Data structures lab – week 6

Welcome back!

Wake-up quiz – LLRB versus RB

- What did our last week results about left-leaning red-black trees show us?

  a) They have less code

  b) They are easier to understand

  c) They are a bit slower than textbook RB

  d) All of the above

Wake-up quiz – LLRB versus RB

- What did our last week results about left-leaning red-black trees show us?
  - a) They have less code
  - b) They are easier to understand
  - c) They are a bit slower than textbook RB
  - d) All of the above
- d is the correct answer
  - Recursion is often shorter, more clear and a bit slower.

Week 5 recap

- Balanced trees
- Left-leaning red-black trees
  - Background for red-black trees
    - 2-3-4 trees
- Assignment 3, part 1

Week 5 class evaluation

- Midterm survey
- Done in class, only 23 were there!
  - Again, probably midterm madness
  - But still disappointing
- Overall ”ok” in speed and difficulty
- Content is interesting – good!

Week 5 class evaluation

- Comments (slightly edited):
  - ”It's early”
    - Can't do anything about that, unfortunately
      - Drink coffee
  - ”Motivate the material”
  - ”Double check content for accuracy”
  - ”Wake-up quizzes are good”
  - ”Good class”, ”good stuff”, ”good work”
    - Thanks

Outline

- Assignment 2 gotchas.
- Balanced trees
  - Revisited
- Assignment 3, part 2

Hints for success

- Hint number 1: Read the assignment
- Hint number 2: Look at your code
- Hint number 3: Comply with standards
- Hint number 4: Use large test cases
- Hint number 5: Use the terminal
- Hint number 6: Use IX and g++
- Hint number 7: Fear the NULL
- Hint number 8: Use a debugger
- Hint number 9: Start earlier

Hints for success

- Hint number 10
- Still to be determined
    - Don't send any more suggestions
- I will make a poll today
    - Go and vote!
        - You do not need to register for this one
- They are all good suggestions!
    - Got the participation on the blog going
        - (even if I had to reward you for it :-)

Hint number 10

- Just a few of them:
  - Read the textbook
  - Organize your code
  - Use Google
  - Go to office hours
  - Comment well
  - Write object-oriented in C++
  - Hang out in Deschutes 100
- See them all on the blog!

Assignment 2 gotchas

- Being too fancy is not always good if you cannot finish on time
  - Start with the basics
    - e.g. remove was not required for A2.
      - So don't spend time implementing it unless you have the time.
- Remember the hints
  - e.g. “look at your code”
    - Comment out code used for timing.

Assignment 2 gotchas

- Be aware that I do check for plagiarism
  - I use a special tool to check your submissions.
  - There is a borderline case for this assignment.
  - Do not copy from each other!
  - Study groups are fine
    - Discuss a solution outline, not the solution itself.

Assignment 2 gotchas

- “What to turn in”:
  1. Linked List implementation
  2. BST implementation
  3. Small discussion
- This was apparently ambiguous
  - I'm sorry for that? Not really.
    - If in doubt, ask!

Red-black trees – again

- Properties:
  - Every node is either red or black
  - The root is black
  - Every leaf is black
    - In Cormen, every leaf is a special NIL node.
  - If a node is red, both children are black
  - All simple paths from a node to descendant leaves contain the same number of black nodes.

