Script generated by TTT

Title: FDS (08.06.2018)

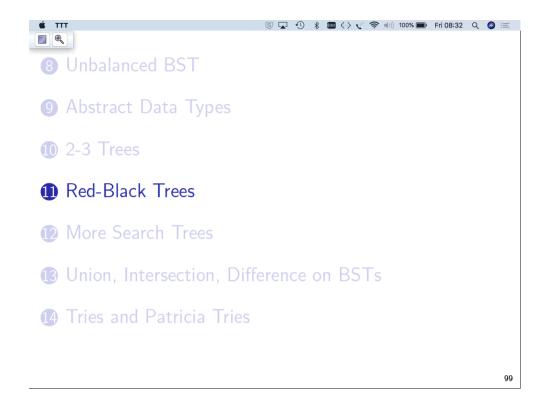
Date: Fri Jun 08 08:32:13 CEST 2018

Duration: 93:00 min

Pages: 89



HOL/Data_Structures/
RBT_Set.thy





Relationship to 2-3-4 trees

Idea: encode 2-3-4 trees as binary trees;



Relationship to 2-3-4 trees

Color

Idea: encode 2-3-4 trees as binary trees; use color to express grouping

$$\langle \rangle \approx \langle \rangle$$

101

104



Structural invariants

Logarithmic height

 $invh :: 'a \ rbt \Rightarrow bool$ $invh \langle \rangle = True$ $invh \langle _{-}, \ l, _{-}, \ r \rangle = (invh \ l \wedge invh \ r \wedge bh(l) = bh(r))$

Lemma
$$rbt \ t \Longrightarrow h(t) \le 2 * \log_2 |t|_1$$



Insertion



Insertion

 $insert :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt$

 $insert :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt$ $insert \ x \ t = paint \ Black \ (ins \ x \ t)$

108



Insertion

Insertion

```
insert :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt
insert \ x \ t = paint \ Black \ (ins \ x \ t)
ins :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt
ins \ x \ \langle \rangle = R \ \langle \rangle \ x \ \langle \rangle
```

 $insert :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt$ $insert \ x \ t = paint \ Black \ (ins \ x \ t)$ $ins :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt$ $ins \ x \langle \rangle = R \ \langle \rangle \ x \ \langle \rangle$ $ins \ x \ (B \ l \ a \ r) = (case \ cmp \ x \ a \ of \ a \ r)$ $LT \Rightarrow baliL (ins x l) a r$ $EQ \Rightarrow B \ l \ a \ r$ $GT \Rightarrow baliR \ l \ a \ (ins \ x \ r))$

Insertion

```
insert :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt
insert \ x \ t = paint \ Black \ (ins \ x \ t)
ins :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt
ins \ x \ \langle \rangle = R \ \langle \rangle \ x \ \langle \rangle
ins \ x \ (B \ l \ a \ r) = (case \ cmp \ x \ a \ of
LT \Rightarrow baliL \ (ins \ x \ l) \ a \ r
| EQ \Rightarrow B \ l \ a \ r
| GT \Rightarrow baliR \ l \ a \ (ins \ x \ r))
ins \ x \ (R \ l \ a \ r) = (case \ cmp \ x \ a \ of
LT \Rightarrow R \ (ins \ x \ l) \ a \ r
| EQ \Rightarrow R \ l \ a \ r
| GT \Rightarrow R \ l \ a \ (ins \ x \ r))
```

Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

109



Insertion

```
insert :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt
insert x \ t = paint \ Black \ (ins \ x \ t)
ins :: 'a \Rightarrow 'a \ rbt \Rightarrow 'a \ rbt
ins x \ \langle \rangle = R \ \langle \rangle \ x \ \langle \rangle
ins x \ (B \ l \ a \ r) = (case \ cmp \ x \ a \ of
LT \Rightarrow baliL \ (ins \ x \ l) \ a \ r
| EQ \Rightarrow B \ l \ a \ r
| GT \Rightarrow baliR \ l \ a \ (ins \ x \ r))
ins x \ (R \ l \ a \ r) = (case \ cmp \ x \ a \ of
LT \Rightarrow R \ (ins \ x \ l) \ a \ r
| EQ \Rightarrow R \ l \ a \ r
| GT \Rightarrow R \ l \ a \ (ins \ x \ r))
```



108

Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

• Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r baliL (R (R t_1 a_1 t_2) a_2 t_3) a_3 t_4 = R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)

109



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r baliL $(R (R t_1 a_1 t_2) a_2 t_3) a_3 t_4$ = $R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$ baliL $(R t_1 a_1 (R t_2 a_2 t_3)) a_3 t_4$ = $R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r baliL $(R \ (R \ t_1 \ a_1 \ t_2) \ a_2 \ t_3) \ a_3 \ t_4$ $= R \ (B \ t_1 \ a_1 \ t_2) \ a_2 \ (B \ t_3 \ a_3 \ t_4)$ baliL $(R \ t_1 \ a_1 \ (R \ t_2 \ a_2 \ t_3)) \ a_3 \ t_4$ $= R \ (B \ t_1 \ a_1 \ t_2) \ a_2 \ (B \ t_3 \ a_3 \ t_4)$
- Principle: replace Red-Red by Red-Black

. . . .



