* Việc chèn một nút vào cây n-nút đỏ-đen có thể được thực hiện trong thời gian O (lgn). Chúng ta sử dụng một phiên bản sửa đổi nhỏ của thủ tục TREE-INSERT (Phần 12.3) để chèn nút z vào cây T như thể nó là một cây tìm kiếm nhị phân thông thường, sau đó chúng ta có màu z đỏ. Để đảm bảo rằng các thuộc tính màu đỏ-đen được bảo toàn, chúng ta gọi một thủ tục phụ trợ RB-INSERT-FIXUP để đổi màu các nút và thực hiện các phép quay. Các cuộc gọi RB-INSERT (T, z) chèn nút z, mà quan trọng của nó là eld được giả định đã được lấp vào, vào cây màu đỏ-đen T

<hình>

* Có bốn sự khác biệt giữa các thủ tục TREE-INSERT và RBINSERT. Thứ nhất, tất cả các thể hiện của NIL trong TREE-INSERT được thay thế bởi nil [T]. Thứ hai, chúng ta đặt left [z] và right [z] tonil [T] trong dòng 14-15 của RB-INSERT, để duy trì cấu trúc cây thích hợp. Thứ ba, chúng ta có màu z đỏ trong dòng 16. Thứ tư, bởi vì màu z đỏ có thể gây ra sự vi phạm một trong các thuộc tính màu đỏ-đen, chúng ta gọi RB-INSERT-FIXUP (T, z) trong dòng 17 của RB-INSERT để khôi phục lại các thuộc tính màu đỏ-đen

<code>

* Để hiểu RB-INSERT-FIXUP hoạt động như thế nào, chúng ta sẽ phá vỡ sự kiểm tra của chúng ta về mã thành ba bước chính. Trước tiên, chúng ta sẽ xác định những gì vi phạm các thuộc tính màu đỏ-đen được giới thiệu trong RB-INSERT khi nút z được chèn vào và có màu đỏ. Thứ hai, chúng ta sẽ xem xét mục tiêu chung của vòng lặp while trong các dòng 1-15. Cuối cùng, chúng ta sẽ khám phá mỗi ba trường hợp1 trong đó vòng lặp trong khi bị hỏng và xem cách họ đạt được mục tiêu. Hình 13.4 cho thấy cách RB-INSERT-FIXUP hoạt động như thế nào trên cây mẫu màu đỏ-đen. Tính chất màu đỏ-đen nào có thể bị vi phạm khi cuộc gọi đến RB-INSERTFIXUP? Bất động sản 1 chắc chắn tiếp tục giữ, cũng như tài sản 3, bởi vì cả hai đứa trẻ của nút đỏ được chèn vào mới là không có trọng điểm [T]. Property 5, nói rằng số nút màu đen là như nhau trên mỗi đường đi từ nút agiven, là satisfedaswell, bởi vì node z thay thế sentinel (màu đen), vànode z isredwith sentinel children. Do đó, tài sản duy nhất có thể bị vi phạm là thuộc tính 2, yêu cầu gốc phải là màu đen và thuộc tính 4 cho biết nút màu đỏ không thể có con đỏ. Cả hai vi phạm có thể xảy ra do z màu đỏ. Bất động sản 2 bị vi phạm nếu z là gốc, và tài sản 4 bị vi phạm nếu cha của z là màu đỏ. Hình 13.4 (a) cho thấy một sự vi phạm thuộc tính 4 sau khi nút z đã được chèn. Vòng lặp while trong các dòng 1-15 duy trì hệ số bất biến ba phần sau đây:
* <hinh>
* Hình 13.4 Hoạt động của RB-INSERT-FIXUP. (a) Một nút z sau khi chèn. Vì z và cha mẹ p [z] cả hai đều màu đỏ, một sự vi phạm thuộc tính 4 xảy ra. Vì chú y của z là màu đỏ, trường hợp 1 trong mã có thể được áp dụng. Các nút được đổi màu và con trỏ z được di chuyển lên cây, kết quả là cây thể hiện7 trong (b). Một lần nữa, z và cha mẹ nó đều đỏ, nhưng chú z của y là màu đen. Vì z là con phải của p [z], trường hợp 2 có thể được áp dụng. Vòng trái được thực hiện, và cây mà kết quả được hiển thị trong (c). Nowz là con còn lại của cha mẹ, và trường hợp 3 có thể được áp dụng. Một vòng quay bên phải cho phép cây trong (d), là cây hợp pháp màu đỏ-đen.
