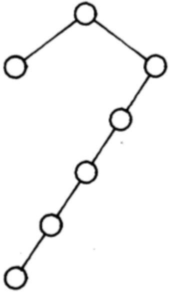
Drawing Rooted Trees . . .

Aditya Siram (@deech)

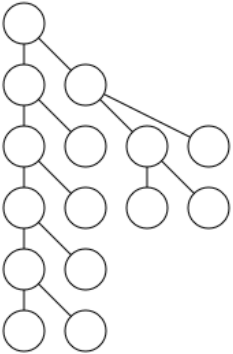
February 15, 2016

Outline

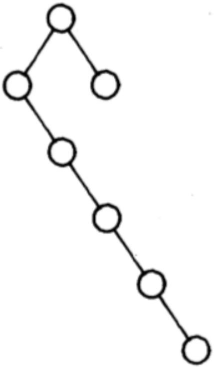
Ugly Tree



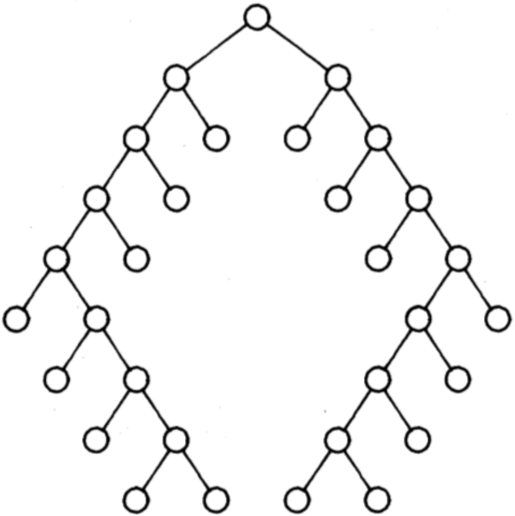
Ugly Tree



Nice Tree

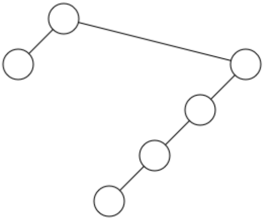


Nice Tree



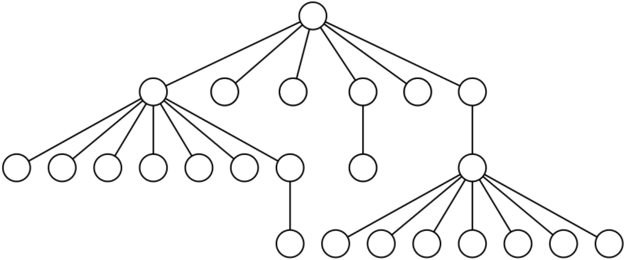
Knuth

- Knuth (1971)



- 'nuf said.

The Big Picture



- Quadratic is easy
  - Update every node in case of conflicts
  - Undergraduate exercise
- Linear is *hard*
  - 25 years (1981 - 2006)
  - 33 years with Knuth (1971 - 2006)

The Big Picture

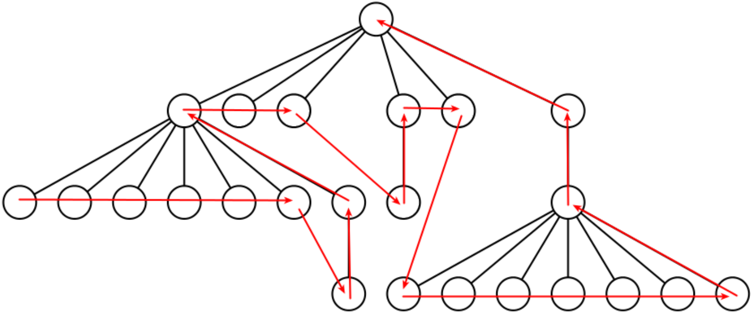
- Two major players
- Walker (1989)
- Buccheim (2006)
  - Assisted by Reingold (1981)

The Big Picture

- Two passes
- First pass sets X coordinates naively, records conflicts.
- Second pass updates X coordinates accounting for conflicts.

The Big Picture

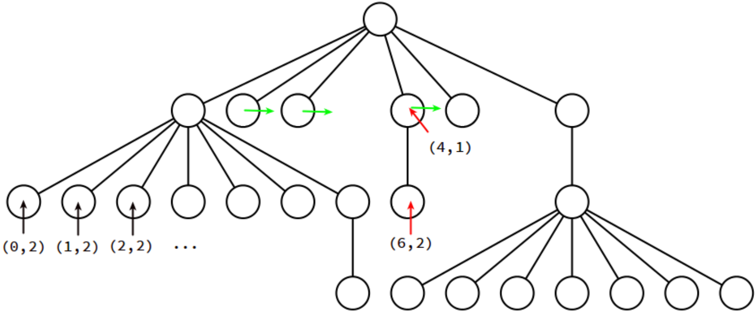
- Do a post order traversal



- This is an ugly tree

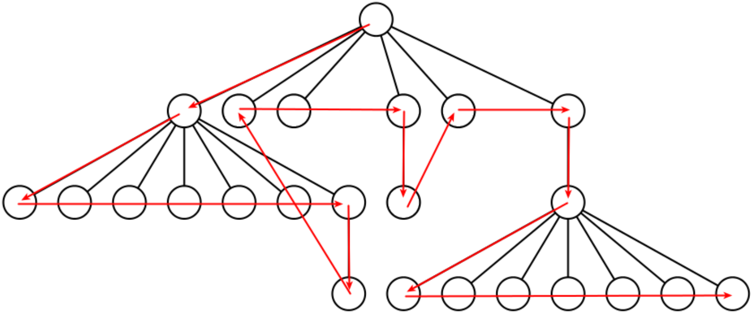
The Big Picture

- First pass
- Save each node's X
- Record shifts (green arrows) but don't execute