Adjusting colors

 $baliL.\ baliR: 'a\ rbt \Rightarrow 'a \Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r $baliL (R (R t_1 a_1 t_2) a_2 t_3) a_3 t_4$ $= R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$ $baliL (R t_1 a_1 (R t_2 a_2 t_3)) a_3 t_4$ $= R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$
- Principle: replace Red-Red by Red-Black
- Final equation: $baliL \ l \ a \ r = B \ l \ a \ r$



Adjusting colors

 $baliL.\ baliR:: 'a\ rbt \Rightarrow 'a\Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/rbaliL (R (R t_1 a_1 t_2) a_2 t_3) a_3 t_4 $= R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$ $baliL (R t_1 a_1 (R t_2 a_2 t_3)) a_3 t_4$ $= R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$
- Principle: replace Red-Red by Red-Black



Adjusting colors

 $baliL.\ baliR::'a\ rbt \Rightarrow 'a\Rightarrow 'a\ rbt \Rightarrow 'a\ rbt$



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r $baliL (R (R t_1 a_1 t_2) a_2 t_3) a_3 t_4$ $= R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$ $baliL (R t_1 a_1 (R t_2 a_2 t_3)) a_3 t_4$ $= R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)$
- Principle: replace Red-Red by Red-Black
- Final equation:



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r baliL (R (R t_1 a_1 t_2) a_2 t_3) a_3 t_4 = R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4) baliL (R t_1 a_1 (R t_2 a_2 t_3)) a_3 t_4 = R (B t_1 a_1 t_2) a_2 (B t_3 a_3 t_4)
- Principle: replace Red-Red by Red-Black
- Final equation: $baliL \ l \ a \ r = B \ l \ a \ r$



Logarithmic height

Lemma

$$rbt \ t \Longrightarrow h(t) \le 2 * \log_2 |t|_1$$

107



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

• Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$



Adjusting colors

baliL, baliR :: 'a $rbt \Rightarrow$ 'a $rbt \Rightarrow$ 'a rbt

- Combine arguments l a r into tree, ideally $\langle l, a, r \rangle$
- Treat invariant violation Red-Red in l/r baliL $(R \ (R \ t_1 \ a_1 \ t_2) \ a_2 \ t_3) \ a_3 \ t_4$ = $R \ (B \ t_1 \ a_1 \ t_2) \ a_2 \ (B \ t_3 \ a_3 \ t_4)$ baliL $(R \ t_1 \ a_1 \ (R \ t_2 \ a_2 \ t_3)) \ a_3 \ t_4$ = $R \ (B \ t_1 \ a_1 \ t_2) \ a_2 \ (B \ t_3 \ a_3 \ t_4)$
- Principle: replace Red-Red by Red-Black
- Final equation: $baliL \ l \ a \ r = B \ l \ a \ r$



Description of the Control of the Co

The while loop in lines 1-15 maintains the following three-part invariant at the start of each invariant of the loom

- Node z is red.
 If z a is the rest, then z a is black
- c. If the tree violates any of the red-black properties, then it violates at moon of them, and the violation is of either property 2 or property 4. If the violation property 2, it is because a in the root and in red. If the time violates property 3.

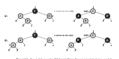
Part (c), which death with with time of red-black properties, in more central downing that RE-043307 INCOF restores the mid-black properties the parts and (b), which we use along the ways understand distantion in the code. Resembly the focusing on code z and notice mast z in the tree, it halps to know the part (c) that z in and W and the support z in the z in the z is known to part (c) that z in the z is the z in z in z in z in z in the z in z in

too or no loop, that each intractor minimized that too primitarilar, the material residence where the same an amen's generally at loop permission segments. Then, as we a examine how the body of the loop week in more detail, we shall appear that the loop maintains the invention upon each intention. Along the way, we shall also demonstrate them each limitation of the policy and the same and the

Initialization: Prior to the first keration of the loop, we stated with a red-black tree with no violations, and we added a red node z. We show that each part of

- a. When RB-000000 PREUP is called, z is the red node that was abled. b. If z, p is the root, then z, p started out black and did not change prior to the
- call of RB-045507-PULUE c. We have almoly som that properties 1, 3, and 5 hold when RB-145507 PULUP is called.
- If the tree violates property 2, then the rad root must be the newly said node e, which is the only interest node in the tree. Because the present both children of z are the nazine, which in black, the tree does not it while property 4. Thus, this violation of property 2 is the only violation and-black properties in the entire tree.