Red-black trees – again

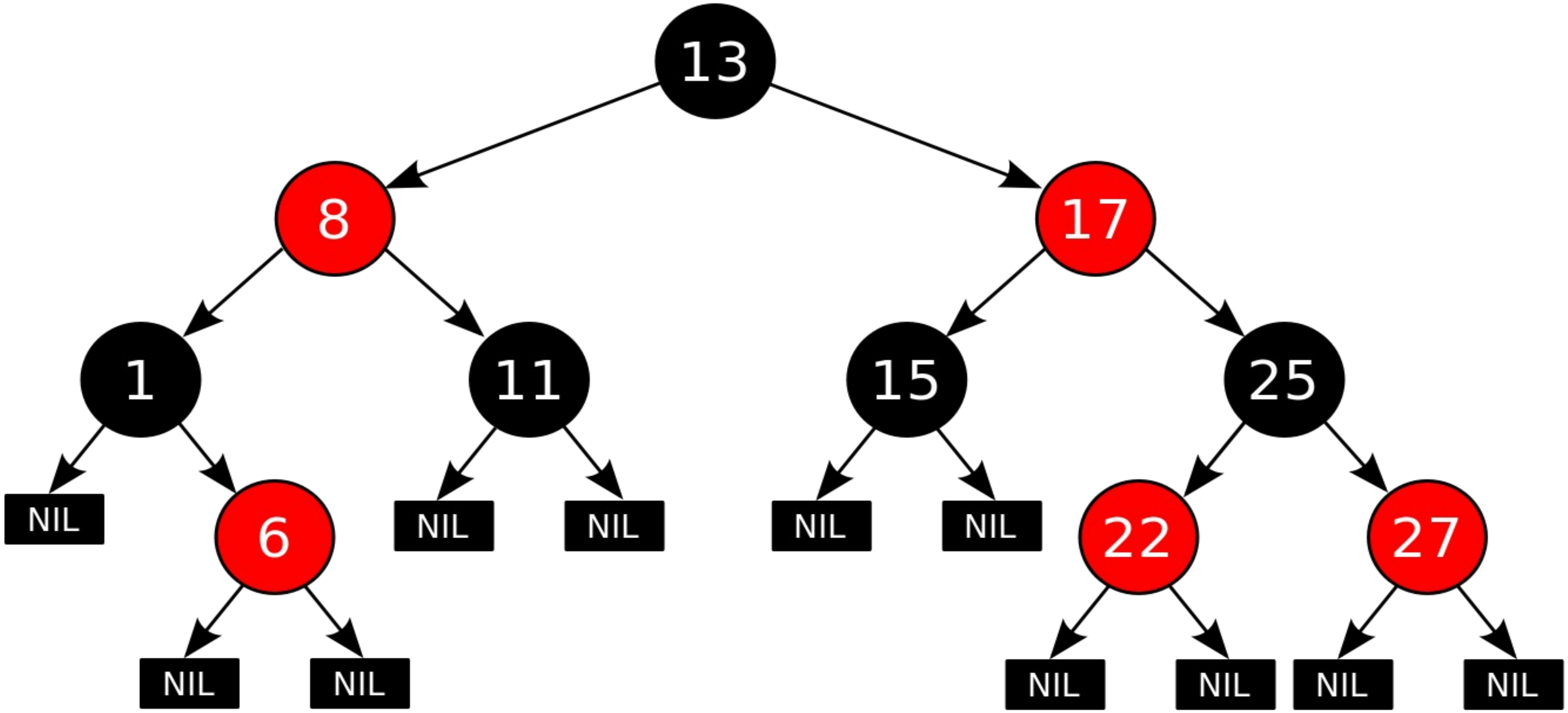
- Balancing happens at insertion
  - And deletion
- All other operations are the same as for BST
- Red-black trees *guarantee:*
- O(lg n) insertion
- O(lg n) deletion
- O(lg n) search

Left-leaning red-black trees

- Something new and exciting (2007)
- Same performance as red-black trees
- Requires all red nodes to be ”left-leaning”
- Simpler to implement
  - Especially because of recursion
    - Remember the first wake-up quiz?