* Vào đầu mỗi vòng lặp của vòng lặp, a. Nút z có màu đỏ. b. Nếu p [z] là gốc, thì p [z] là màu đen. c. Nếu có sự vi phạm các thuộc tính màu đỏ-đen, có nhiều nhất là một hành vi vi phạm, và cả hai thuộc tính tài sản 2. 4. Nếu isaviolation của tài sản 2, nó xảy ra bởi vì z là gốc và màu đỏ. Nếu có sự vi phạm thuộc tính 4, nó xảy ra vì cả z và p [z] đều có màu đỏ.
* Phần (c), xử lý các vi phạm thuộc tính màu đỏ-đen, là trọng tâm hơn cho thấy rằng RB-INSERT-FIXUP phục hồi các thuộc tính màu đỏ-đen so với các phần (a) và (b), chúng tôi sử dụng để hiểu tình huống trong mã. Bởi vì chúng ta sẽ tập trung vào nút z và các nút gần nó trong cây, rất hữu ích để biết từ (a) z rằng có màu đỏ. Weshall sử dụng một phần (b) để cho thấy rằng nút p [p [z]] tồn tại khi chúng ta tham khảo nó trong các dòng 2, 3, 7, 8, 13 và 14. Nhớ lại rằng chúng ta cần chứng minh rằng bất biến vòng lặp là đúng trước đến lần lặp đầu tiên của vòng lặp, mỗi lần lặp giữ lại bất biến vòng lặp, và rằng bất biến vòng lặp sẽ cho chúng ta một thuộc tính hữu ích tại kết cuối vòng lặp. Chúng ta bắt đầu với các đối số khởi tạo và chấm dứt. Sau đó, khi chúng ta kiểm tra cơ thể của vòng lặp hoạt động như thế nào chi tiết hơn, chúng ta sẽ lập luận rằng vòng lặp duy trì bất biến theo từng lần lặp. Trên đường đi, chúng tôi cũng sẽ chứng minh rằng có hai kết quả có thể có của mỗi lần lặp của vòng lặp: con trỏ z di chuyển lên cây hoặc một số vòng quay được thực hiện và vòng kết thúc. Khởi tạo: Trước khi lặp lại vòng lặp đầu tiên của vòng lặp, chúng ta bắt đầu với một cây màu đỏ-đen mà không vi phạm, và chúng ta thêm một nút màu đỏ z. Chúng tôi chỉ ra rằng mỗi phần của bất biến được giữ tại thời điểm RB-INSERT-FIXUP được gọi là: a. Khi RB-INSERT-FIXUP được gọi, z là nút màu đỏ đã được thêm vào. b. Nếu p [z] là gốc, sau đó p [z] bắt đầu màu đen và không thay đổi trước khi cuộc gọi của RB-INSERT-FIXUP. c. Chúng ta đã thấy các thuộc tính 1, 3, và 5 giữ khi RB-INSERTFIXUP được gọi. Nếu đó là sự kết hợp của thuộc tính 2, thì nó phải là nút nút bên trong zed, đó là nút nội bộ duy nhất trong cây. Bởi vì cha mẹ và cả hai đứa trẻ của z arethe sentinel, vốn là màu đen, cũng không hề có sự gián đoạn của tài sản 4. Như vậy, vi phạm tài sản 2 là vi phạm duy nhất các thuộc tính redblack trong toàn bộ cây. Nếu có sự vi phạm thuộc tính 4 thì vì con của nút z là các trọng điểm đen và cây không có các vi phạm khác trước khi được thêm vào, vi phạm phải là vì cả z và p [z] đều màu đỏ. Hơn nữa, không có vi phạm khác của tài sản màu đỏ-đen.