With tree volume property 4, then, became the Children of node 2 are four methods and the tree had no other violations prior to 2 being added, the



require two teams to the processing at a content of the property is the state, that it is an analysis of the processing the processing that it is a substitute of A_i , A_i , and A_i has a likely seen of an all high for some A_i , A_i , and A_i has a likely support of the processing the property A_i and A_i has a likely support of the processing A_i and A_i has a likely support of the processing A_i and A_i has a likely support of the processing A_i and A_i has a likely support of the processing A_i and A_i has a likely support of the processing A_i and A_i has a likely support of the processing A_i and A_i has a likely support of the processing A_i and A_i has a likely support of the processing A_i has a likely support of the processing A_i has a likely support A_i and A_i has a likely support A_i in the processing A_i has a likely support A_i in the processing A_i has a likely support A_i in the processing A_i has a likely support A_i in the processing A_i has a likely support A_i in the processing A_i has a likely support A_i in the processing A_i has a likely support A_i in the processing A_i in the processin

If make of 'e' is the root at the said of the next illustration, then can I convent to lower visition (pumperly 4 in the laters). Since \mathcal{C} is real and it is the root, proposing 2 theorems the only one that is vicinate, and the 'elektrini' to root in the said in the root in the said of the root in the said of the root is the said of the root is formed as the said of the root of the root in the said and a vicination of property 2. Cans I command to the root vicination of property 4 the saids at the root of the foresterior. It then mode \mathcal{C}' and it the $\mathcal{C}_{\mathcal{C}}$ when $\mathcal{C}_{\mathcal{C}}$ is well as $\mathcal{C}_{\mathcal{C}}$ is the said $\mathcal{C}_{\mathcal{C}}$ in the said \mathcal{C}_{\mathcal

Case 2: zh uncle y k Nack and z ir a right chi

In case 2 and 3, the cities of z h uncle y is black. We distinguish the two case according to whether z is a right or left child. We distinguish the two case case 2, which is shown in Figure 13.6 together with case 3, to case 2, note in a right child of its parent. We immediately use a left rotation to transform in a right child of its parent. We immediately use a left rotation to transform

Proof in CLRS

At Aprile

no other mid-black properties.

Thereismistes: When the long-testiment, it does no because z.p in black. (If z is the more, ben z.p is the sected 7-m2, which is black.) Thus, the tree does not violate property 4 is long-testimation. By the long-invation, the only preparely than englist field the black line groups? 2. Line 16 recommon this property, one, that

that engith that included is projectly T. Like 10 melones that projectly, then, we then $\mathbb{E} T^2 \sim \operatorname{transpire} T_{t}$ that extracts in $\mathbb{E} T^2 \sim \operatorname{transpire} T_{t}$ that extracts in $\mathbb{E} T^2 \sim \mathbb{E} T^2$

We defengate one 1 from case 2 and 3 by the color of a's present with or "unite" Lise 3 radius y priess to it works a p.p.y pile, and lise 4 was color. If y is not, then we accurate case 1. O home inc, control present to case and 3. In all these cases, a's grandpasser a p.p. pinhiback, drone its person a.p. not, and property 4 is schooled only home was and a.p.

Care I: L'et marke y à red Figure 13.5 donce the situation for one 1 (line 5-th, which course when both Lp and y as red. Hearness Lp job black, we can odde both Lp and y black, thorstly hings the professor of and Lp yboth balley and, and we can red at Lp ared, thereby melaticaling appears 5. We then report the while loop with Lp p as the new rode L. The primer x consequence while in the stars. Now, we show that case I materials the loop invarient at the stars of the next section. We can no shows no do to the present invarience and x = no.

 Because this iteration online z, p, p and, node z is red at the start of the next investige.

h The node ε', ρία ε, ρ, ρ, ρ is this iteration, and the order of this node does not change. If this node is the root, it was black price to this iteration, and it remains black at the start of the start iteration.
c. We have already agoust that case 1 maintains property 5, and it does not introduce a violation of property 1.0.



The second secon

both and cyams and, the nation offers switer for bischhalger of under on proper. The Share we are and and Share's personal process of the in blad, does otherwise we would have account on an 1. Additionally, the best of the second of the second of the second of the second of the the second of the best field 3 tensors, and a felt removing open intent is to the other does not lead to the second, and after removing open intent is to the other we monate some code changes and a sight remove, which preserve years and the second of the second we monate some code changes and a sight remove, when there is no work intention is not second or the second of the second of the second of the work intention of the second of the second of the second of the second of the work intention of the second of the work intention of the second of

a Case 2 makes a roote to a se, which is red. No further change to a or in color

- occurs to case 2 and 3. In Case 3 makes z, p black, so that if z, p is the root at the start of the next
- c. As in case 1, proportion 1, 3, and 5 are maintenand in cases 2 and 3. Since node 2 is not the not in case 2 and 3, we know that there is no vitation for properly 2. Cases 2 and 3 do not be tended as violation of propersions that only node that it made not become a child of a black node by metion in case 3.