The Big Picture

- Pre-order second pass

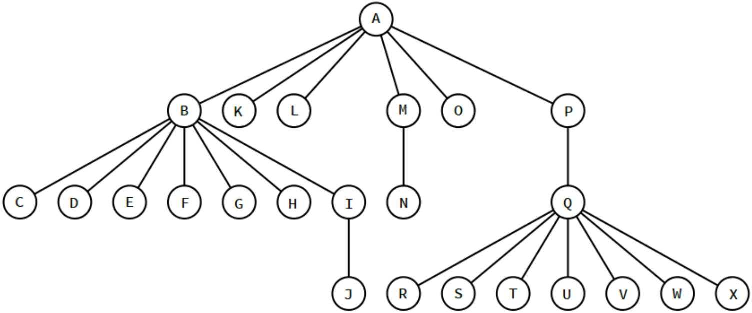


- Add shifts to X-coordinate

Walker

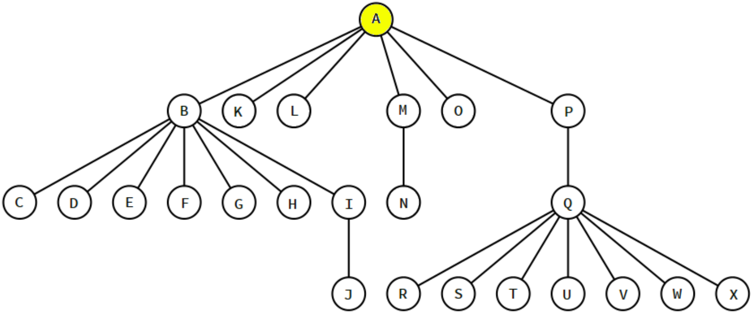
- Left neighbors

  []



- Left neighbors

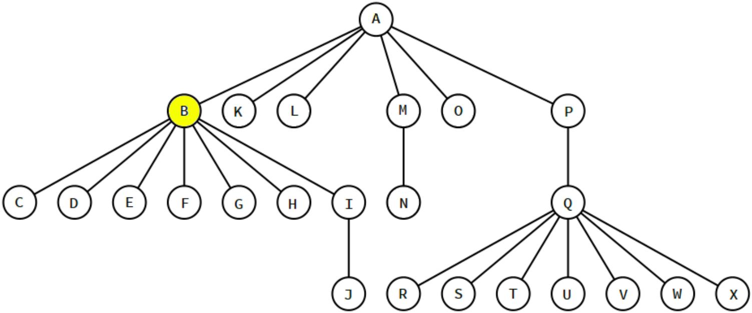
  [A]



Walker

- Left neighbors

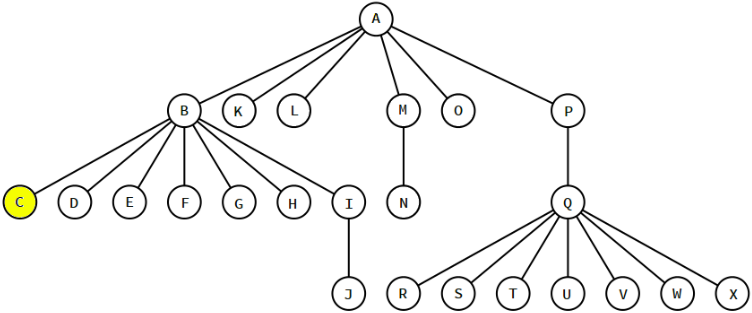
  [A,B]



Walker

- Left neighbors

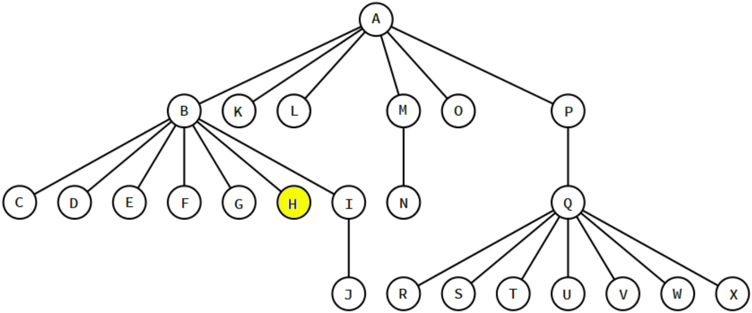
  [A,B,C]



Walker

- Left neighbors

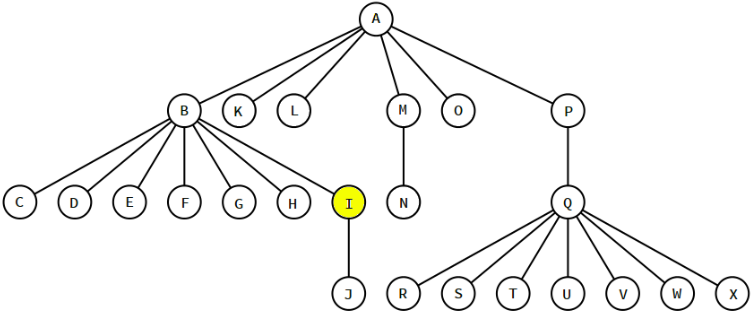
  `[A,B,H]`



Walker

- Left neighbors

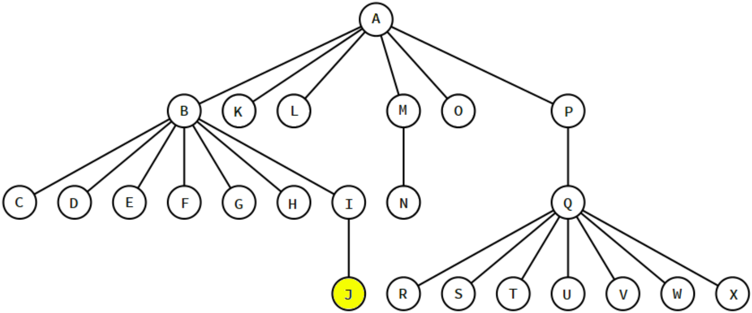
  `[A,B,I]`



Walker

- Left neighbors

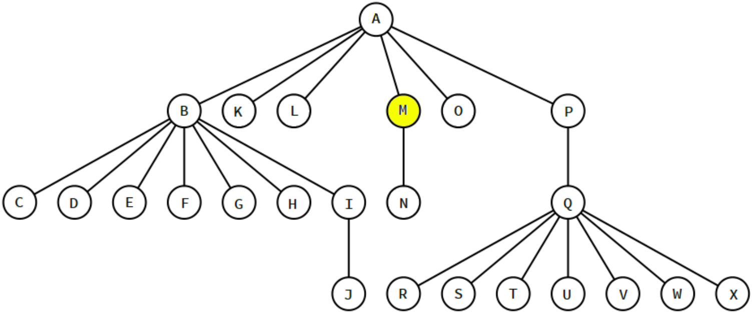
  [A,B,I,J]