* Chấm dứt: Khi vòng kết thúc, nó làm như vậy bởi vì p [z] là màu đen. (Nếu z là gốc, thì p [z] là nốt sentinel [T], màu đen) Vì vậy, không có sự vi phạm thuộc tính 4 tại chấm dứt vòng lặp. Bởi bất biến vòng lặp, tài sản duy nhất có thể không giữ lại được là thuộc tính 2. Line16 khôi phục lại thuộc tính này, quá, để khi RB-INSERT-FIXUP kết thúc, tất cả các thuộc tính màu đỏ-đen giữ.
* Bảo trì: Có sáu trường hợp cần xem xét trong vòng lặp while, nhưng ba trong số đó đối xứng với ba vòng còn lại, phụ thuộc vào việc cha của z [z] là đứa trẻ còn lại hay con phải của ông z của [z [z ], được xác định trong dòng 2. Chúng ta chỉ đưa ra mã cho tình huống trong đó p [z] là một đứa trẻ còn lại. Các nút p [p [z]] tồn tại, vì bởi một phần (b) của vòng lặp bất biến, nếu p [z] là gốc, sau đó p [z] là màu đen. Vì chúng ta nhập một vòng lặp lặp lại chỉ khi p [z] có màu đỏ, chúng ta biết rằng p [z] không thể là gốc. Do đó, p [p [z]] tồn tại. Case1 được phân biệt từ cặp 2 và 3 của cặp vợ chồng z'sparent, hoặc "chú". Dòng 3 làm cho chú ý đến z của z chú ý [p [p [z]], và một bài kiểm tra được thực hiện trong dòng 4. Nếu y là màu đỏ, sau đó trường hợp 1 được thực hiện. Nếu không, điều khiển chuyển sang các trường hợp 2 và 3. Trong cả ba trường hợp, ông bà z của [z]] là màu đen, vì cha / mẹ của nó [z] màu đỏ, và thuộc tính 4 chỉ bị vi phạm giữa z và p [z] .
* Trường hợp 1: z chú y là màu đỏ Hình 13.5 cho biết tình huống của trường hợp 1 (dòng 5-8). Trường hợp 1 được thực hiện khi cả hai p [z] andy là màu đỏ. Vì p [z] là màu đen, chúng ta có thể tô màu cả p [z] và y đen, do đó giải quyết vấn đề z và p [z] cả hai đều màu đỏ, và màu [p [z]] màu đỏ, do đó duy trì tài sản 5. Chúng ta lặp lại vòng lặp với p [p [z]] như nút z mới. Con trỏ z di chuyển lên hai cấp độ trong cây. Bây giờ chúng ta chỉ ra rằng trường hợp 1 duy trì vòng lặp bất biến vào lúc bắt đầu lặp lại tiếp theo. Chúng ta sử dụng z để chỉ nút z trong lần lặp hiện tại, và z = p [p [z]] để biểu thị nút z tại bài kiểm tra trong dòng 1 trong lần lặp tiếp theo. a. Bởi vì màu lặp lặp lại này [p [z]] màu đỏ, node z được định vị tại đầu của phép lặp kế tiếp. b. Các nút p [z] isp [p [p [z]]] trong lần lặp lại này, và màu sắc của nút này không thay đổi. Nếu nút này là gốc, nó đã là màu đen trước khi lặp lại này, và nó vẫn còn màu đen ở đầu của lặp tiếp theo. c. Chúng ta đã lập luận rằng trường hợp 1 duy trì tài sản 5, và rõ ràng không đưa ra một sự vi phạm thuộc tính 1 hoặc 3. Nếu nút z là gốc ở đầu của lần lặp tiếp theo, thì trường hợp 1 đã điều chỉnh sự vi phạm đơn lẻ của tài sản 4 trong lần lặp lại này. Vì z là màu đỏ và nó là gốc, thuộc tính 2 trở thành một trong những vi phạm duy nhất và vi phạm này là do z.