Preservation of invariant

After 14 simple lemmas:

Theorem

 $rbt \ t \Longrightarrow rbt \ (insert \ x \ t)$

110

Ougan 12 Red Block Trees

The while loop in lines 1-15 maintains the following three-part invariant at the start of such iteration of the loop:

- a. Node z is md.
- c. If the tree violates any of the red-black properties, then it violates at me one of them, and the violation is of either property 2 or property 4. If it

Part (c), which dash with vishtims of ref-black properties, is more central aboving that EE-003397-TEUP restress the mid-black properties then peris and (b), which was along the ways to understand distinctions in the code, linear was along the ways to understand distinction in the code, linear was 1 he focusing on node z and nodes near z in the tree, it helps to know the part (c) the z in m. But all all maps m (c) to the that the node z, p_p exists when restresses in the 2, 2, 3, 3, 4, 3 and 4.

invarions given us a undefic property at loop invariantion.
We start with the initialization and steministics arguments. Then, as we are in those the body of the loop works in some detail, we shall argue that the invariant spon each itemities. Along the way, we shall also demonstrate that each invariant spon each itemities. Along the way, we shall also demonstrate that each invariant or loop in two possible outcomes; either the point

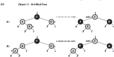
Initialization: Prior to the first keration of the loop, true with no violations, and we added a red node

- the invariant holds at the time RE-DOSE V-PULUP is called:

 a. When RE-DOSE V-PULUP is called: a in the real mode that was abled.
- b. If z.p is the root, then z.p started out black and did not change prior to the call of RB-besser-Puxure
- c. We note almost you may proportion 1, 5, and 5 must work not not not not put the facilitat. If the true violates property 2, then the red root must be the newly addenote c, which is the only betarned node in the true. Became the present in both children of c are the notified, which is black, the true does not identified to the property 4. Then, this violation of money? 2 the notified within the notion?

both children of z are the sentinel, which is black; the true does not also white property 4. Thus, this violation of property 2 is the only violation of and-black properties in the entire true.

If the true violation property 4, then, because the children of node z are black number & and the true had no order violations price to z being added, the



Piger at 1.8.5. Case if of the personion is to be more if there is properly it is window, does at and its person if per in the first of the contraction of the late of the contraction o

If mode I've the root is the state of the accumulation, their case I cream the loos violation of property 4 in the itematics. Some of I and and it its mot, property 2 becomes the only one that is violated, and the violated due to e'.
If node e' is not the most at the stat of the next itemation, then case I it most creamed a violation of property 2. Case I commend the lone violated of property 4 that exhibit as the start of this terration. It then most all the lone violates of property 4 that exhibit as the start of this terration. It then most all the lone violates of property 4 that exhibit as the start of this terration. It then most all the lone violates are the lone violates of property 4 that exhibit as the start of this terration. It then most all the lone violations of property and the lone violates are the lone violates of property and the lone violates of th

Case 2: 15 uncle y & Nackand 1 is a right ch

Case 2: th seeds y it blanchand; it is algorithm!
In many 2 and 3, he caller of x based y is black. We distinguish that two can
accrossing to whether x is a right or last child of x.p. 1.5m. 1.5m. 1.5m.
accrossing to whether x is right or last child of x.p. 1.5m. 3. In one 2, note
that the contract of x.p. 1.5m. 1.5m. 2. Note
that yield the contract of x.p. 1.5m. 2. Note that the contract of x.p.
the contract of x.p. 1.5m. 2. Note that the contract of x.p. 1.5m. 2. Note
that yield the contract of x.p. 1.5m. 2. Note that the contract of x.p. 1.5m. 2. Note
that yield the contract of x.p. 1.5m. 2. Note that the contract of x.p. 1.5m. 2. Note that the contract of x.p. 1.5m. 2.

**The contract of x.p. 1.5m. 2. Note that the contract of x.p. 2. Note that the contract of x.p. 2.

**The contract of x.p. 2. Note that X.p. 2.

**The contract of x.p. 2. Note that X.p. 2.

**The contract of x.p. 2.

**The contr

Proof in CLRS

If it is a section which is the because both z and z,p are med. Moreover, the tree vi

We missisted that the component into the contract pair before the contract pair to the sent of 1 and, which is black? Then, the me distribution of the contract pairs and the contract

then x, p is block. Since we note a long iteration only if x, p is red, we line that x, p cannot be the rest. Hence, x, p, p and x. We distinguish case 1 from cases 2 and 3 by the solor of x's parent's obbs or "white" Line 3 red are y points in x in order $x, p \neq x$ pile, and line x man ysolor. If y is set, then we execute case 1. O there we, control passes in cases and 3. y of the control y is the set of y in y is the control y in y.