Walker

- Left neighbors

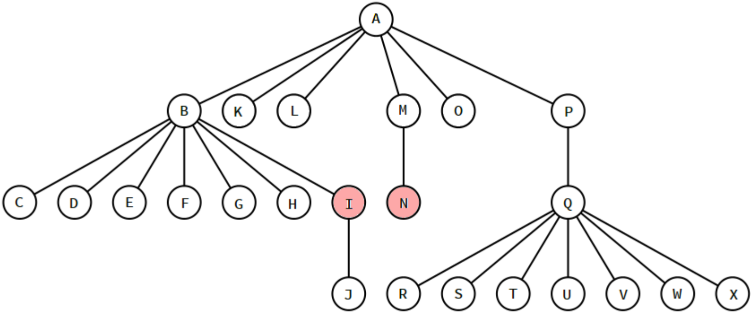
  [A,M,I,J]



Walker

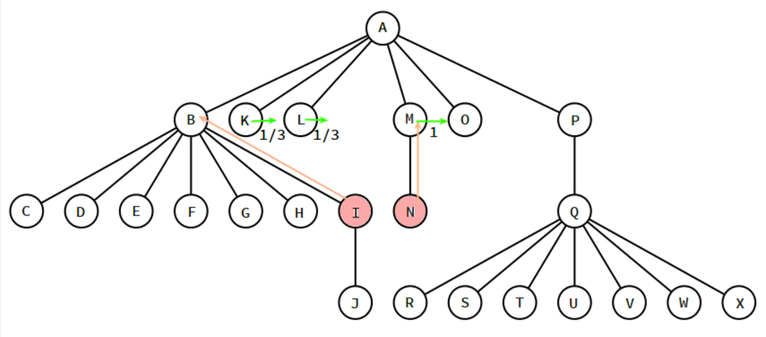
- Left neighbors

```
[A,M,N,J] (* conflicts with I *)
```



Walker

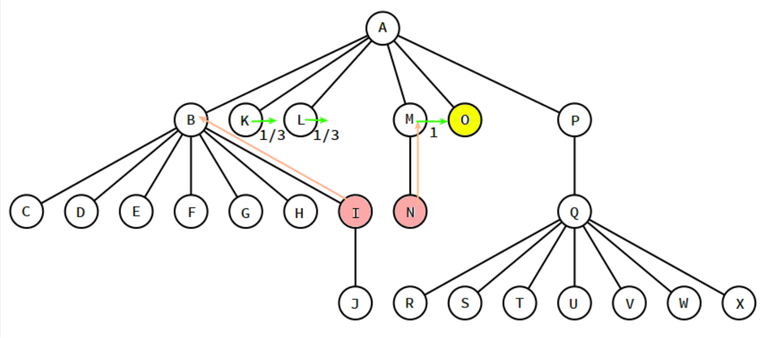
- APPORTION
- Track back to greatest common ancestors
- Record shifts in siblings between conflicting roots



Walker

- Left neighbors

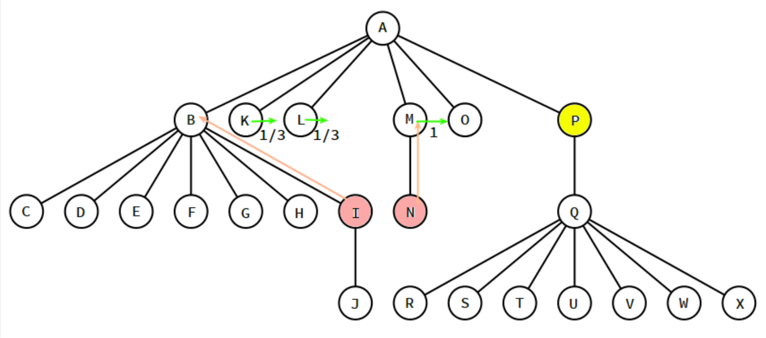
  [A,O,N,J]



Walker

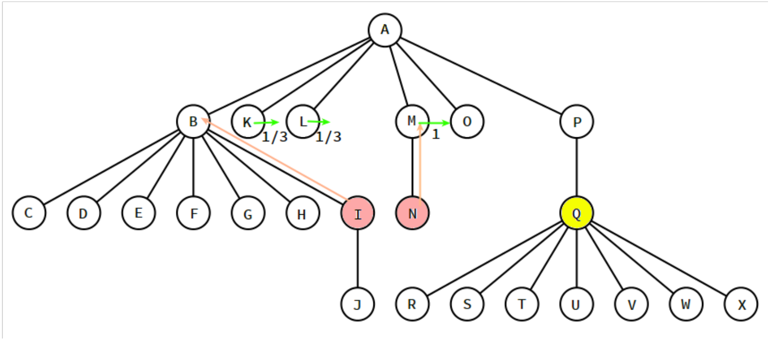
- Left neighbors

  `[A,P,N,J]`



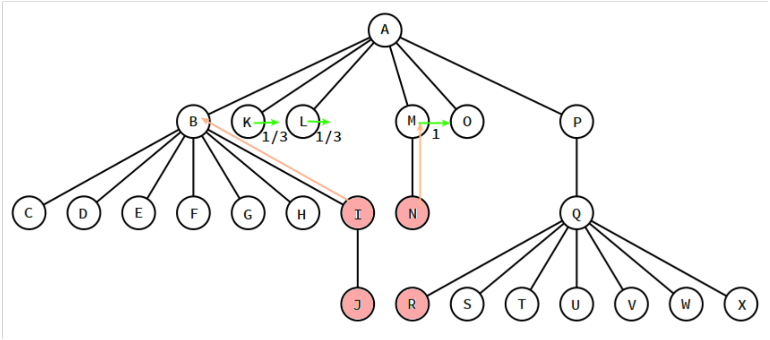
- Left neighbors

  [A,P,Q,J]