* <hình>
* Hình 13.5 Trường hợp 1 của quá trình thực hiện RB-INSERT. Property4isviolated, vì z anditsparent p [z] có cả màu đỏ. Cùng một hành động được thực hiện xem (a) z là con đúng hay (b) z là con còn lại. Mỗi subtrees α, β, γ, δ, andε có một gốc rễ đen, và mỗi một có cùng một chiều cao đen. Mã cho trường hợp 1 thay đổi màu sắc của một số nút, bảo vệ tài sản 5: tất cả các đường dẫn xuống từ một nút đến một lá có cùng một số người da đen. Vòng lặp while vẫn tiếp tục với ông bà p z [z]] của nút z với z. Bất kỳ sự vi phạm nào đối với tài sản 4 bây giờ chỉ có thể xảy ra giữa z mới, màu đỏ, và phụ huynh của nó, nếu nó là màu đỏ nữa. Nếu nút z không phải là gốc ở đầu của lần lặp tiếp theo, thì trường hợp 1 đã không tạo ra sự vi phạm tài sản 2. Trường hợp 1 sửa chữa vi phạm đơn lẻ của thuộc tính 4 đã tồn tại vào lúc bắt đầu lặp lại này. Nó sau đó được thực hiện với z màu đỏ và trái p [z] một mình. Nếu p [z] là màu đen, không có vi phạm tài sản 4. Nếu p [z] bị nhà, màu z redcreatedoneviolation của tài sản 4between z và p [z]
* Trường hợp 2: z chú của y là màu đen và z là một con phải Trường hợp 3: z của chú y là màu đen và z là một con còn lại Trong trường hợp 2 và 3, màu sắc của z z chú y là màu đen. Hai trường hợp được phân biệt bằng cách z là một đứa trẻ phải hay trái của p [z]. Dòng 10-11 là trường hợp 2, được thể hiện trong hình 13.6 cùng với trường hợp 3. Trong trường hợp 2, nút z là một con phải của cha mẹ nó. Ngay lập tức chúng ta sử dụng một phép quay trái để biến tình huống thành trường hợp 3 (dòng 12-14), trong đó nút z là một con còn lại. Bởi vì cả hai z và p [z] có màu đỏ, sự quay không ảnh hưởng đến chiều cao đen của các nút và thuộc tính 5. Cho dù chúng ta nhập trường 3 trực tiếp hay thông qua trường hợp 2, z chú y của màu đen, vì nếu không chúng ta sẽ thực hiện trường hợp 1 Ngoài ra, nút p [p [z]] tồn tại, vì chúng ta đã lập luận rằng nút này tồn tại vào thời điểm đó
* <hinh>
* Hình 13.6 Các trường hợp 2 và 3 của thủ tục RB-INSERT. Như trong trường hợp 1, tài sản 4 bị vi phạm trong trường hợp 2 hoặc trường hợp 3 vì z và cha mẹ p [z] đều có màu đỏ. Mỗi subtrees α, β, γ, andδ có gốc đen (α, β, andγ từ thuộc tính 4, và δ bởi vì nếu không chúng ta sẽ là trường hợp 1), và mỗi một có cùng độ cao đen. Trường hợp 2 được biến đổi thành trường hợp 3 bởi một vòng quay trái, bảo vệ tài sản 5: tất cả các đường đi xuống từ nút đến lá có cùng số lượng người da đen. Trường hợp 3 gây ra một số thay đổi màu sắc và xoay một chiều, cũng bảo vệ tài sản 5. Các whileloop sau đó chấm dứt, bởi vì tài sản 4 được thỏa mãn: không còn hai nút màu đỏ trong một hàng. các dòng 2 và 3 đã được thực hiện, và sau khi di chuyển z lên một cấp trong dòng 10 và sau đó xuống một mức trong dòng 11, thì danh tính của p [p [z]] vẫn không thay đổi. Trong trường hợp 3, chúng ta thực hiện một số thay đổi màu sắc và quay phải, bảo vệ tài sản 5, và sau đó, vì chúng ta không còn có hai nút màu đỏ trong một hàng, chúng