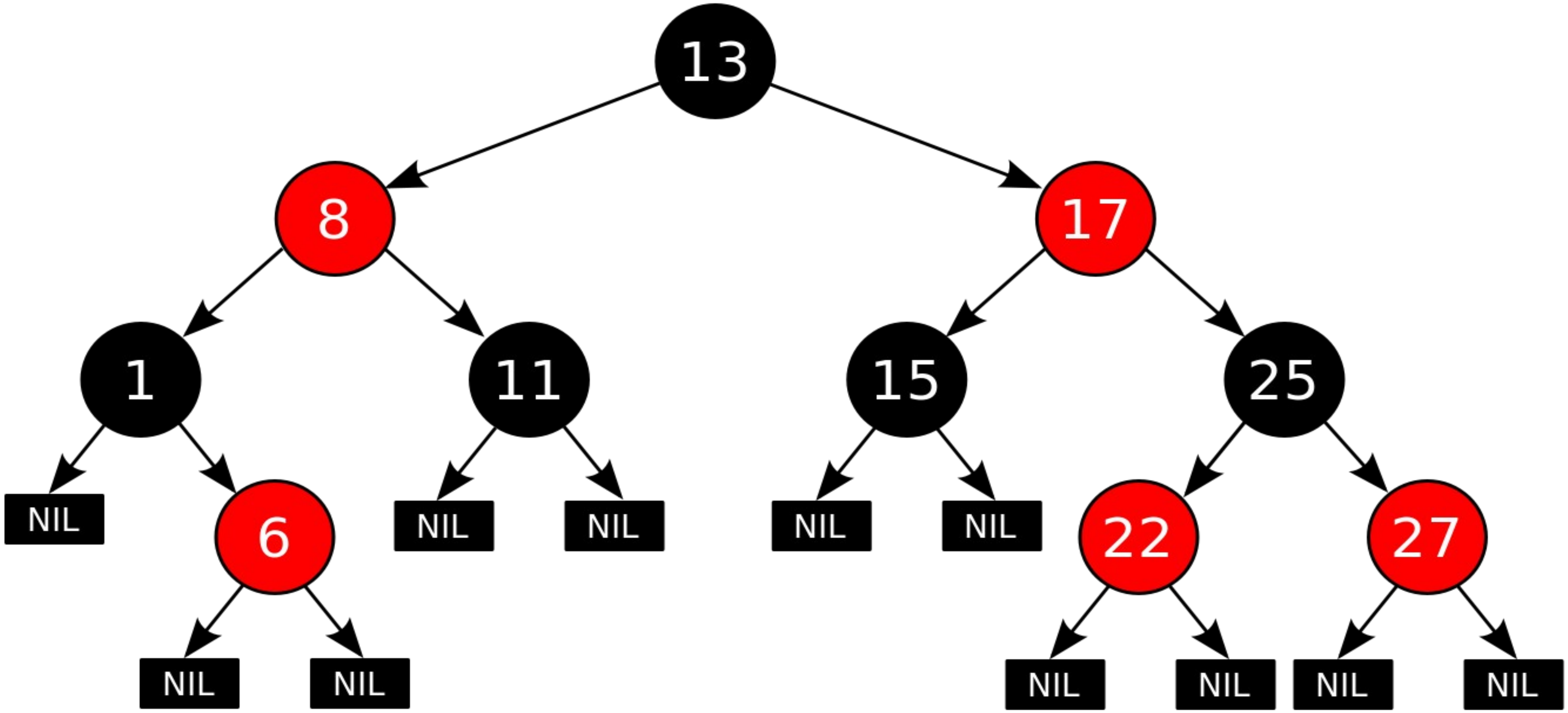
Wake-up quiz – LLRB trees

- Is this a left-leaning red-black tree?



Wake-up quiz – LLRB trees

- Is this a left-leaning red-black tree?