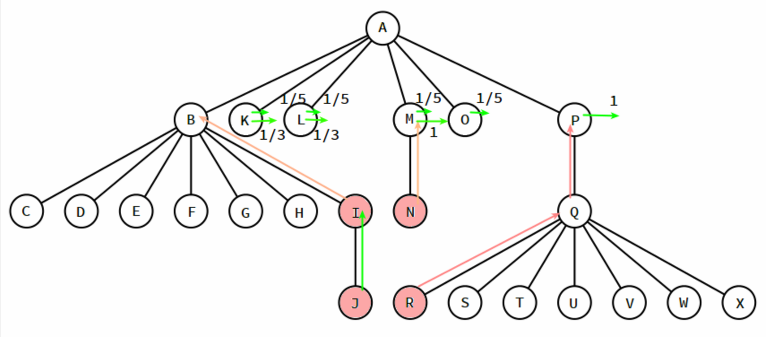
- Left neighbors

  [A,P,Q,R] (* conflicts with J *)



Walker

- APPORTION
- Track back to greatest common ancestors and shift
- Shifts are accumulated!



Walker

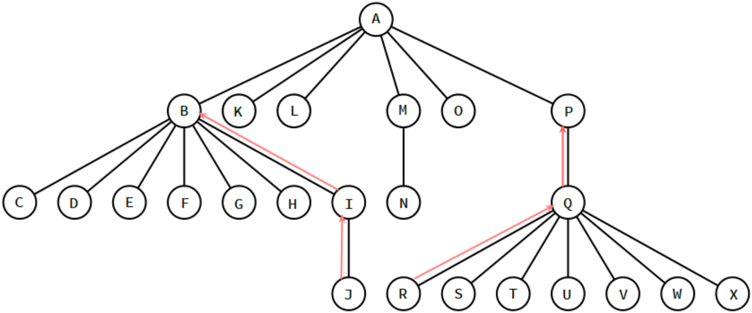
- Second pass, pre-order
- Shifts propogate to subtrees
- Pretty tree

Walker

- Not linear!
- Discovered (and fixed) by Buccheim in *2006*!
    - 16 years
- *All* the problems are in APPORTION.

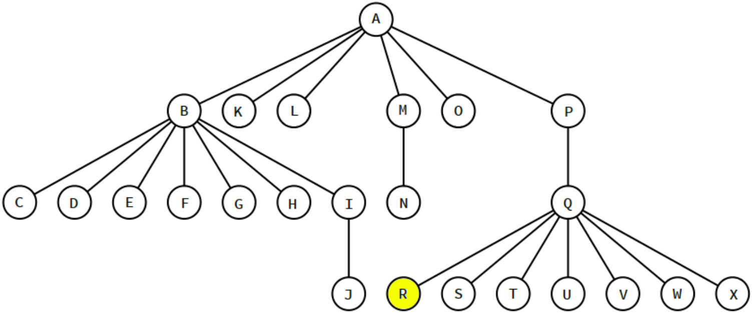
Buccheim

- Finding common ancestors is not linear!



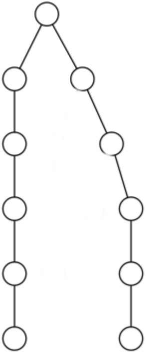
Leftmost node blowup

- Finding the leftmost node is also not linear
- Not as obvious
- Buccheim shows a *pattern* of trees that cause quadratic blowup.



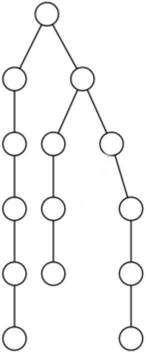
Leftmost node blowup

- Pick a k, say, 3
  - Hang two chains of (2 * k) from root.



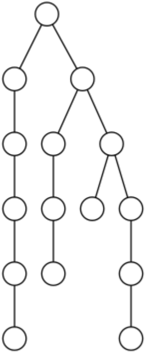
Leftmost node blowup

- (k - (current level))-th odd number chains on right
    - current level = 1, k = 3, (3 - 1)th odd number = 3



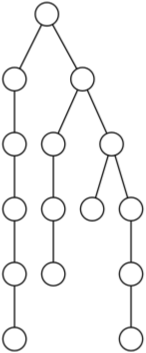
Leftmost node blowup

- (k - (current level))-th odd number chains on right
  - current level = 2, k = 3, (3 - 2)th odd number = 1



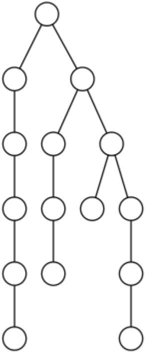
Leftmost node blowup

- Total nodes = sum(k even numbers) + k
    - $6 + 4 + 2 + 3 = 15$



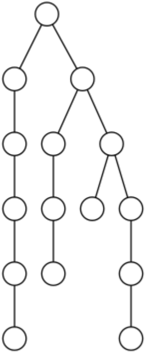
Leftmost node blowup

- Total nodes = k * (k + 1) + k
  - 3 * (3 + 1) + 3 = 15



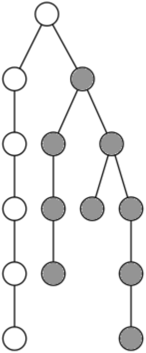
Leftmost node blowup

- Total nodes = $k^2 + k + k = \theta(k^2)$
  - $3 * (3 + 1) + 3 = 15$



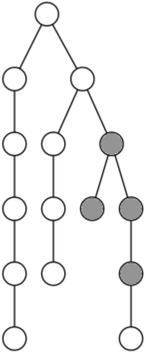
Leftmost node blowup

- Number of comparisons
  - level 1 = 9



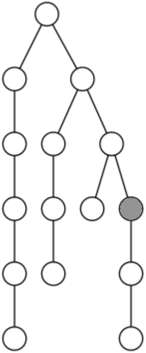
Leftmost node blowup

- Number of comparisons
  - level 2 = 4



Leftmost node blowup

- Number of comparisons
  - level 3 = 1

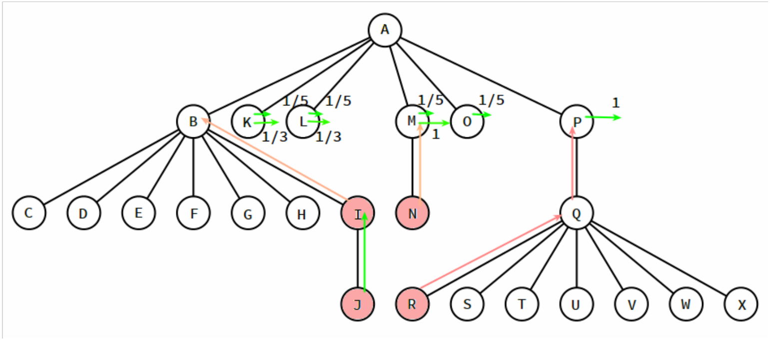


Leftmost node blowup

- Total # of calls
  - $1 + 4 + 9$
- sum of k squares
  - $k^3/3 + k^2/2 + k/6$
  - $\theta(k^3)$
- nodes, k^2
- calls, k^3
- k, $nodes^{(1/2)}$
- $\Omega(k^3) = \Omega(nodes^{(3/2)})$