One I: z'v muck y k and Figure 15.5 shows the situation for case 1 (time 5–6), which occurs who then $L_{x}y$ and y are red. Resease $L_{x}y$ is black, we can order both $L_{x}y$ and y black, then y in freigh the profitten of z and $L_{x}y$ both being red, and we are order $L_{x}y$ rend, then y multistiving reporty S. We then report the while large with $L_{x}x$ y and then so reds z. The profitte z remains y to the latter than

to denote the node that will be called node a at the test in line I upon the ne keration.

invaries.

It has node $x' \cdot p$ is $x, p \cdot p, p$ in this iteration, under order of this node down to change. If this node is the root, it was black prior to this iteration, and it measure black at the start of the seat invaries.

We have allowed prompt the court of measurements and it does not this invaries.



Figure 15.6. Cases I and it of Engineering 15 to 15 t

such a cut a para and, the contribution that is this blobblegie or the opposity. She believe were can all finitely of broughosts, 2, clust is blob, diese otherwise were confidence excellent data. I. Additionally, such cap actus, how we where appell but the sole existed of the first such cap actus, how we where appell but the sole existed of the first such cap actus, how we where appell but the sole desired in the first shown one look is lost 11, the feating of c.p.p remains undanged. Incomtract the such as the contribution of the contribution of the constraints of the contribution of the contribution of the constraints of the contribution of

argument, it is well not ready upon the next test in line 1, and the loop next was in manuse again.)

a. Chec. 2 makes a resistant on a which is not. No further change to a set become

- occurs in cases 2 and 3.

 b Case 3 makes at p. Made, so that if at p is the root at the start of the next installor, its bits at p. Made, so that if at p is the root at the start of the next installor, it is because 1.3. and 5 are resistabled in cases 2 and 3.
- Carlo is care 1, properties 1, 2, and 5 one malemined in cases 2 and 2.

 Now node, it is not fis much in cases 2 and 3, we know that there is no visit of property 2. Cases 2 and 3 do not introduce a visition of properties the order of the cases 3 and 3 do not introduce a visition of properties on the only node that introde not become a child of a black node by middle in case 3.



111

Deletion

 $delete \ x \ t = paint \ Black \ (del \ x \ t)$



Deletion

```
\begin{aligned} \operatorname{delete} x & t = \operatorname{paint} Black \ (\operatorname{del} x \ t) \\ \operatorname{del} _{-} & \langle \rangle &= \langle \rangle \\ \operatorname{del} _{x} & \langle _{-}, \ l, \ a, \ r \rangle &= \\ (\operatorname{case} \ \operatorname{cmp} \ x \ a \ \operatorname{of} \\ LT &\Rightarrow \\ & \operatorname{if} \ l \neq \langle \rangle \wedge \operatorname{color} \ l = Black \\ & \operatorname{then} \ \operatorname{bald} L \ (\operatorname{del} \ x \ l) \ a \ r \ \operatorname{else} \ R \ (\operatorname{del} \ x \ l) \ a \ r \\ | \ EQ &\Rightarrow \operatorname{combine} \ l \ r \\ | \ GT &\Rightarrow \\ & \operatorname{if} \ r \neq \langle \rangle \wedge \operatorname{color} \ r = Black \\ & \operatorname{then} \ \operatorname{bald} R \ l \ a \ (\operatorname{del} \ x \ r) \ \operatorname{else} \ R \ l \ a \ (\operatorname{del} \ x \ r) ) \end{aligned}
```



Deletion

Tricky functions: baldL, baldR, combine

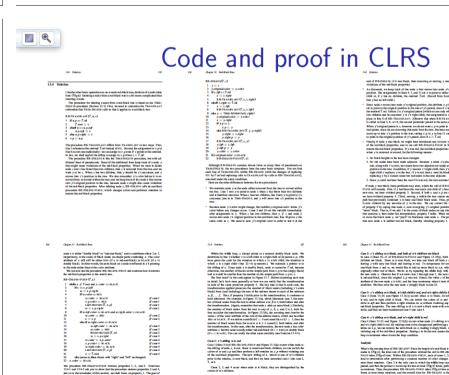
11



Deletion

Tricky functions: baldL, baldR, combine

12 short but tricky to find invariant lemmas with short proofs. The worst:











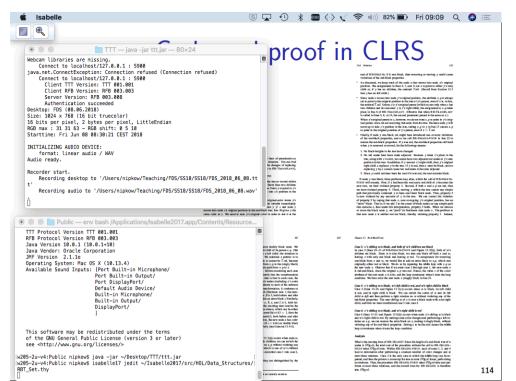














Source of code

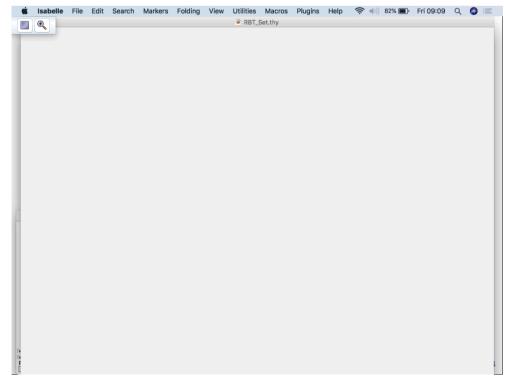
Insertion:

Okasaki's Purely Functional Data Structures

Deletion:

Stefan Kahrs. Red Black Trees with Types.

J. Functional Programming. 1996.





- 8 Unbalanced BST
- 9 Abstract Data Types
- **10** 2-3 Trees
- Red-Black Trees
- 12 More Search Trees
- (B) Union, Intersection, Difference on BSTs
- 14 Tries and Patricia Tries



AVL Trees

[Adelson-Velskii & Landis 62]

116

AVL Trees

[Adelson-Velskii & Landis 62]

• Every node $\langle l, r \rangle$ must be balanced: $|h(l) - h(r)| \le 1$



More Search Trees

AVL Trees

Weight-Balanced Trees

AA Trees

Scapegoat Trees



Weight-Balanced Trees

[Nievergelt & Reingold 72,73]

• Parameter: balance factor $0 < \alpha \le 0.5$



Weight-Balanced Trees

[Nievergelt & Reingold 72,73]

- Parameter: balance factor $0 < \alpha \le 0.5$
- Every node $\langle l, r \rangle$ must be balanced:

$$\alpha \le |l|_1/(|l|_1 + |r|_1) \le 1-\alpha$$

100



Weight-Balanced Trees

[Nievergelt & Reingold 72,73]

- Parameter: balance factor $0 < \alpha \le 0.5$
- Every node $\langle l, r \rangle$ must be balanced: $\alpha \leq |l|_1/(|l|_1 + |r|_1) \leq 1-\alpha$
- Insertion and deletion: single and double rotations depending on subtle numeric conditions



120

Weight-Balanced Trees

[Nievergelt & Reingold 72,73]

- Parameter: balance factor $0 < \alpha \le 0.5$
- Every node $\langle l, -, r \rangle$ must be balanced:

$$\alpha \le |l|_1/(|l|_1 + |r|_1) \le 1-\alpha$$

- Insertion and deletion: single and double rotations depending on subtle numeric conditions
- Nievergelt and Reingold incorrect



Weight-Balanced Trees

[Nievergelt & Reingold 72,73]

- Parameter: balance factor $0 < \alpha \le 0.5$
- Every node $\langle l, r \rangle$ must be balanced: $\alpha \leq |l|_1/(|l|_1 + |r|_1) \leq 1-\alpha$
- Insertion and deletion: single and double rotations depending on subtle numeric conditions
- Nievergelt and Reingold incorrect
- Mistakes discovered and corrected by [Blum & Mehlhorn 80] and [Hirai & Yamamoto 2011]

120

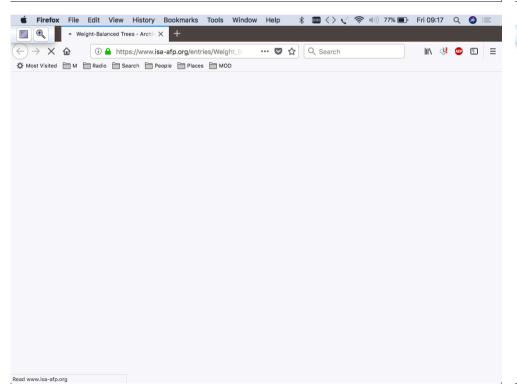


Weight-Balanced Trees

[Nievergelt & Reingold 72,73]

- Parameter: balance factor $0 < \alpha \le 0.5$
- Every node $\langle l, r \rangle$ must be balanced: $\alpha \leq |l|_1/(|l|_1 + |r|_1) \leq 1-\alpha$
- Insertion and deletion: single and double rotations depending on subtle numeric conditions
- Nievergelt and Reingold incorrect
- Mistakes discovered and corrected by [Blum & Mehlhorn 80] and [Hirai & Yamamoto 2011]
- Verified implementation in Isabelle's Archive of Formal Proofs.