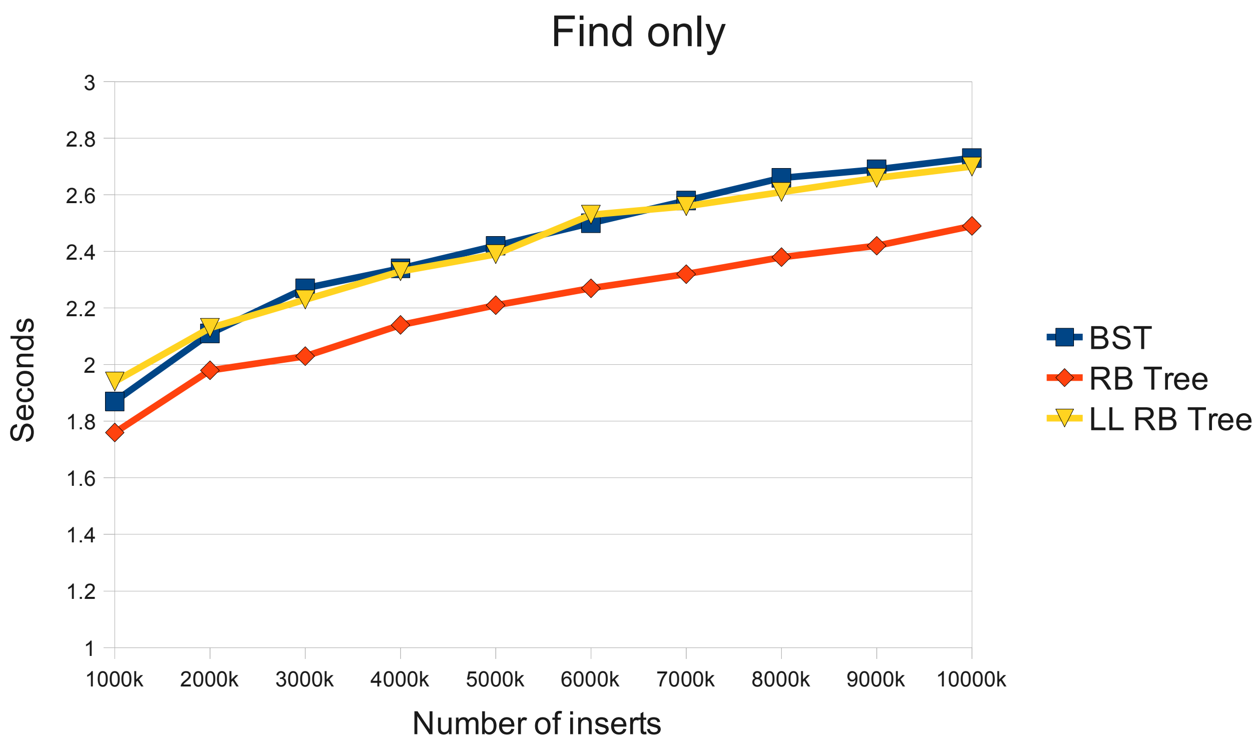
- No... why?

RB versus LLRB

- Insert operation:
  - ~60 versus ~20 lines of code
- Rotations:
  - ~15 versus ~10 lines of code
- That's why you are implementing a LLRB
  - Also because it is new and exciting

RB versus LLRB

- Last week:



- LLRB a bit slower than RB

RB versus LLRB

- Me to Robert Sedgewick (edited):
  - ”My initial findings are that the LLRB trees actually are slower than "normal" RB trees”
- Response (edited):
  - ”If you're finding a significant difference in tree height, I'd be very surprised.”
  - ”For most applications the cost of insert() is insignificant compared to search()”

RB versus LLRB

- Find operation is

$$T = O(h)$$

- We hope find (if we believe RS)

$$h_{RB} = h_{LLRB} = c*\lg(n)$$

- Or at least just an insignificant difference between them.

