CMPT307: Red-Black Trees

Week 3-2

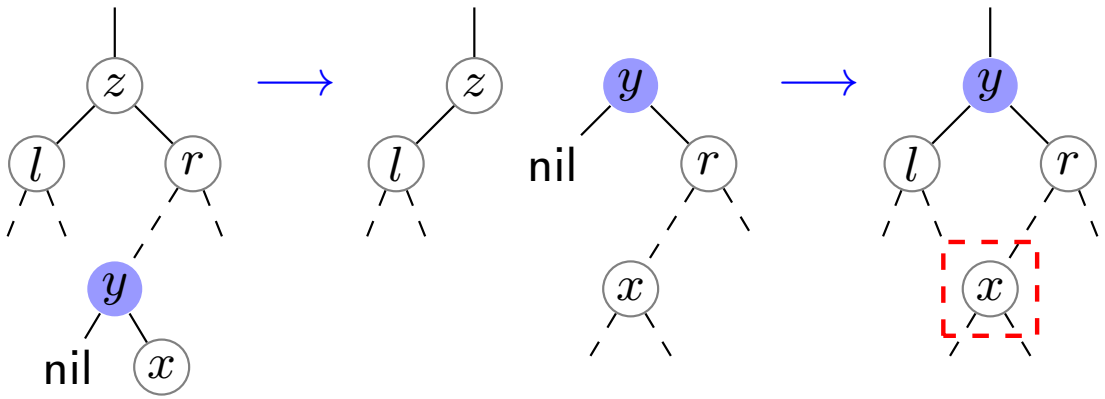
Xian Qiu

Simon Fraser University

xianq@sfu.ca

Deletion

recall the deletion (of z) in binary search trees



what if it is a red-black tree?

- ▷ implement the above and let y.color $= z$.color
- ▷ no problem if y was red
- ▷ problems may occur if y was black

Idea

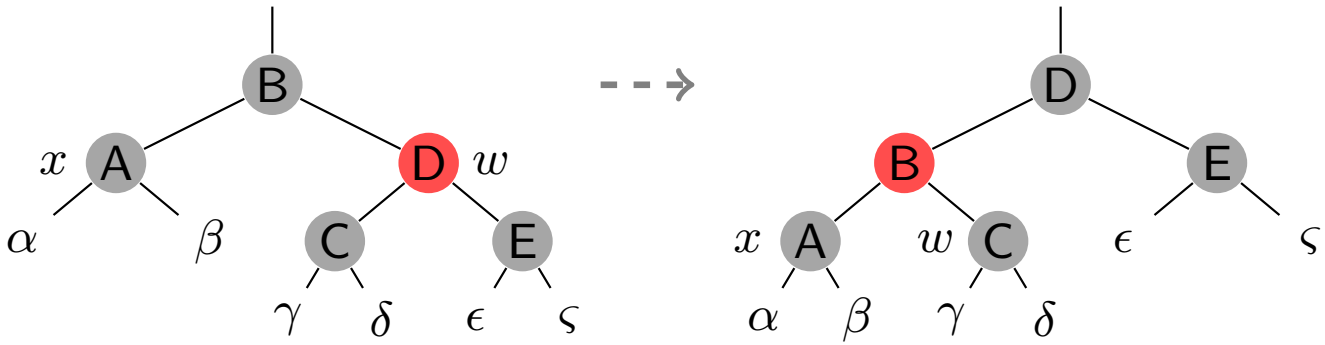
Key idea

Add an extra black to x when counting black-height, thus the black-height property holds as before.

- x might be red-and-black (if x.color $=$ RED) or double black (if x.color $=$ BLACK)
- if x is red-and-black, then color x black (done)
- else use ROTATION to modify tree, then remove the extra black
- look at x's sibling w (always existed?)
- in the following, assume $x = x$.p.left (else, similar)

Case 1

w is red



switch the colors of w and w.p;

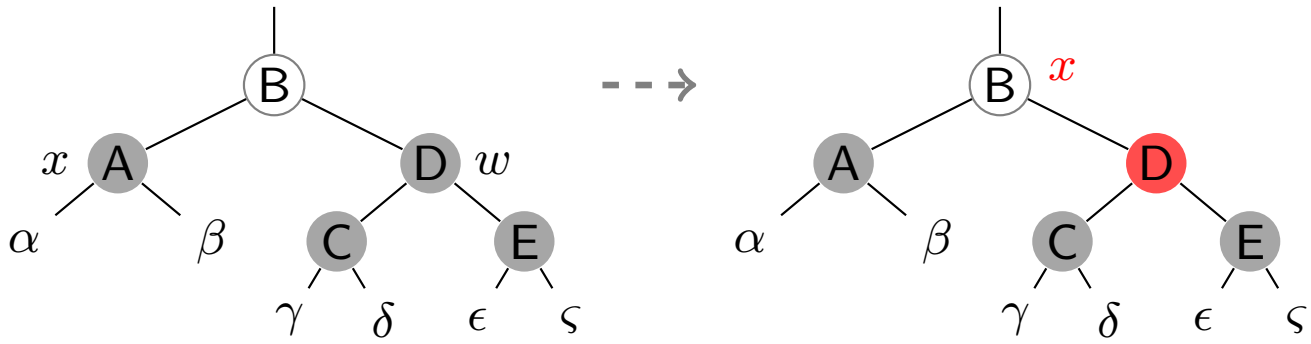
LEFT-ROTATE(w.p);

update w;

