CSC 225

Algorithms and Data Structures: I

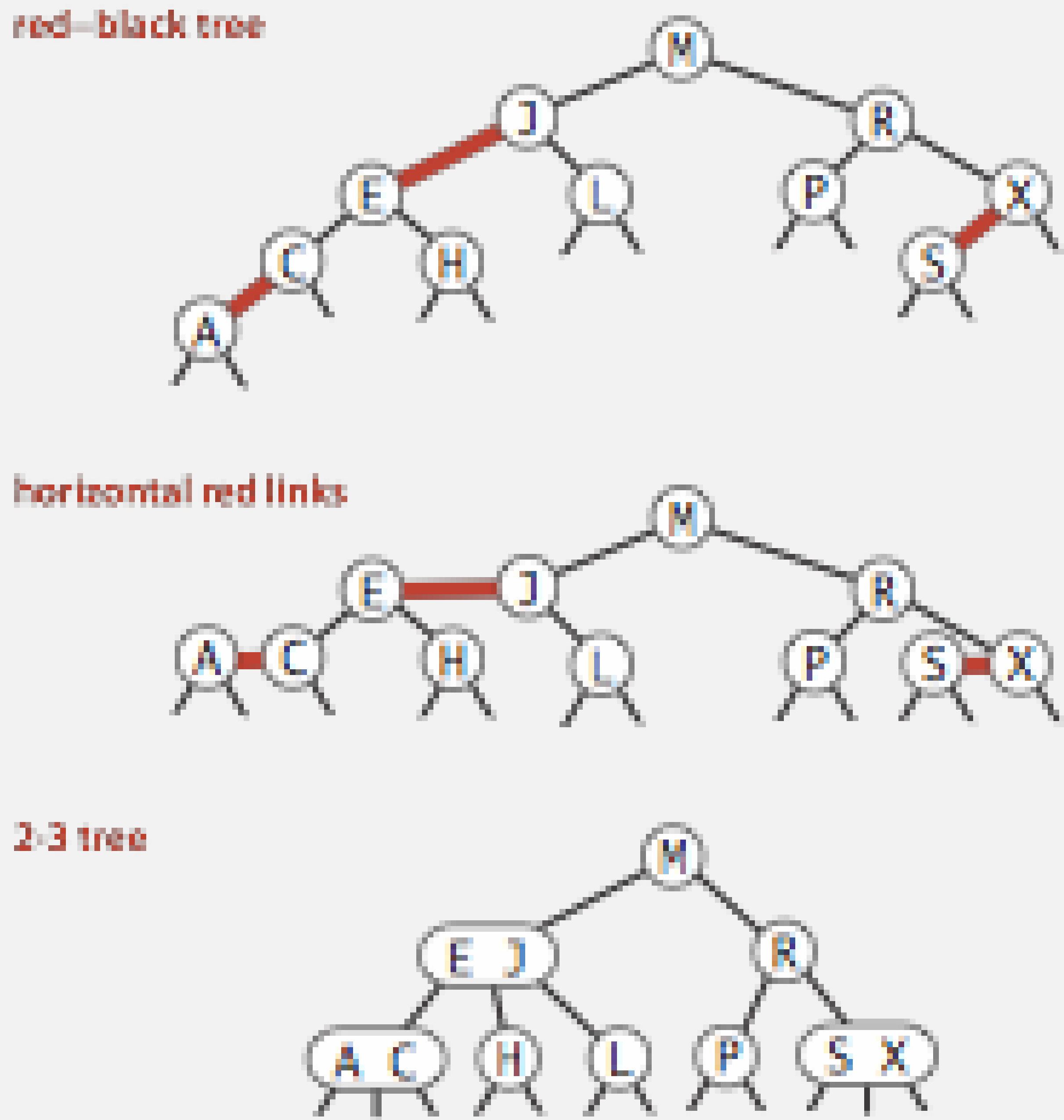
Rich Little
rlittle@uvic.ca

Red-black trees and 2-3 trees

- There is a 1-1 correspondence between red-black BSTs and 2-3 trees.
- To see this, imagine that red links are collapsed: the collapsed nodes correspond to 3-nodes, all others to 2-nodes.