ta đã làm xong. Thân của vòng lặp while không được thực hiện lần khác, vì p [z] bây giờ là màu đen. Bây giờ chúng ta chỉ ra rằng trường hợp 2 và 3 duy trì bất biến vòng lặp. (Như chúng tôi đã chỉ ra, p [z] sẽ trở thành màu đen khi thử nghiệm tiếp theo ở dòng 1, và thân vòng lặp sẽ không thực hiện lại.) a. Trường hợp 2 làm cho z điểm đến p [z], màu đỏ. Không có sự thay đổi nào khác về z hoặc màu sắc của nó xảy ra trong trường hợp 2 và 3. b. Trường hợp 3 làm cho p [z] màu đen, do đó nếu p [z] là gốc ở đầu của lần lặp tiếp theo, nó là màu đen. c. Như trong trường hợp 1, các thuộc tính 1, 3 và 5 được duy trì trong các trường hợp 2 và 3. Vì nút z không phải là gốc trong trường hợp 2 và 3, chúng ta biết rằng không có vi phạm tài sản 2. Trường hợp 2 và 3 không giới thiệu sự vi phạm thuộc tính 2, vì nút duy nhất được tạo thành màu đỏ sẽ trở thành một đứa trẻ của một nút màu đen khi xoay vòng trong trường hợp 3. Trường hợp 2 và 3 chính xác sự vi phạm tài sản 4 và chúng không đưa ra một hành vi vi phạm khác. Sau khi chỉ ra rằng mỗi lần lặp của vòng lặp duy trì bất biến, chúng tôi đã chỉ ra rằng RB-INSERT-FIXUP khôi phục lại các thuộc tính màu đỏ-đen một cách chính xác
* Phân tích Thời gian chạy của RB-INSERT là gì? Vì chiều cao của cây màu đỏ-đen trên n nút là O (lgn), các dòng 1-16 của RB-INSERT mất O (lgn) thời gian. Trong RB-INSERTFIXUP, vòng lặp while chỉ lặp lại khi trường hợp 1 được thực hiện, và sau đó con trỏ z di chuyển hai cấp lên cây. Tổng số lần vòng lặp while có thể được thực hiện là O (lgn). Vì vậy, RB-INSERT mất tổng cộng O (lgn) thời gian. Thật thú vị, nó không bao giờ thực hiện nhiều hơn hai vòng quay, kể từ khi vòng lặp while chấm dứt nếu trường hợp 2 hoặc trường hợp 3 được thực hiện. Bài tập 13.3-1 Trong dòng 16 của RB-INSERT, chúng ta thiết lập màu sắc của nút mới chèn z thành màu đỏ. Lưu ý rằng nếu wehad chọn màu z của z thành màu đen, thì thuộc tính 4 của cây ared-black sẽ không bị vi phạm. Tại sao chúng ta không chọn để thiết lập màu của z thành màu đen? 13.3-2 Cho thấy các cây màu đỏ-đen kết quả sau khi chèn các con chuột 41,38,31, 12,19,8 vào một cây đỏ-đen ban đầu trống rỗng. 13.3-3 Giả sử độ cao của mỗi subtrees α, β, γ, δ, ε trong các hình 13.5 và 13.6 là k. Nhãn mỗi nút trong mỗi đường với độ cao đen để xác minh rằng thuộc tính 5 được bảo toàn bởi sự chuyển đổi được chỉ định. 13.3-4 Giáo sư Teach lo ngại rằng RB-INSERT-FIXUP có thể thiết lập màu [nil [T]] thành RED, trong trường hợp đó phép thử trong dòng 1 sẽ không gây ra vòng lặp chấm dứt khi z là gốc. Cho thấy mối quan tâm của giáo sư là không có căn cứ bằng cách lập luận rằng RBINSERT-FIXUP không bao giờ đặt màu [nil [T]] sang RED. 13.3-5 Xem xét một cây màu đỏ-đen được hình thành bằng cách chèn n nút với RB-INSERT. Cho rằng nếu n> 1, cây có ít nhất một nút màu đỏ. 13.3-6 Đề xuất làm thế nào để thực hiện RB-INSERT hiệu quả nếu biểu diễn cho cây redblack không bao gồm lưu trữ cho con trỏ phụ huynh.

