.1, 0.3]

  D_7 = [0.3, 0.4]

- Hãy sử dụng cây R với K=3 để biểu diễn các dữ liệu trên bao gồm các thông tin về tên dữ liệu và vector tương ứng của dữ liệu đó
- Giả sử muốn xóa D_1 thì cấu trúc cây thay đổi thế nào?