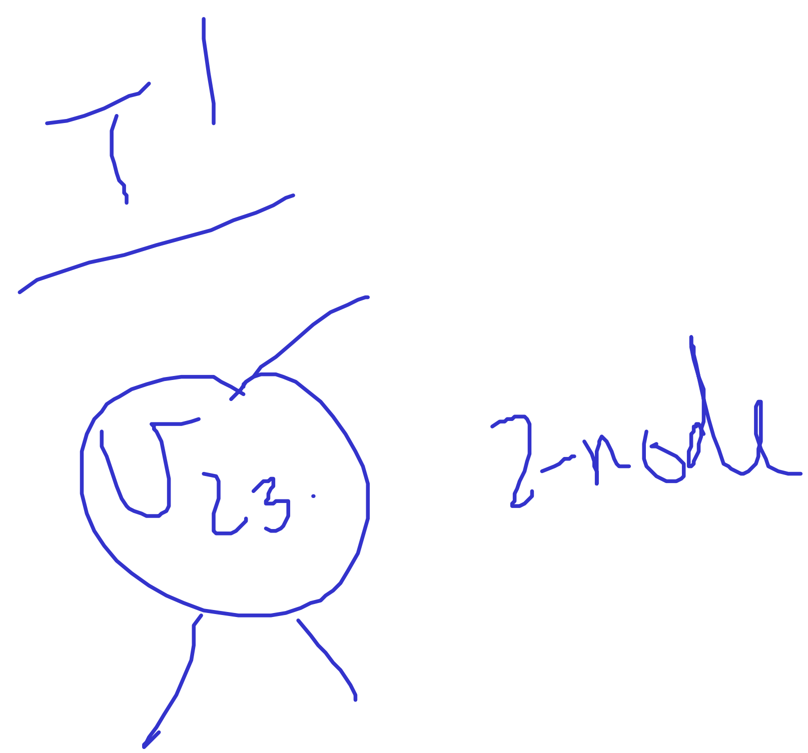
Left-leaning red-black BSTs: 1-1 correspondence with 2-3 trees

Key property. 1-1 correspondence between 2-3 and LLRB.



Theorem: For every red-black tree there is a 2-3 tree

- Given a red-black tree T, we build a 2-3 tree T'
- The nodes of the 2-3 tree T' are obtained as follows.
    - for every node v_{rb} in the red-black tree with key k that is incident to black edges only create a 2-node v_{23} for the 2-3 tree with key k. That is: $23(v_{rb}) = v_{23}$



Theorem: For every red-black tree there is a 2-3 tree

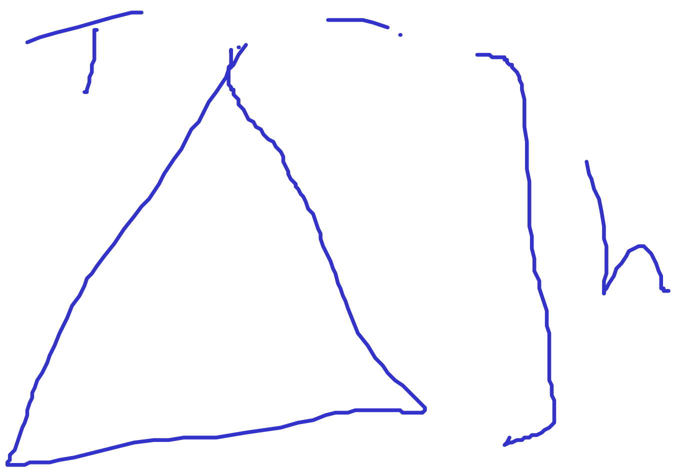
- Given a red-black tree T, we build a 2-3 tree T'
- The nodes of the 2-3 tree T' are obtained as follows.
  - for every red edge and its incident two nodes v_{rb} and w_{rb}—where w_{rb} is the parent of v_{rb}—containing keys k_1 and k_2, respectively, create a 3-node vw_{23} containing keys k_1 and k_2 (with k_1 being the left entry). That is: $23(v_{rb}) = 23(w_{rb}) = vw_{23}$

Theorem: For every red-black tree there is a 2-3 tree

- The edges of the 2-3 tree T' are obtained as follows
  - Let $u_{rb}v_{rb}$ be a **black** edge in the red-black tree. Then $23(u_{rb})23(v_{rb})$ is an edge in the 2-3 tree. Further,

    - if u_{rb} is the left child of v_{rb} then $23(u_{rb})$ is the left child of $23(v_{rb})$
    - else if u_{rb} is the right child of v_{rb}, and the edge between v_{rb} and it's parent is red, then $23(u_{rb})$ is $23(v_{rb})$'s middle child
    - else $23(u_{rb})$ is the right child of $23(v_{rb})$

Theorem: The height, h, of a red-black tree with n keys is $O(\log(n))$



with n keys, i.e. n internal nodes. Let $n(d)$ be the min. number of internal nodes in a red-black tree of black-depth d.

$d=1$: $n(d)=1$

$d=2$: $n(d)=3$

$d=3$: $n(d)=7$

black-depth d,

$n(d)=2^d-1$ internal nodes

Theorem: The height, h, of a red-black tree with n keys is $O(\log(n))$

Any red-black tree with n int. nodes has at least, n(d) nodes if black-depth is d.