- ▷ black height property holds
- ▷ now w is black, then distinguish w's children

Case 2

(w is black) its two children are black



take one black off x and w;

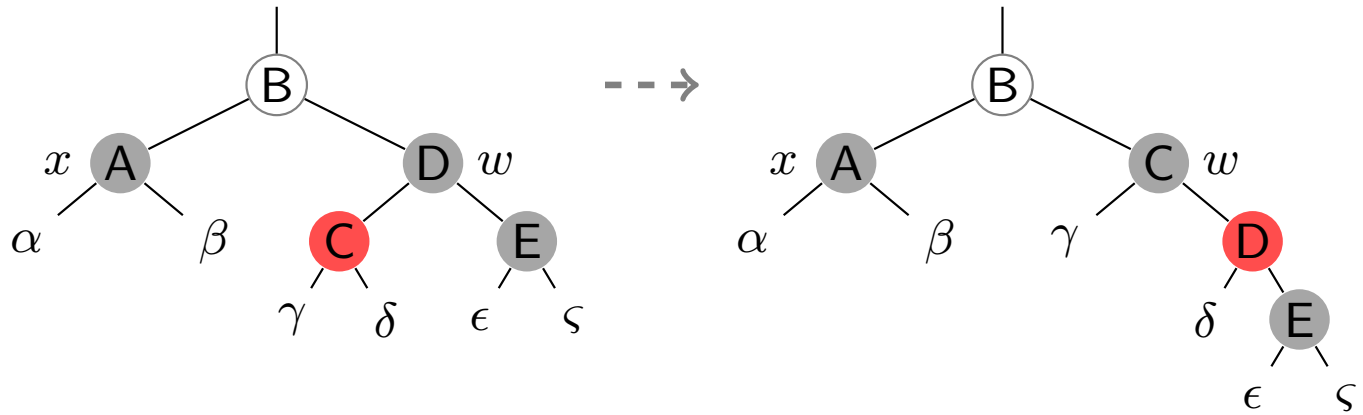
add an extra black to x.p;

update x and w;

- ▷ if enter case 2 via case 1, then new x is red-and-black
- ▷ color x black and terminate else go to case 3 or 4

Case 3

(w is black) w.left is red and w.right is black



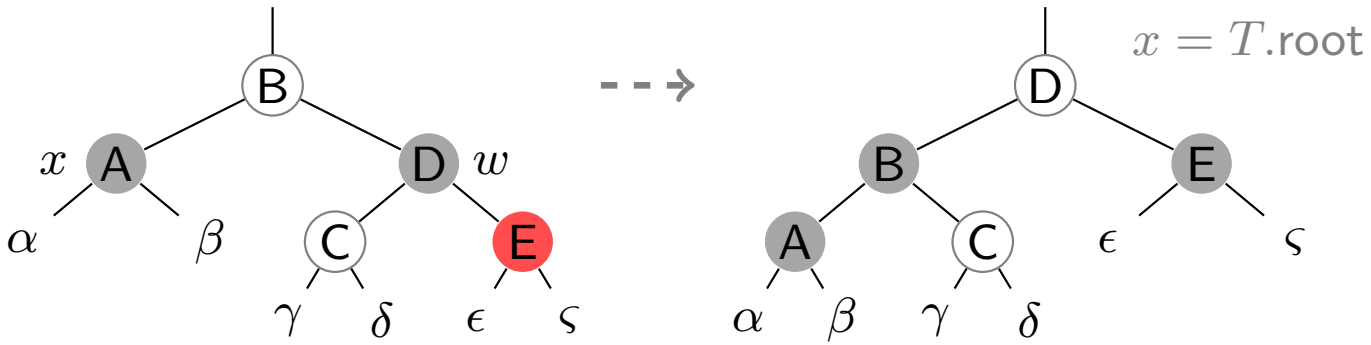
switch the colors of w and w.left;

RIGHT-ROTATE(w);

- ▷ black-height property holds
- ▷ go to case 4

Case 4

(w is black) w.right is red



swith the colors of w and w.p;

LEFT-ROTATE(w.p);

w.color = BLACK and remove the extra black of x;

update $x := T$.root

- ▷ black-height property holds
- ▷ terminate (running time?)

Pseudocode

RB-DELETE-FIXUP(T,x)

```
1  while x ≠ T.root and x.color == BLACK do
2      if x == x.p.left then
3          w = x.p.right ;                                   // sibling w
4          if w.color == RED then
5              w.color = BLACK; x.p.color = RED ;            // case 1
6              LEFT-ROTATE(T, x.p); w = x.p.right;
7          if w.left.color == BLACK and w.right.color == BLACK then
8              w.color = RED; x = x.p ;                      // case 2
9          else if w.right.color == BLACK then
10             w.left.color = BLACK; w.color = RED ;         // case 3
11             RIGHT-ROTATE(T, w); w = x.p.right;
12         w.color = x.p.color; x.p.color = BLACK ;          // case 4
13         w.right.color = BLACK; LEFT-ROTATE(T, x.p); x = T.root;
14     else ... same as above, with "left" and "right" exchanged
15 x.color = BLACK;
```

Summarization

	insert	delete	search	sort keys
linked list				
doubly linked list				
hashing with chaining				
open addressing				
binary search tree				
red-black tree				