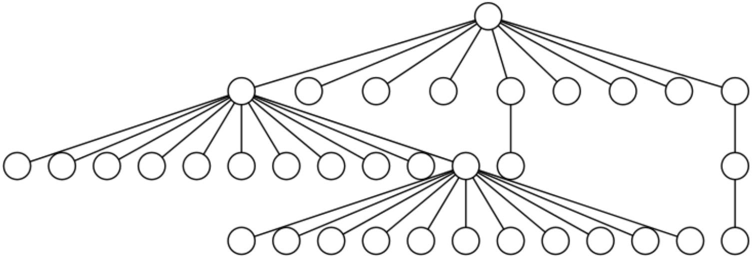
Shifting blowup

- Shifting is also not linear!
- Again, Buccheim shows a pattern of trees



Shifting blowup

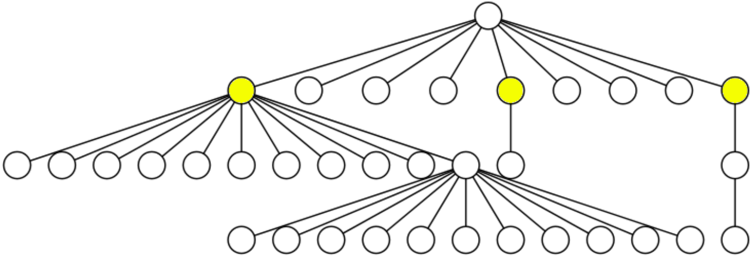
- Pick a k, say 3



- Will cause quadratic shifting

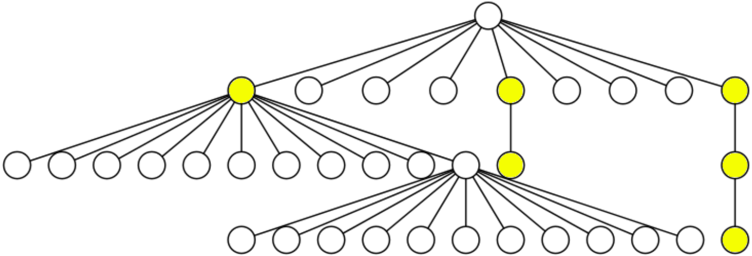
Shifting blowup

- Hang k nodes from root



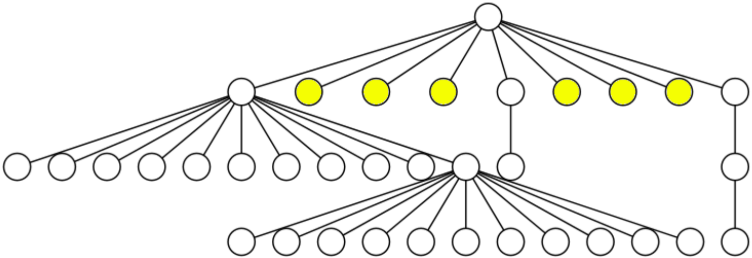
Shifting blowup

- Each of the k nodes is a chain k nodes high



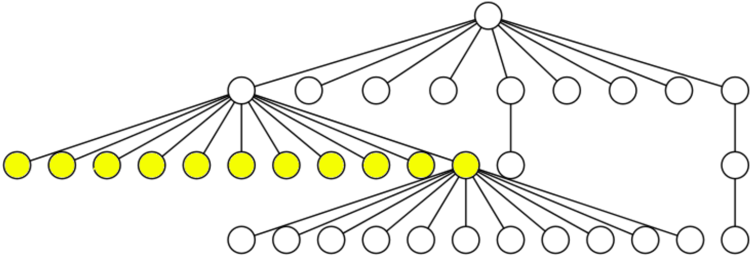
Shifting blowup

- Add k nodes in between those nodes



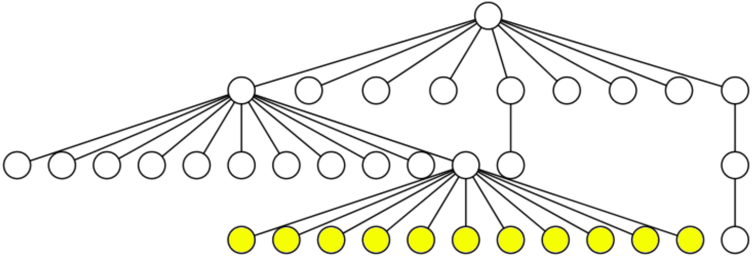
Shifting blowup

- Add 2k + 5 children to leftmost node



Shifting blowup

- Add 2k + 5 nodes to rightmost of leftmost upto k - 1



Shifting blowup

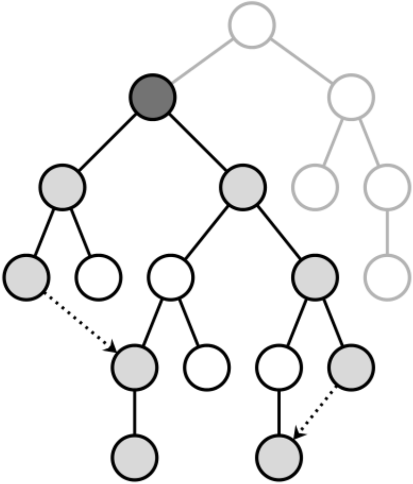
- Again shows k^2 nodes and k^3 runtime
- Similar procedure as before
- Omitted for time!