$$n \geq n(d) = 2^d - 1$$

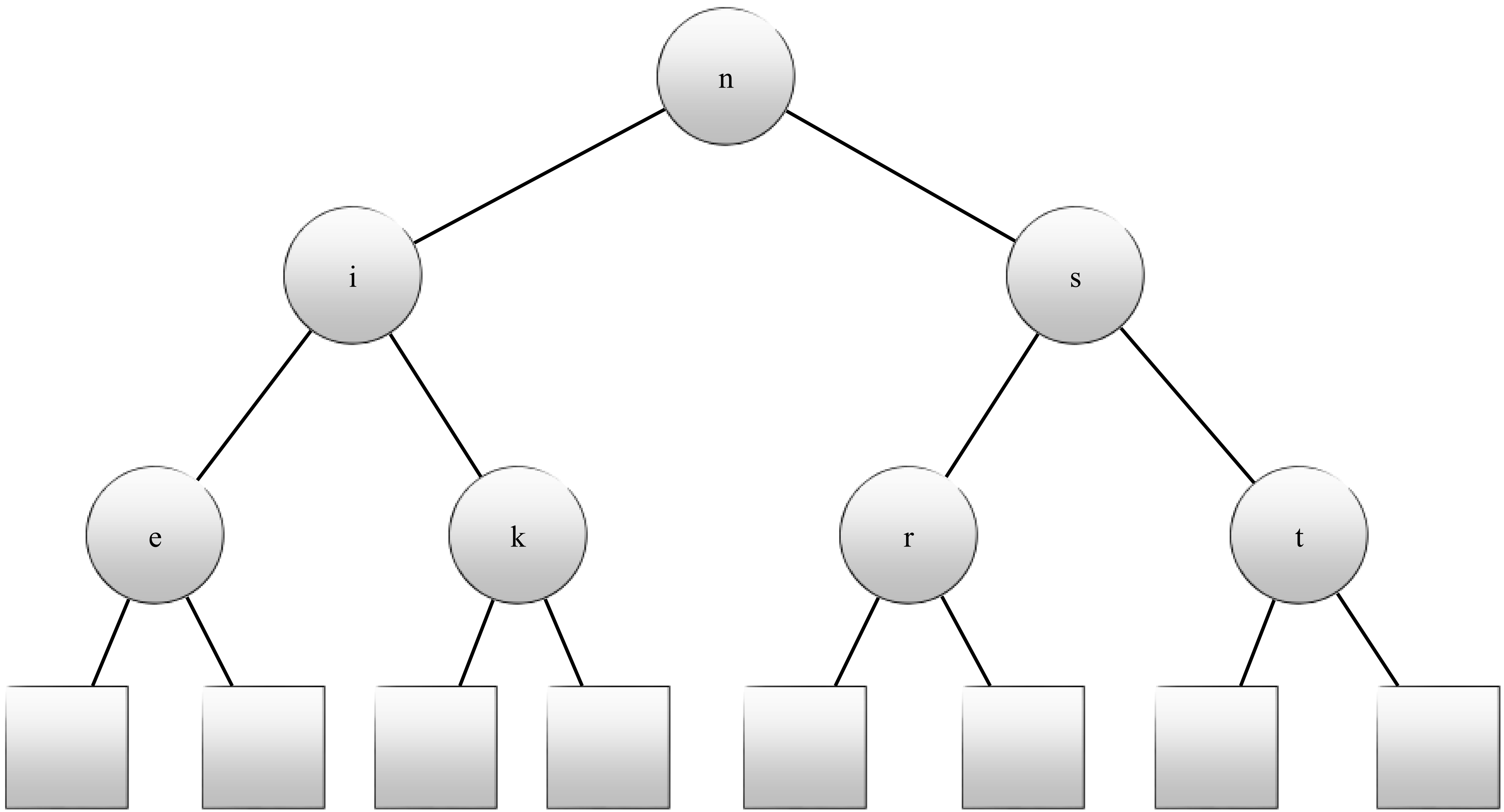
$$n+1 \geq 2^d$$

$$h \leq 2d \leq 2\log(n+1) \in O(\log n)$$

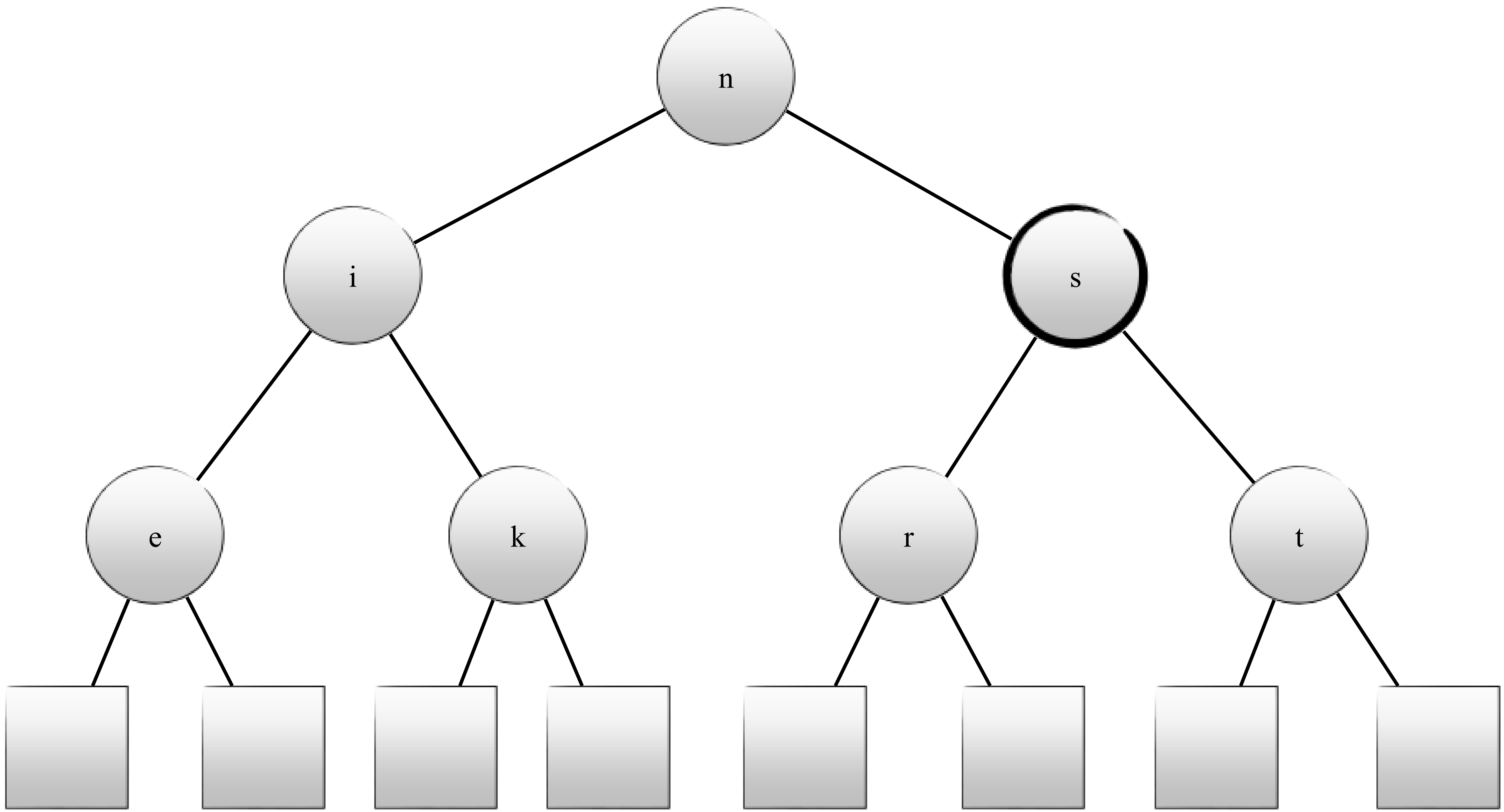
Deletion from 2-3/red-black trees

- Deletion is more problematic than insertion. Why?
- During insertion we can only cause problems with the colouring of the nodes
  - i.e. violate the left red and/or one red edge properties.
- During deletion we can cause black-depth imbalance as well.