DELETE

* Có ba khác biệt giữa các thủ tục TREE-DELETE và RBDELETE. Đầu tiên, tất cả các tham chiếu đến NIL trong TREE-DELETE được thay thế bằng các tham chiếu tothesentinel nil [T] trongRB-DELETE. Thứ hai, bài kiểm tra cho biết x là NILinline7 của TREE-DELETE được loại bỏ, và phân công p [x] ← p [y] được thực hiện vô điều kiện trong dòng 7 của RB-DELETE. Vì vậy, nếu x là số không xác định [T], con trỏ gốc của nó chỉ tới cha mẹ của nút được ghép nối y. Thứ ba, một cuộc gọi đến RBDELETE-FIXUP được thực hiện trong các dòng 16-17 nếu y là màu đen. Nếu y là màu đỏ, các thuộc tính màu đỏ-đen vẫn giữ khi y được ghép nối, vì những lý do sau:

• không có độ cao màu đen trong cây đã thay đổi.

• không có nút màu đỏ đã được làm liền kề, và

• vì y không thể là gốc rễ nếu nó đỏ, gốc rễ vẫn còn màu đen.

* Nút x được truyền đến RB-DELETE-FIXUP là một trong hai nút: hoặc là nút duy nhất của con y trước khi y được nối ra nếu y có con không phải là trọng điểm [T], hoặc, nếu y không có trẻ em, x là sentinel nil [T]. Trong trường hợp thứ hai, phép gán vô điều kiện trong dòng 7 bảo đảm rằng cha mẹ của x bây giờ là nút mà trước đó là cha của y, cho dù x là một nút nội tại có chìa khóa hoặc không có giá trị [T]. Bây giờ chúng ta có thể kiểm tra làm thế nào các thủ tục RB-DELETE-FIXUP phục hồi các thuộc tính redblack đến cây tìm kiếm
* <code>
* Nếu nút spliced-out y trong RB-DELETE là màu đen, có thể xảy ra ba vấn đề. Trước tiên, nếu y đã là gốc và một con đỏ của y trở thành gốc mới, chúng ta đã vi phạm thuộc tính 2. Thứ hai, nếu cả hai x và p [y] (mà bây giờ cũng là p [x]) đều màu đỏ, thì chúng ta đã vi phạm tài sản 4. Thứ ba, sự loại bỏ của y gây ra bất kỳ đường dẫn mà trước đây chứa y để có một nút đen ít hơn. Như vậy, tài sản 5 bây giờ bị vi phạm bởi bất kỳ tổ tiên của y trong cây. Chúng tôi có thể khắc phục vấn đề này bằng cách nói nút x có một màu đen "thêm". Nghĩa là, ifweadd 1 với đếm các nút màu đen trên bất kỳ đường dẫn có chứa x, sau đó theo giải thích này, thuộc tính 5 giữ. Khi chúng ta chia ra nút đen y, chúng ta "đẩy" bóng đen của nó lên con của nó. Vấn đề là ở chỗ nút x không có màu đỏ và màu đen, do đó vi phạm tài sản 1. Thay vào đó, nút x là "double black" hoặc "red-and-black" và nó đóng góp 2 hoặc 1 lần tương ứng vào đếm các nút màu đen trên các đường dẫn có chứa x. Thuộc tính Thecolor của x sẽ vẫn là RED (nếu x là màu đỏ và đen) hoặc BLACK (nếu x là màu đen gấp đôi). Nói cách khác, màu đen thêm vào một nút được lặp lại trong x của trỏ đến nút chứ không phải là thuộc tính màu
* Thủ tục RB-DELETE-FIXUP khôi phục thuộc tính 1, 2, và 4. Các bài tập 13.4-1 và 13.4-2 yêu cầu bạn chỉ ra rằng thủ tục khôi phục các thuộc tính 2 và 4, và do đó trong phần còn lại của phần này, chúng ta sẽ tập trung vào tài sản 1. Mục tiêu của vòng lặp while trong các dòng 1-22 là di chuyển cây màu đen lên cây cho đến khi 1. điểm x đến một nút màu đỏ và đen, trong trường hợp chúng ta tô màu x (đơn lẻ) màu đen trong dòng 23 , 2. x điểm đến gốc, trong trường hợp màu đen thêm có thể được chỉ đơn giản là "loại bỏ", hoặc 3. có thể quay và rã phù hợp có thể được thực hiện.