Contours

- Ancestor and leftmost node problem fixed by contours
- Introduced by Reingold (1981)
- Maintains the *shape* of the tree outlines, not left/rightmost nodes

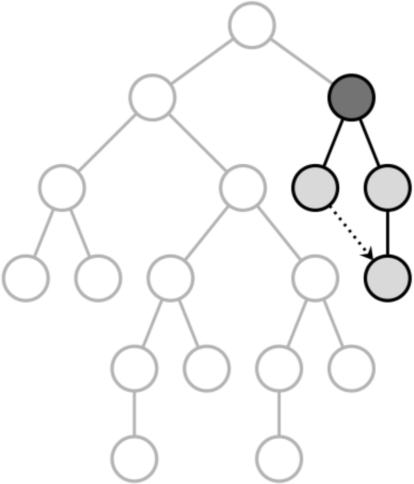
Contours

- Every inner/outer leaf stores a link to left/rightmost child
- Just re-uses the same space for left/right nodes



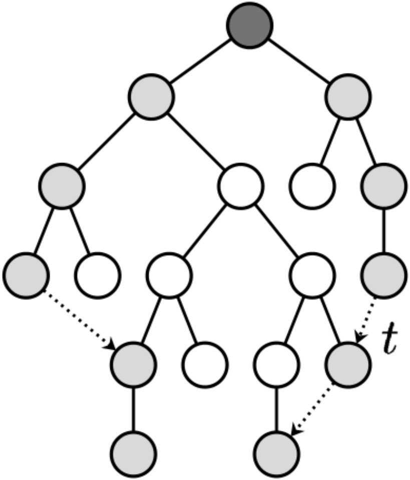
Contours

- Adding new subtree is added, it's inner links are now useless.



Contours

- Adds a link the rightmost child of the left tree
- If the right tree were bigger the left outer leaf adds a link



Contours

- At most 1 link added per subtree combination - linear!
- To get the leftmost, just follow the left link - linear!

Ancestors

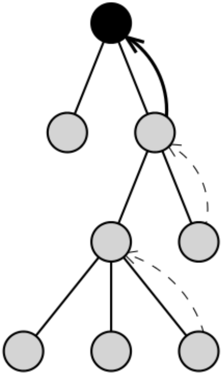
- Finding the common ancestors in linear time
- The ancestor is the right subtree is just the root

Ancestors

- Each node points to it's parent
- Also a pointer called *defaultAncestor*
- If parent isn't the common ancestor, default to *defaultAncestor*

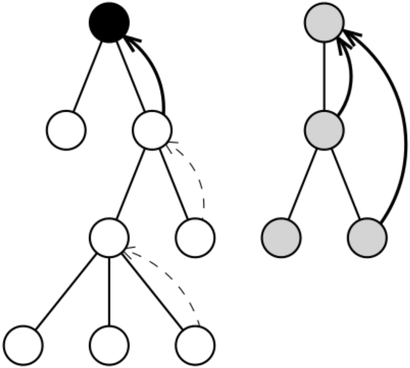
Ancestors

- Leftmost subtree set *defaultAncestor* to root.
- Solid arrows are correct links, dashed are expired links



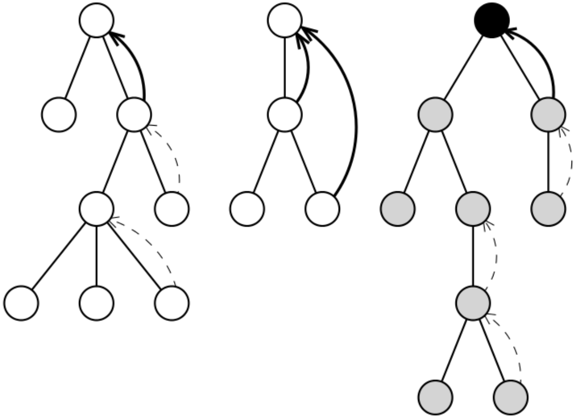
Ancestors

- When right is smaller, right contour is updated
- No expired links, always point to the correct ancestor



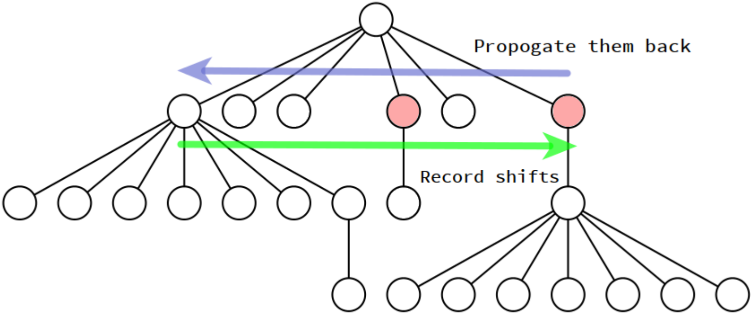
Ancestors

- When right is bigger, *defaultAncestor* now points to it's root
- The right shadows everything before it!

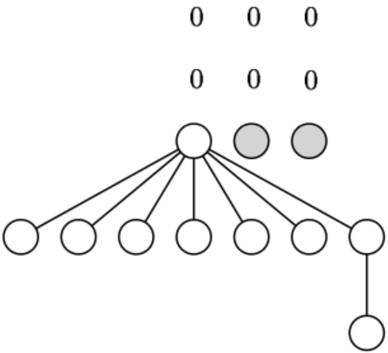


Shifting (left -> right)

- Don't update every intermediate sibling for each conflict!
- Records *shift*, and *change* and aggregate them backwards once last sibling has been reached

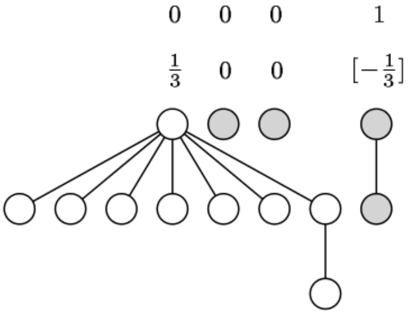


- Maintain pointer to leftmost sibling.
- Top number is *shift*, bottom is *change*.
- No conflicts so far



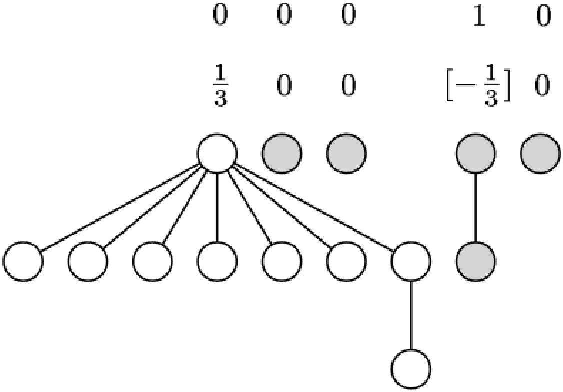
Shifting (left -> right)

- Conflict!
- Shift is 1
- Change is 1/(number of intermediate subtrees + 1) = 1/3
- Add -1/3 on conflict node, add +1/3 to leftmost node
- This is confusing!