12





More Search Trees

AVL Trees
Weight-Balanced Trees

AA Trees

Scapegoat Trees



AA trees

[Arne Andersson 93, Ragde 14]



AA trees

[Arne Andersson 93, Ragde 14]

• Simulation of 2-3 trees by binary trees $\langle t_1, a, t_2, b, t_3 \rangle \rightsquigarrow \langle t_1, a, \langle t_2, b, t_3 \rangle \rangle$

12



AA trees

[Arne Andersson 93, Ragde 14]

- Simulation of 2-3 trees by binary trees $\langle t_1, a, t_2, b, t_3 \rangle \rightsquigarrow \langle t_1, a, \langle t_2, b, t_3 \rangle \rangle$
- Height field (or single bit) to distinguish single from double node



122

AA trees

[Arne Andersson 93, Ragde 14]

- Simulation of 2-3 trees by binary trees $\langle t_1, a, t_2, b, t_3 \rangle \rightsquigarrow \langle t_1, a, \langle t_2, b, t_3 \rangle \rangle$
- Height field (or single bit) to distinguish single from double node
- Code short but opaque



AA trees

[Arne Andersson 93, Ragde 14]

- Simulation of 2-3 trees by binary trees $\langle t_1, a, t_2, b, t_3 \rangle \rightsquigarrow \langle t_1, a, \langle t_2, b, t_3 \rangle \rangle$
- Height field (or single bit) to distinguish single from double node
- Code short but opaque
- 4 bugs in *delete* in [Ragde 14]: non-linear pattern;



AA trees

[Arne Andersson 93, Ragde 14]

- Simulation of 2-3 trees by binary trees $\langle t_1, a, t_2, b, t_3 \rangle \rightsquigarrow \langle t_1, a, \langle t_2, b, t_3 \rangle \rangle$
- Height field (or single bit) to distinguish single from double node
- Code short but opaque
- 4 bugs in delete in [Ragde 14]: non-linear pattern; going down wrong subtree; missing function call; off by 1



AA trees

[Arne Andersson 93, Ragde 14]

After corrections, the proofs:

 Code relies on tricky pre- and post-conditions that need to be found



122

AA trees

[Arne Andersson 93, Ragde 14]

After corrections, the proofs:

- Code relies on tricky pre- and post-conditions that need to be found
- Structural invariant preservation requires most of the work





AVL Trees Weight-Balanced Trees AA Trees

Scapegoat Trees



Scapegoat trees

[Anderson 89, Igal & Rivest 93]

124



Scapegoat trees

[Anderson 89, Igal & Rivest 93]

Central idea:

Don't rebalance every time, Rebuild when the tree gets "too unbalanced"



Scapegoat trees

[Anderson 89, Igal & Rivest 93]

Central idea:

Don't rebalance every time, Rebuild when the tree gets "too unbalanced"

• Tricky: amortized logarithmic complexity analysis



- 8 Unbalanced BST
- Abstract Data Types
- **10** 2-3 Trees
- Red-Black Trees
- More Search Trees
- 13 Union, Intersection, Difference on BSTs
- 14 Tries and Patricia Tries

126

One by one (Union)

127



One by one (Union)

What is better: Adding smaller set to bigger or bigger to smaller?



One by one (Union)

What is better: Adding smaller set to bigger or bigger to smaller?

c(x) = cost of adding element to set of size x



One by one (Union)

What is better: Adding smaller set to bigger or bigger to smaller?

c(x) = cost of adding element to set of size x

• Smaller (*m* elements) into bigger (*n* elements):



One by one (Union)

What is better:

Adding smaller set to bigger or bigger to smaller?

c(x) = cost of adding element to set of size x

• Smaller (m elements) into bigger (n elements): Cost = $c(n) + \cdots + c(n + m - 1)$

12



One by one (Union)

What is better: Adding smaller set to bigger or bigger to smaller?

c(x) = cost of adding element to set of size x

- Smaller (m elements) into bigger (n elements): $Cost = c(n) + \cdots + c(n + m - 1)$
- Bigger into smaller: $Cost = c(m) + \cdots + c(m+n-1)$



One by one (Union)

What is better:

Adding smaller set to bigger or bigger to smaller?

c(x) = cost of adding element to set of size x

- Smaller (m elements) into bigger (n elements): $Cost = c(n) + \cdots + c(n + m - 1)$
- Bigger into smaller: $Cost = c(m) + \cdots + c(m + n 1)$
- $c(x) = \log_2 x \Longrightarrow$ $\mathsf{Cost} = O(m * \log_2(n + m))$



One by one (Union)

What is better:

Adding smaller set to bigger or bigger to smaller?