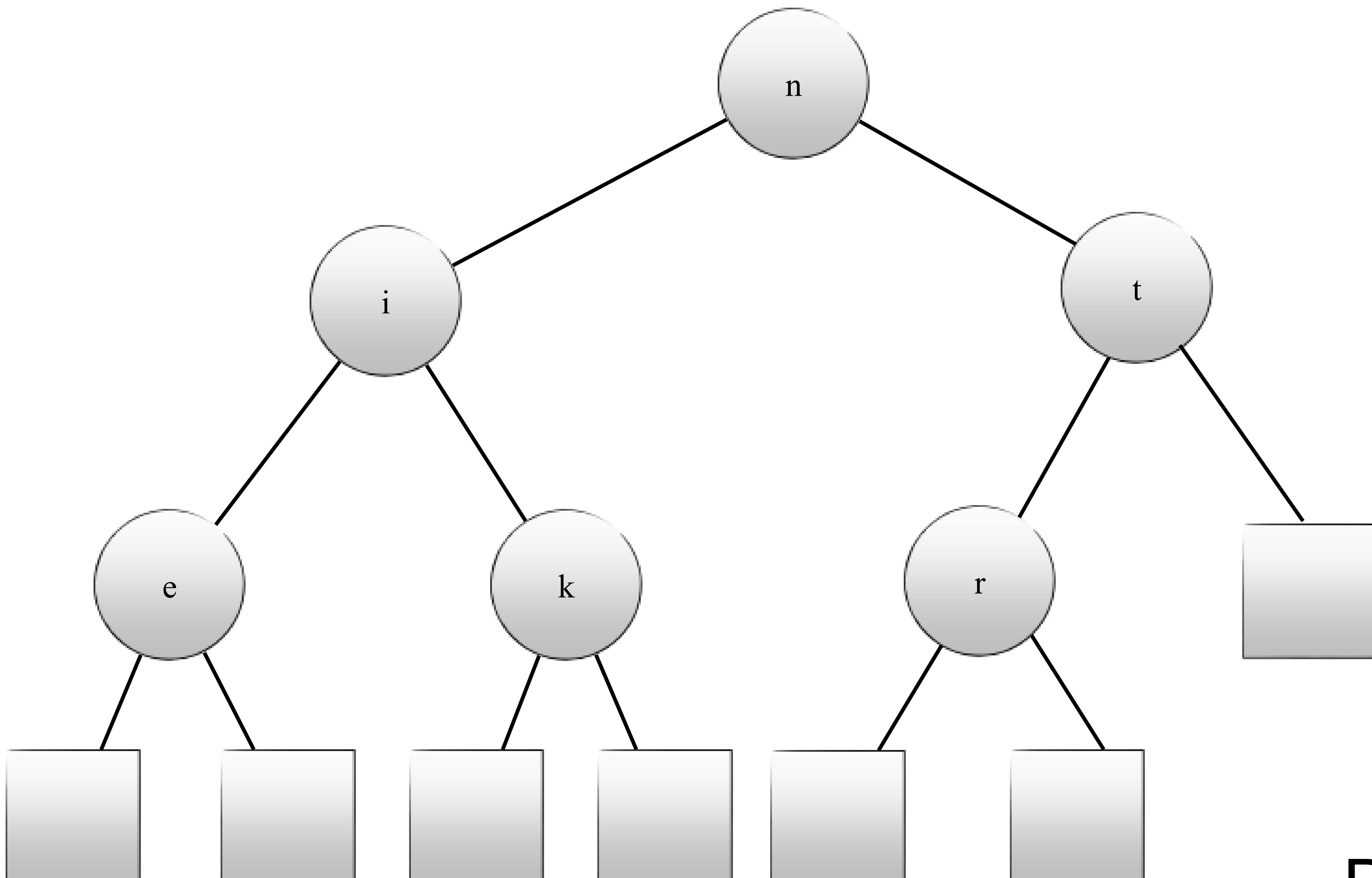
Delete key *s*



Delete key s

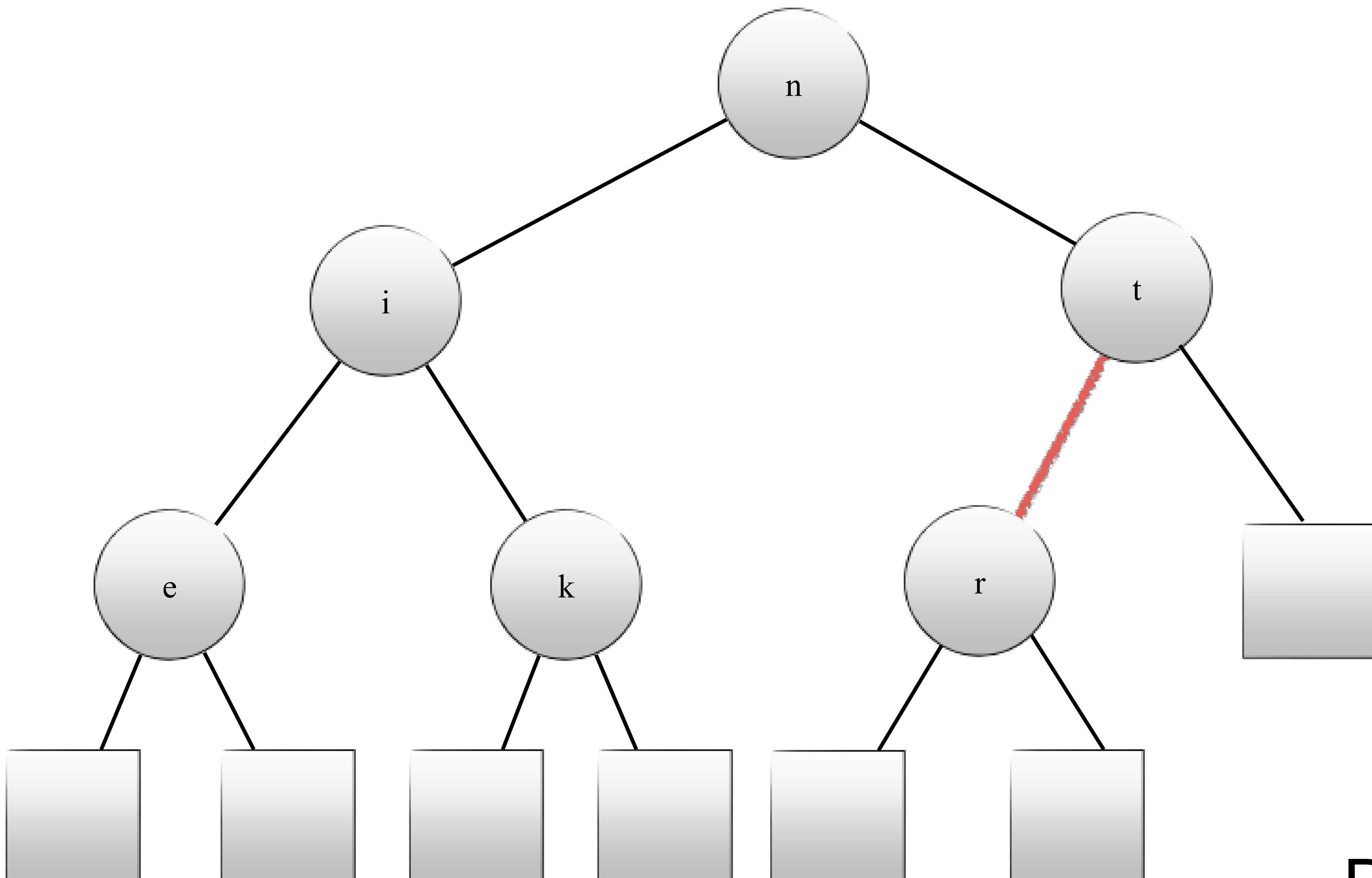


Delete key s



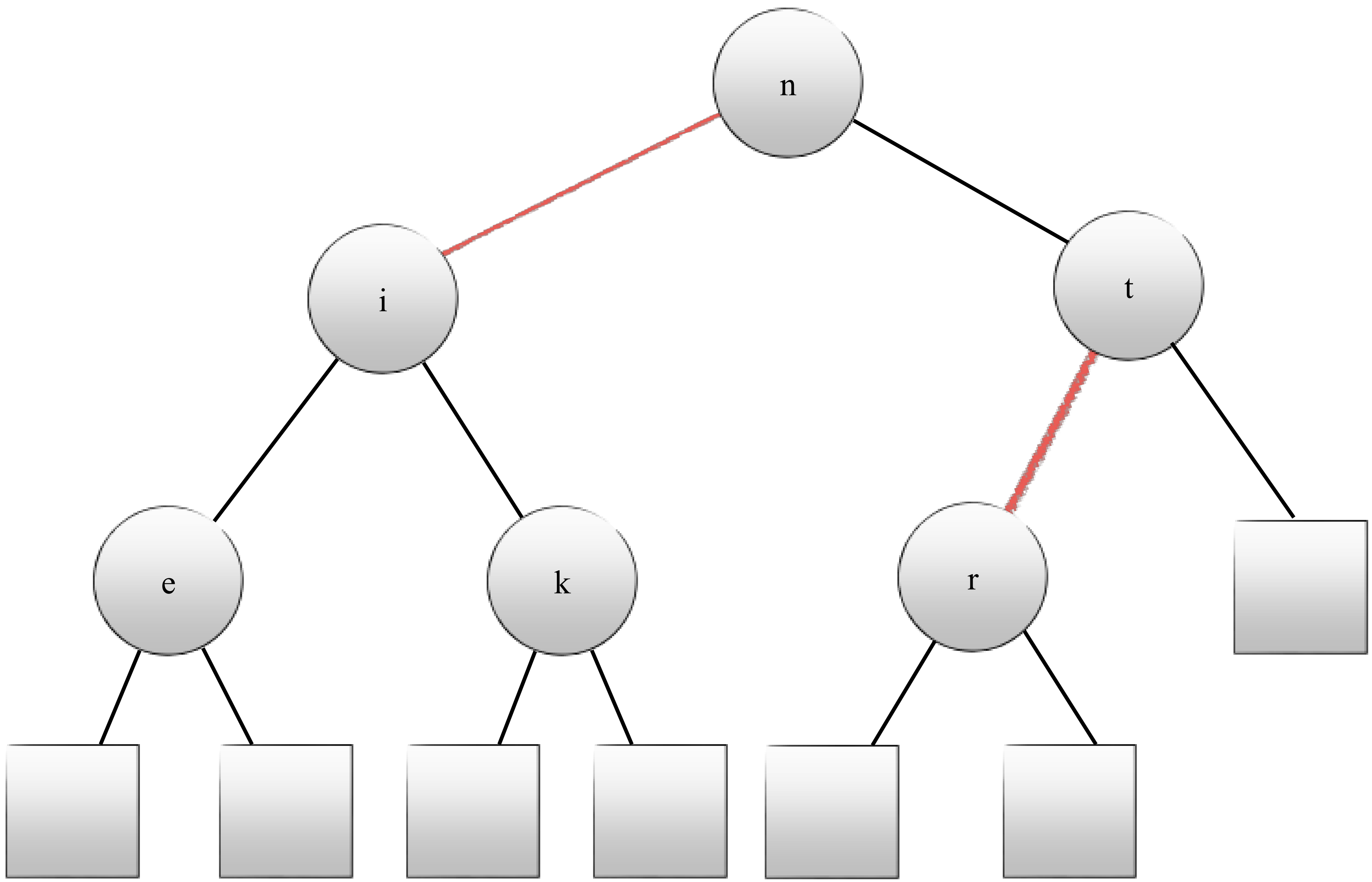
Doesn't work

Delete key s



Doesn't work

Delete key s

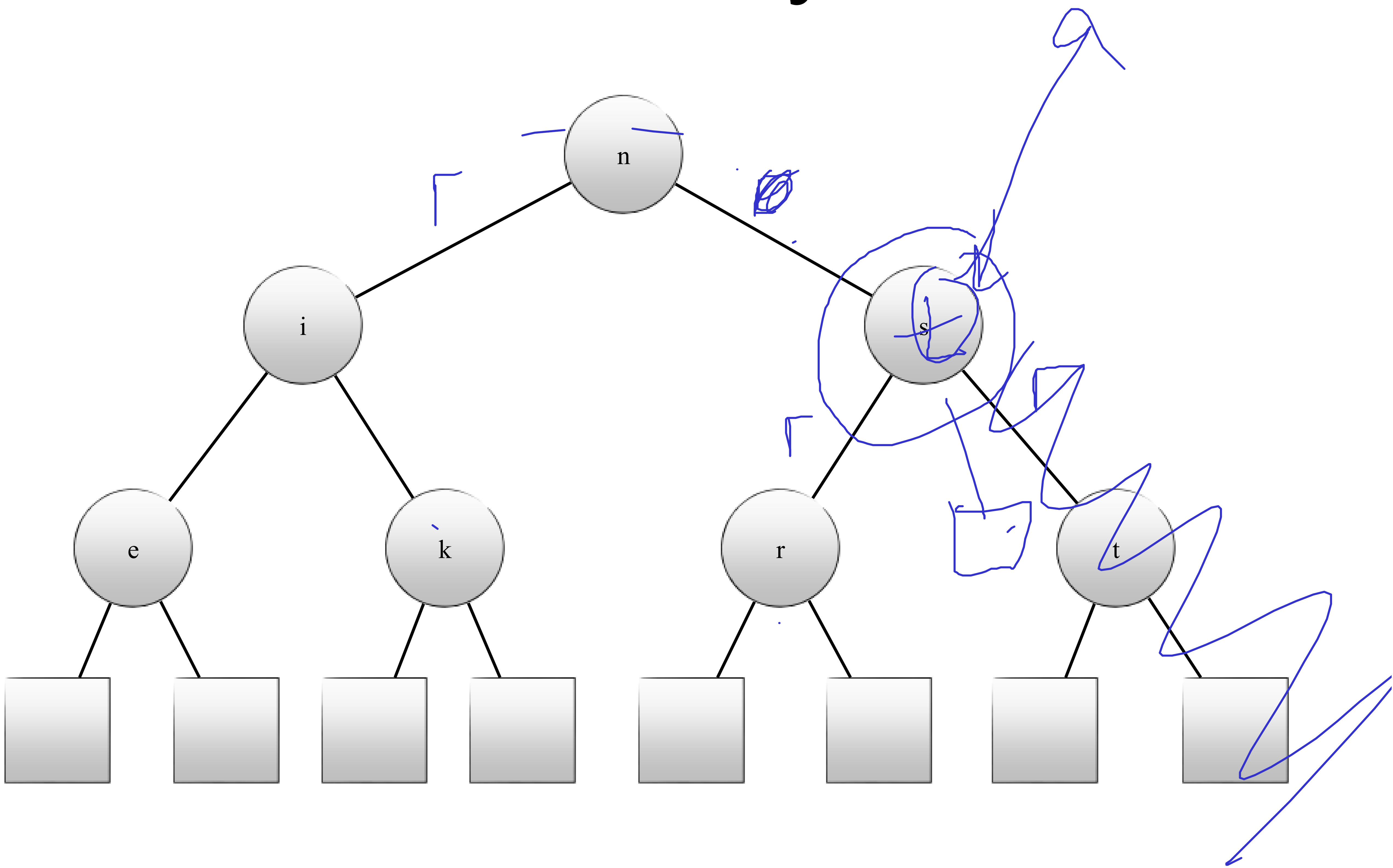


We need!

Deletion from 2-3/red-black trees

- You need to account for a potential imbalance on the way “down”
- So, we rotate and/or color flip both on the way down the recursion as well as on the way up.
- Generally, we make 2-nodes into 3-nodes or 4-nodes on the way down to accommodate a removal then correct the red edges on the way back up if needed.

Delete key *s*



Red-black Tree History

- In 1972, Rudolf Bayer[5] invented a data structure that was a special order-4 case of a B-tree. These trees maintained all paths from root to leaf with the same number of nodes, creating perfectly balanced trees. However, they were not *binary* search trees. Bayer called them a "symmetric binary B-tree" in his paper and later they became popular as 2–3–4 trees or just 2–4 trees.[6]

- In a 1978 paper, "A Dichromatic Framework for Balanced Trees",[7] Leonidas J. Guibas and Robert Sedgewick derived the red–black tree from the symmetric binary B-tree.[8] The color "red" was chosen because it was the best-looking color produced by the color laser printer available to the authors while working at Xerox PARC.[9] Another response from Guibas states that it was because of the red and black pens available to them to draw the trees.[10]

- In 1993, Arne Andersson introduced the idea of a right leaning tree to simplify insert and delete operations.[11]

- In 1999, Chris Okasaki showed how to make the insert operation purely functional. Its balance function needed to take care of only 4 unbalanced cases and one default balanced case.[12]