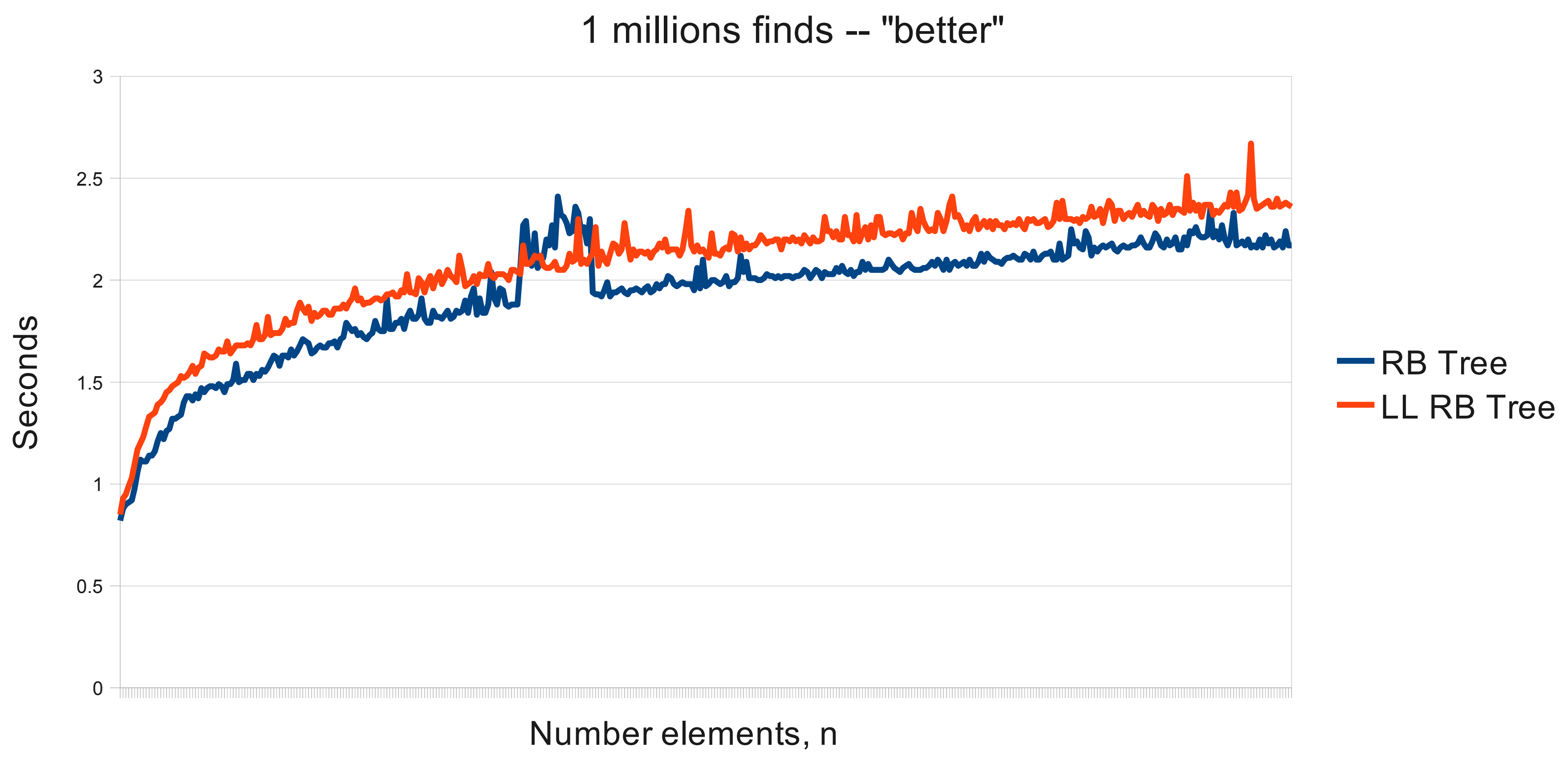
RB versus LLRB

- New results are in!
- 405 testcases
    - Why not just 400?
        - Well, should have been 500
            - But I got tired of waiting for the generator
- Increases of 10,000 (i.e. max 4,050,000)
- 1,000,000 find operations for each case.
- Only measure the find operations.
    - Any difference between RB and LLRB?