 $c(x) = \cos t$ of adding element to set of size x

- Smaller (m elements) into bigger (n elements): $Cost = c(n) + \cdots + c(n + m - 1)$
- Bigger into smaller: $\mathsf{Cost} = c(m) + \dots + c(m+n-1)$
- $c(x) = \log_2 x \Longrightarrow$ $\mathsf{Cost} = O(m * \log_2(n + m)) = O(m * \log_2 n)$



• We can do better than $O(m * \log_2 n)$

120



- We can do better than $O(m * \log_2 n)$
- This section:

A parallel divide and conquer approach



127

- We can do better than $O(m * \log_2 n)$
- This section:

A parallel divide and conquer approach

• Cost: $O(m * \log_2(\frac{n}{m} + 1))$



- We can do better than $O(m * \log_2 n)$
- This section:

A parallel divide and conquer approach

- Cost: $O(m * \log_2(\frac{n}{m} + 1))$
- Works for many kinds of balanced trees



- We can do better than $O(m * \log_2 n)$
- This section:

A parallel divide and conquer approach

- Cost: $O(m * \log_2(\frac{n}{m} + 1))$
- Works for many kinds of balanced trees
- For ease of presentation: use concrete type *tree*

Uniform *tree* type

Red-Black trees, AVL trees, weight-balanced trees, etc can all be implemented with one more field per node:

datatype
$$('a, 'b)$$
 $tree = \langle \rangle$
 $\mid Node 'b (('a, 'b) tree) 'a (('a, 'b) tree)$



128

Uniform *tree* type

Red-Black trees, AVL trees, weight-balanced trees, etc can all be implemented with one more field per node:

datatype
$$('a, 'b)$$
 $tree = \langle \rangle$
 $\mid Node 'b (('a, 'b) tree) 'a (('a, 'b) tree)$

We work with this type of trees without committing to any particular kind of balancing schema.



Uniform *tree* type

Red-Black trees, AVL trees, weight-balanced trees, etc can all be implemented with one more field per node:

datatype
$$('a, 'b)$$
 $tree = \langle \rangle$
| $Node$ 'b $(('a, 'b)$ $tree)$ 'a $(('a, 'b)$ $tree)$

We work with this type of trees without committing to any particular kind of balancing schema.

Syntax:

$$\langle b, l, a, r \rangle \equiv Node \ b \ l \ a \ r$$



Just join

Can synthesize all BST interface functions from just one function:

130



Just join

Can synthesize all BST interface functions from just one function:

$$join\ l\ a\ r\ pprox\ Node\ _l\ a\ r$$



129

Just join

Can synthesize all BST interface functions from just one function:

$$join\ l\ a\ r \approx Node\ _l\ a\ r + rebalance$$



Just join

Just join

Given $join :: tree \Rightarrow 'a \Rightarrow tree \Rightarrow tree$ (where tree abbreviates ('a,'b) tree), implement

Given $join :: tree \Rightarrow 'a \Rightarrow tree \Rightarrow tree$ (where tree abbreviates ('a,'b) tree), implement $split :: tree \Rightarrow 'a \Rightarrow tree \times bool \times tree$

13

13



Just join



Just join

Given $join :: tree \Rightarrow 'a \Rightarrow tree \Rightarrow tree$ (where tree abbreviates ('a,'b) tree), implement $split :: tree \Rightarrow 'a \Rightarrow tree \times bool \times tree$

 $insert :: 'a \Rightarrow tree \Rightarrow tree$

Given $join :: tree \Rightarrow 'a \Rightarrow tree \Rightarrow tree$ (where tree abbreviates ('a,'b) tree), implement

 $split :: tree \Rightarrow 'a \Rightarrow tree \times bool \times tree$ $insert :: 'a \Rightarrow tree \Rightarrow tree$

 $union :: tree \Rightarrow tree \Rightarrow tree$



Union, Intersection, Difference on BSTs Correctness

join for Red-Black Trees



Just join

Given $join :: tree \Rightarrow 'a \Rightarrow tree \Rightarrow tree$ (where tree abbreviates ('a,'b) tree), implement

 $split :: tree \Rightarrow 'a \Rightarrow tree \times bool \times tree$

 $insert :: 'a \Rightarrow tree \Rightarrow tree$ $union :: tree \Rightarrow tree \Rightarrow tree$ $join2 :: tree \Rightarrow tree \Rightarrow tree$ $delete :: 'a \Rightarrow tree \Rightarrow tree$