* Trong vòng lặp while, x luôn luôn chỉ đến một nút đen nonroot gấp đôi. Chúng ta xác định trong dòng 2 liệu x là con còn lại hay con phải của cha / mẹ của nó [x]. (Chúng ta đã đưa ra mã cho tình huống trong đó x là một con còn lại, tình huống trong đó x là một đường con phải 22-đối xứng) Chúng ta duy trì một con trỏ w với x của x. Bởi vì nút x là màu đen gấp đôi, nút w không thể là nil [T]; nếu không, số lượng người da đen trên đường đi từ [x] đến (w) đen lá sẽ nhỏ hơn con số trên con đường từ [x] tox.
* Các trường hợp2 trong mã được minh họa trong Hình 13.7. Trước khi xem xét từng trường hợp cụ thể, chúng ta hãy xem xét tổng quát hơn về cách chúng ta có thể xác minh rằng sự chuyển đổi trong mỗi trường hợp bảo vệ tài sản 5. Ý tưởng quan trọng là trong mỗi trường hợp số nút đen (bao gồm cả màu đen thêm x) từ (và bao gồm) gốc của cây con được hiển thị cho mỗi subtrees α, β, ..., ζ được bảo tồn bởi sự chuyển đổi. Như vậy, nếu tài sản 5 giữ trước khi chuyển đổi, nó tiếp tục giữ sau đó. Ví dụ, trong hình 13.7 (a), minh hoạ trường hợp 1, số nút màu đen từ gốc đến cả hai subtree α hoặc β là 3, cả trước và sau khi chuyển đổi. Tương tự, số nút màu đen từ gốc tới bất kỳ γ, δ, ε, andζ là 2, cả trước và sau khi chuyển đổi. Trong hình 13.7 (b), đếm phải bao hàm giá trị c của thuộc tính màu của gốc của cây con được hiển thị, có thể là Đỏ hoặc Đen. Nếu chúng ta đếm đếm (RED) = 0 và đếm (BLACK) = 1, thì số
* <hình>
* Hình 13.7 Các trường hợp trong vòng lặp while của thủ tục RB-DELETE-FIXUP. Các nút đậm có các thuộc tính màu sắc BLACK, các nút bóng mờ nhiều có các thuộc tính màu đỏ RED, và các nút được đánh bóng nhẹ có các thuộc tính màu được biểu diễn bởi c và c, có thể là REDor BLACK. Các chữ cái α, β, ..., ζ đại diện cho subtrees tùy ý. Trong mỗi trường hợp, cấu hình bên trái được chuyển đổi thành cấu hình bên phải bằng cách thay đổi một số màu sắc và / hoặc thực hiện quay. Bất kỳ nút nào được chỉ bởi x có một màu đen thêm và có thể tăng gấp đôi màu đen hoặc màu đỏ và đen. Trường hợp duy nhất gây ra vòng lặp lặp lại là trường hợp 2. (a) Trường hợp 1 được chuyển đổi sang trường hợp 2, 3, hoặc 4 bằng cách trao đổi màu sắc của các nút B và D và thực hiện quay trái. (b) Trong trường hợp 2, màu đen thêm được biểu diễn bởi con trỏ x được di chuyển lên cây bằng cách tô màu nút D màu đỏ và thiết lập x để trỏ đến nút B. Nếu chúng ta nhập trường hợp 2 thông qua trường hợp 1, vòng lặp while kết thúc bởi vì nút x là màu đỏ và đen, và do đó giá trị c của thuộc tính màu của nó là RED. (c) Trường hợp 3 được chuyển đổi sang trường hợp 4 bằng cách trao đổi các màu CO và D và hiệu quả hơn. (d) Incase4, theextrablackrepresented bởi x có thể được gỡ bỏ bằng cách thay đổi một số màu sắc và thực hiện một trái quay (mà không vi phạm các thuộc tính màu đỏ đen), và vòng lặp chấm dứt.