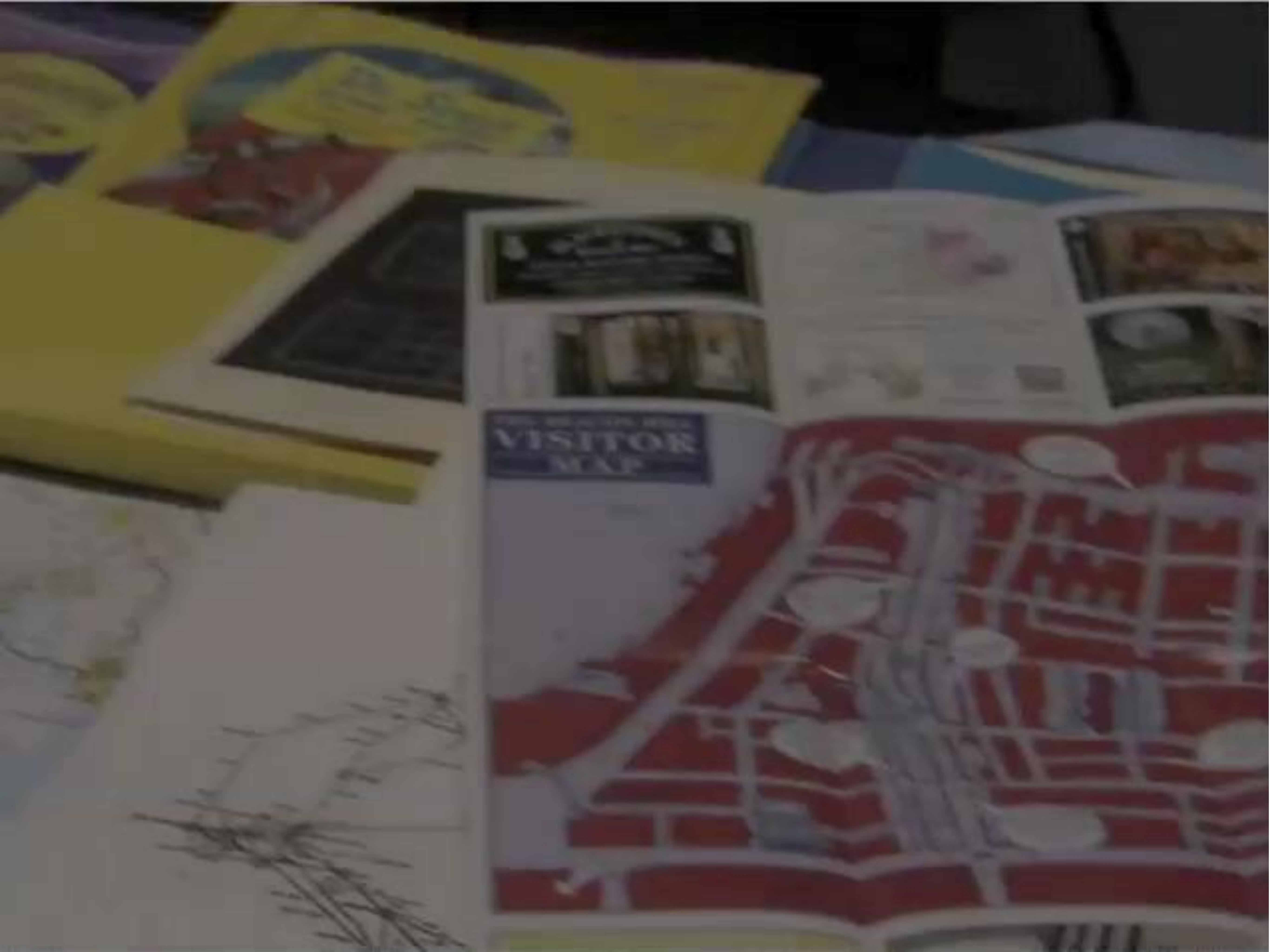
- In 2008, Sedgewick proposed the left-leaning red–black tree, leveraging Andersson’s idea that simplified the insert and delete operations. Sedgewick originally allowed nodes whose two children are red, making his trees more like 2–3–4 trees, but later this restriction was added, making new trees more like 2–3 trees. Sedgewick implemented the insert algorithm in just 33 lines, significantly shortening his original 46 lines of code.[15][16]

Red-Black trees in the wild

Red-black trees are widely used as system symbol tables.

- Java: java.util.TreeMap, java.util.TreeSet.
- C++ STL: map, multimap, multiset.
- Linux kernel: completely fair scheduler, linux/rbtree.h.
- Emacs: conservative stack scanning.

Common sense. Sixth sense.
Together they're the
FBI's newest team.


VISITOR
MAP

Red-black BSTs in the wild

ACT FOUR

FADE IN:

48 INT. FBI HQ - NIGHT 48

Antonio is at THE COMPUTER as Jess explains herself to Nicole and Pollock. The CONFERENCE TABLE is covered with OPEN REFERENCE BOOKS, TOURIST GUIDES, MAPS and REAMS OF PRINTOUTS.

JESS
It was the red door again.

POLLOCK
I thought the red door was the storage container.

JESS
But it wasn't red anymore. It was black.

ANTONIO
So red turning to black means... what?

POLLOCK
Budget deficits? Red ink, black ink?

NICOLE
Yes. I'm sure that's what it is. But maybe we should come up with a couple other options, just in case.

Antonio refers to his COMPUTER SCREEN, which is filled with mathematical equations.

ANTONIO
It could be an algorithm from a binary search tree. A red-black tree tracks every simple path from a node to a descendant leaf with the same number of black nodes.

JESS
Does that help you with girls?