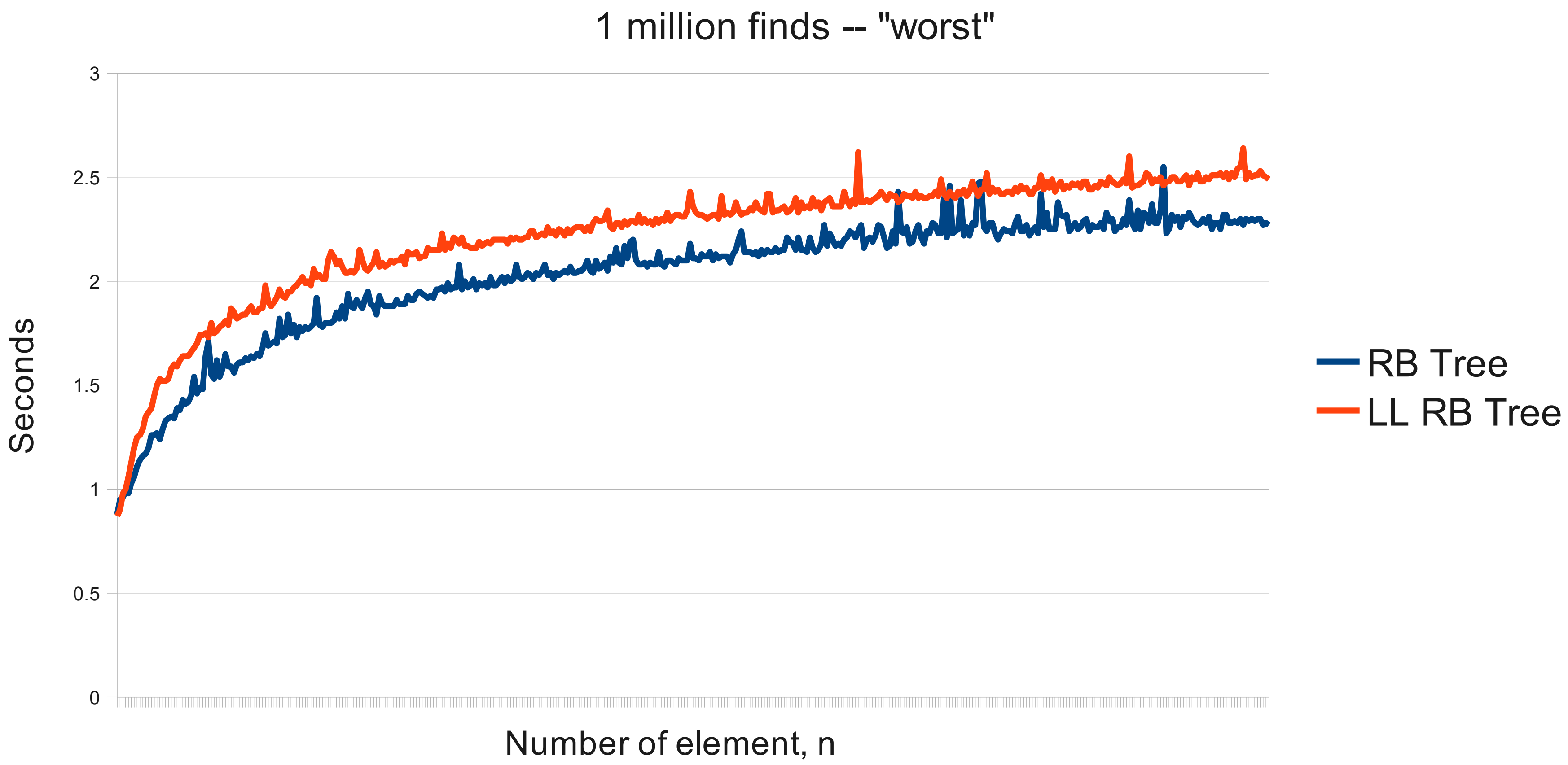
RB versus LLRB

- A reminder:
  - Both trees use the SAME find function.
  - Therefore, the results actually show the difference in average tree height!
    - We cannot use the recursive excuse for bad LLRB performance anymore
- A disclaimer:
  - I use my computer for other things than running tests
    - This might explain fluctuations.

RB versus LLRB – results



RB versus LLRB – results



RB versus LLRB – conclusion

- Tree height is slightly larger for LLRB
  - Not significant though
  - Is outweighed by easier implementation
- Tree height seems to be logarithmically growing

  h = c*lg(n)
- Alright now, I think we're convinced.
  - Let's move on.