* ber của các nút màu đen từ gốc đến α là 2 count (c), cả trước và sau khi chuyển đổi. Trong trường hợp này, sau khi chuyển đổi, nút mới x có thuộc tính màu c, nhưng nút này thực sự là màu đỏ và đen (nếu c = RED) hoặc đôi màu đen (nếu c = màu đen). Các trường hợp khác có thể được kiểm chứng tương tự (xem Bài tập 13.4-5). Trường hợp 1: x của anh chị em w là màu đỏ Case1 (dòng 5-8ofRB-DELETE-FIXUP và Hình 13.7 (a)) xảy ra khi nút w, anh chị em của nút x, màu đỏ. Vì w phải có các con đen, chúng ta có thể chuyển đổi màu của w và p [x] và sau đó thực hiện phép quay trái trên p [x] mà không vi phạm bất kỳ thuộc tính màu đỏ-đen nào. Các anh chị em mới của x, là một trong những đứa trẻ của w trước khi quay, bây giờ là màu đen, và do đó chúng ta đã chuyển trường hợp 1 thành trường hợp 2, 3, hoặc 4. Các trường hợp 2, 3 và 4 xảy ra khi node w là màu đen ; chúng được phân biệt bằng màu sắc của trẻ em w.
* Trường hợp 2: x của anh chị em w là màu đen, và cả hai đứa trẻ của w là màu đen Trong trường hợp 2 (dòng 10-11 của RB-DELETE-FIXUP và Hình 13.7 (b)), cả hai đứa trẻ của w đều có màu đen. Vì w cũng là màu đen, chúng ta lấy một màu đen ra cả x và w, để lại x chỉ với một màu đen và để lại màu đỏ. Để bù cho việc loại bỏ một màu đen khỏi x và w, chúng ta muốn thêm một màu đen thêm vào p [x], mà ban đầu là màu đỏ hoặc đen. Chúng ta làm như vậy bằng cách lặp lại vòng lặp với p [x] như nút mới x. Chú ý rằng nếu chúng ta nhập trường hợp 2 thông qua trường hợp 1, nút x mới là màu đỏ và đen, vì màu nguyên bản [x] là màu đỏ. Do đó, giá trị c của thuộc tính màu của nút mới x là RED, và vòng kết thúc khi nó kiểm tra điều kiện vòng lặp. Nút mới x sau đó được tô màu (đơn lẻ) màu đen ở dòng 23.
* Trường hợp 3: x của anh chị em là màu đen, màu đen của con còn lại là màu đỏ, và con bên phải của màu đen là màu đen Trường hợp 3 (các đường 13-16 và hình 13.7 (c)) xảy ra khi w là màu đen, màu đỏ trái của nó, và bên phải của nó con là màu đen. Chúng ta có thể chuyển đổi màu sắc của w và trái trái của nó sang trái [w] và sau đó thực hiện một vòng quay phải trên w mà không vi phạm bất kỳ thuộc tính màu đỏ-đen. Các anh chị em mới w của x bây giờ là một nút màu đen với một đứa trẻ màu đỏ phải, và do đó chúng tôi đã chuyển trường hợp 3 thành trường hợp 4.
* Trường hợp 4: x của anh chị em w là màu đen, và con phải của w là màu đỏ Trường hợp 4 (dòng 17-21 và hình 13.7 (d)) xảy ra khi nút x của anh chị em x là màu đen và màu đen của con là màu đỏ. Bằng cách thực hiện một số thay đổi màu sắc và thực hiện quay trái trên p [x], chúng ta có thể loại bỏ màu đen thêm vào x, làm cho nó màu đen đơn lẻ, mà không vi phạm bất kỳ thuộc tính màu đỏ-đen. Đặt x là gốc gây ra vòng lặp while kết thúc khi nó kiểm tra điều kiện vòng lặp.