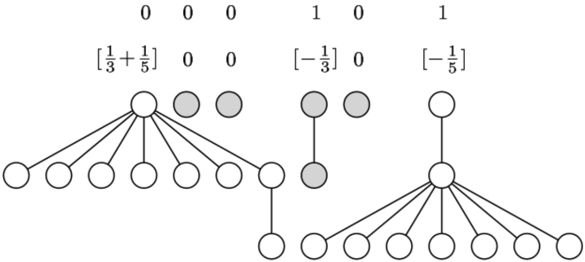
Shifting (left -> right)

- No clashes on the next one.



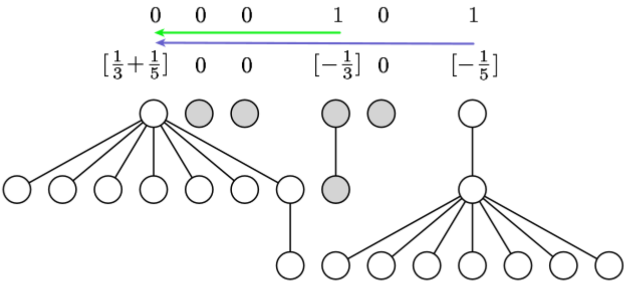
Shifting (left -> right)

- Conflict !
- Shift is 1
- Change is 1/(number of intermediate subtrees + 1) = 1/5
- Add -1/5 on conflict node, *accumulate* +1/5 to leftmost node
- This is still confusing!



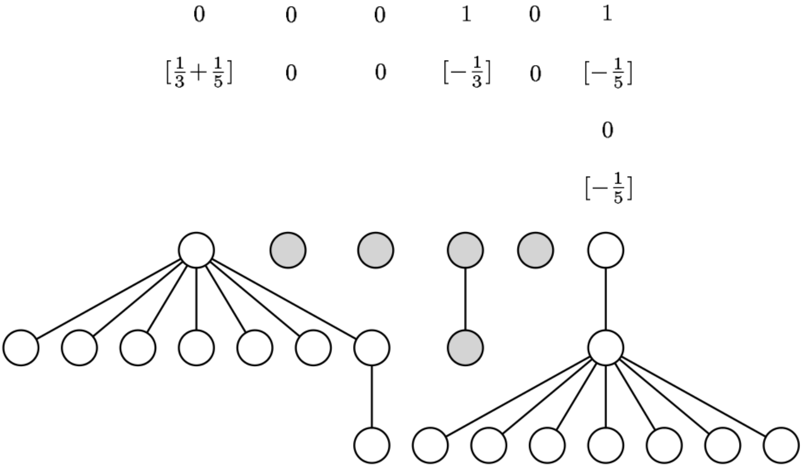
Shifting (taking stock)

- *shift* of 1 from 6th sibling needs to be distributed back by fifths (blue line)
- *shift* of 1 from 4th sibling needs to be distributed back by thirds (green line)



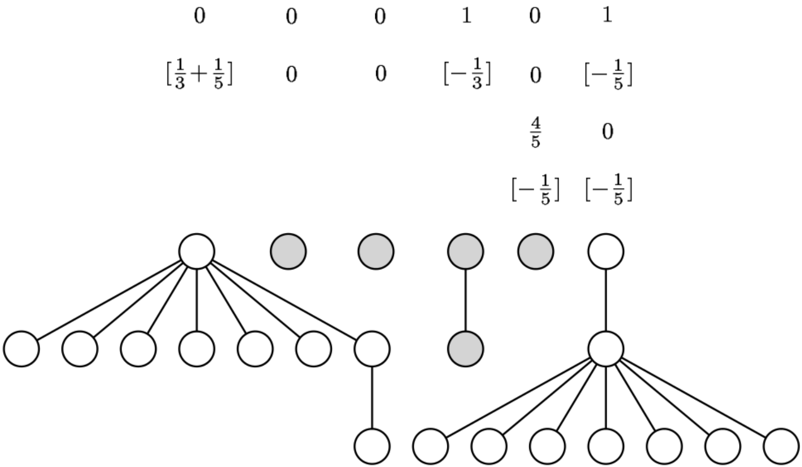
Shifting (right -> left)

- Top two numbers are the same as before
- Bottom two are the new *shift* and propagated *change*.
- Rightmost node does not move, shift is 0



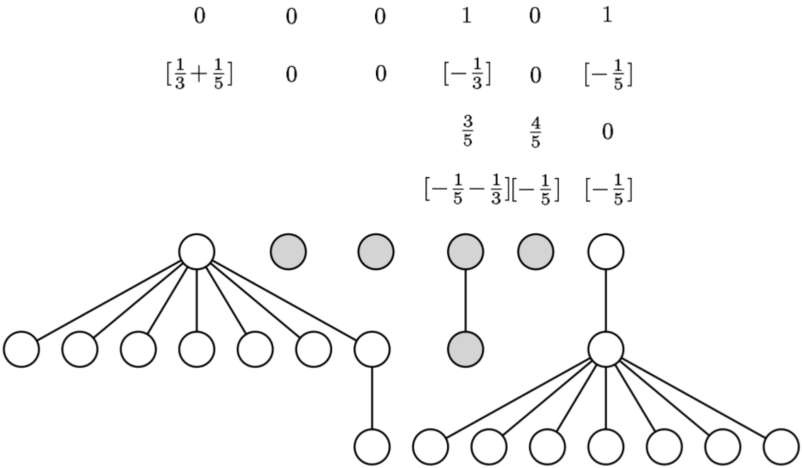
Shifting (right -> left)