Assignment 3 – part 1

- Implement a left-leaning red-black tree.
  - Support insert
  - Do not bother about deletion
  - Support find
    - You should already have this from A2.
- Use *any language* you like
  - Except Java!
- Testcase generator from A2 works fine for testing

Assignment 3 – HELP

- Did anyone implement anything yet?
- Once again, I advertise for HELP
- This Monday, 5 pm, Deschutes 100
- Also, office hours.
- By the way, did anyone notice anything special about the HELP acronym yet?

Assignment 3 – HELP

- Did anyone implement anything yet?
- Once again, I advertise for HELP
- This Monday, 5 pm, Deschutes 100
- Also, office hours.
- By the way, did anyone notice anything special about the HELP acronym yet?
  - It is recursive!
  - HELP Enhances the Learning Process

Assignment 3 – part 2

- A little bit on the board...
- … and then over to the website

Assignment 3 – part 2

- So we have to deal with order statistics
- CLRS:

  "the *i*th order statistic of a set of *n* elements ... is simply the element in the set with the *i*th smallest key.

- S = {5, 3, 6, 8, 2}
- What is the 4th order statistic (OS) in S?

Order statistics

- So we have to deal with order statistics
- CLRS:

> "the *i*th order statistic of a set of *n* elements ... is simply the element in the set with the *i*th smallest key.

- S = {5, 3, 6, 8, 2}
- What is the 4th order statistic (OS) in S?
  - 6... because it is the 4th smallest number
  - How did you do this?

Order statistics – method

- S = {5, 3, 6, 8, 2}
- We want to find 4th OS.
  - Sort?
  - Count?
  - Use magic powers?
- S' = {2, 3, 5, 6, 8}
  - This is easier, right?
  - We just count to the 4th number.

Wakeup quiz – Order statistics

- The outline we have just sketched for finding the *i*th order statistic has a running time of:

  a) O(1)

  b) O(lg n)

  c) O(n)

  d) O(n lg n)

  e) O(n^2)

Wakeup quiz – Order statistics

- The outline we have just sketched for finding the *i*th order statistic has a running time of:

  a) O(1)

  b) O(lg n)

  c) O(n)

  d) O(n lg n)

  e) O(n^2)

- The correct answer is d.

Order statistics – method

- Sorting takes $\Omega(n*\lg(n))$
- Going through the list takes $O(n)$
  - *m* OS queries thus take $O(m*n)$
  - If *m* is close to *n* the overall running time is O(n^2)!
- Can we do better than this?
  - Yes

Augmenting red-black trees

- “Some engineering situations … require a dash of creativity”
- “...often, it will suffice to augment a textbook data structure”
- We will augment a red-black tree
  - Making an order-statistics tree

Order statistics tree

- Add data to a node called *size*
- For a node *x*:
- *x.size* = *x.left.size* + *x.right.size* + 1
- Let's do this on the board!
  - For the seven dwarves

Order statistics tree

- Finding the rank (*i*th OS) for a node *x*.
- Outline:
  - We start at x
  - Go up the tree to the root
    - i.e. maximum *h* steps
  - Along the way we calculate the size of all nodes preceding *x*.
- Since our tree has height $h = \lg n$, OS-rank runs in $O(\lg n)$ time.

OS – finding the rank

- Finding the rank (*i*th OS)

OS-rank(T,x)

```
r = x.left.size+1
y = x
While (y != T.root)
    If (y == y.p.right)
        r = r + y.p.left.size + 1
    y = y.p
return r
```

Assignment 3

- Step 1: Implement the LLRB
  - Left/right rotate, color flip, insert
- Step 2: Augment the LLRB with dynamic order statistics.
  - *size* for each node
    - Modifications to insert and rotation
  - OS-rank implementation
- Step 3: Solve the problem
  - *name* for each node

Assignment 3 – tips

- Reading in a number and name

```
int key; string name;
cin >> key >> skipws >> name;
```

- Finding the correct rank
  - High scores should have highest rank
  - Use textbook OS-rank
  - Return (*n* + *1*) – OS-rank
    - Or reverse the tree on insertion

Thank you

Questions?