- Back one node, shift for this node is 1 - 1/5 = 4/5
- 4/5 propagates back

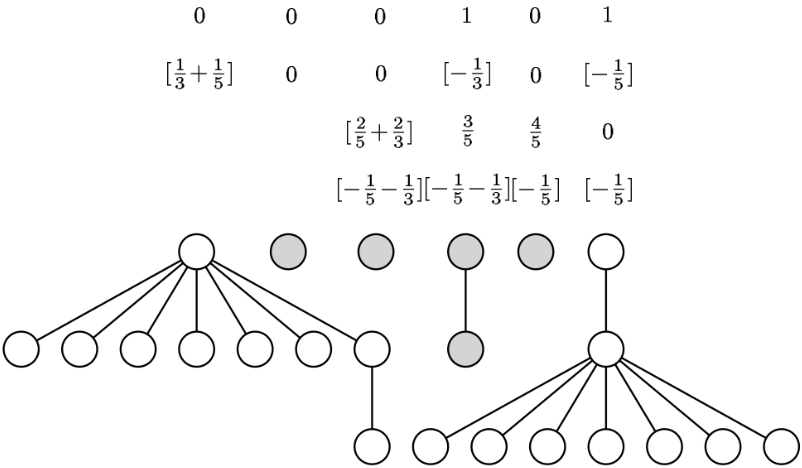


Shifting (right -> left)

- Back another node, shift for this node is 4/5 - 1/5 = 3/5
- We're picking up another shift of 1 & a change of -1/3
- shift of 1 3/5 & change -1/5 - 1/3 propagates back!

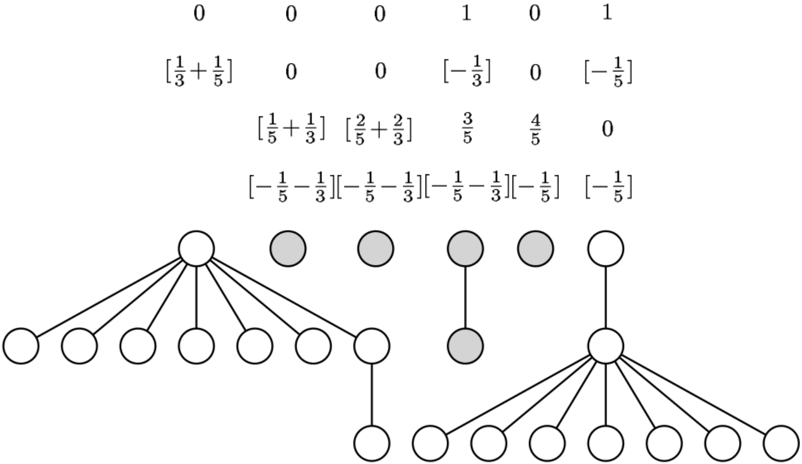


- No conflicts!
- (1 3/5) + (-1/5 - 1/3) = 2/3 + 2/5!
- Shift 2/3 + 2/5 and change (-1/5 - 1/3) propagates back!



Shifting (right -> left)

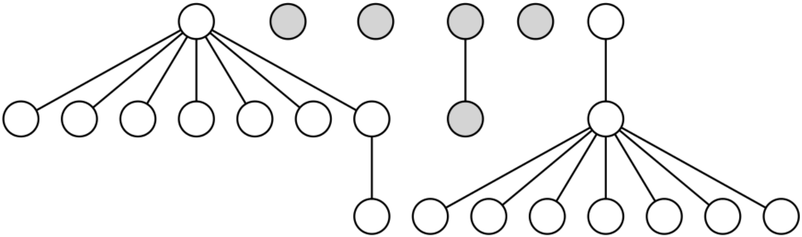
- No conflicts!
- (2/3 + 2/5) + (-1/5 - 1/3) = 1/5 + 1/3
- Shift (1/5 + 1/3), change (-1/5 - 1/3) propagates back!



Shifting (right -> left)

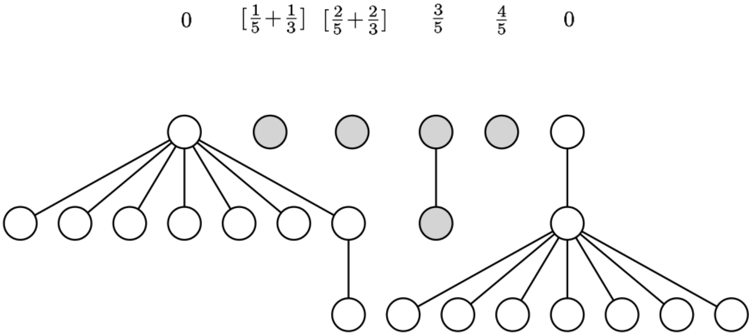
- No conflicts!
- The changes accumulated going left -> right was a stopper!
- (1/5 + 1/3) + (-1/5 - 1/3) = 0!

0	0	0	1	0	1
$[\frac{1}{3}+\frac{1}{5}]$	0	0	$[-\frac{1}{3}]$	0	$[-\frac{1}{5}]$
0	$[\frac{1}{5}+\frac{1}{3}]$	$[\frac{2}{5}+\frac{2}{3}]$	$\frac{3}{5}$	$\frac{4}{5}$	0
0	$[-\frac{1}{5}-\frac{1}{3}]$	$[-\frac{1}{5}-\frac{1}{3}]$	$[-\frac{1}{5}-\frac{1}{3}]$	$[-\frac{1}{5}]$	$[-\frac{1}{5}]$



Second pass

- Pre-order, just like Walker
- We're done!



Resources

- Bill Mill's page .. `http://billmill.org/pymag-trees/`
- Reingold's paper . . . `http://reingold.co/tidier-drawings.pdf`
- Walker's paper . . . `http://www.cs.unc.edu/techreports/89-034.pdf`
- Buccheim's paper . . . `https://github.com/tristanpenman/n-puzzle/blob/master/doc/reference/Drawing%20Rooted%20Trees%20in%20Linear%20Time.pdf`