MIT OpenCourseWare
http://ocw.mit.edu

6.006 Introduction to Algorithms
Spring 2008

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.

6.006 Recitation

Build 2008.17

Quiz Review

- Problems
  - Interesting
  - Get you in the right mindset
  - We'll run out of time, not problems
- No Concepts - if you don't know them by now, ask in office hours

Problem 1: Poke

- A[1 ... n] sorted array of integers
  - can contain negative integers
  - no duplicates
- Want: i s.t. A[i] = i
  - if multiple possibilities, one will suffice
  - if no such i exists, say so

Poke: Intuition

- Play with the examples on the right
- Build the intuition
- Figure it out

1	2	3	4	5	6	7	8
-3	-1	1	3	**5**	7	9	11

1	2	3	4	5	6	7	8
2	4	6	8	10	12	14	16

1	2	3	4	5	6	7	8
-1	0	1	2	4	5	6	**8**

1	2	3	4	5	6	7	8
1	3	4	6	9	10	11	13

Poke: Solution

- **Key**: if $A[i] > i$, $A[j] > j$ for any $j > i$
  - proof: $A[i \ldots j]$ has $j - i + 1$ cells, must contain $j - i + 1$ values; integers only, no duplicates, so $A[j] \geq i + (j - i) + 1 > j$
- Solution: using key above, adapt binary search to find i
- Time: $O(\log(n))$

Problem 2: Knapsack

- $A[1 \ldots n]$ numbers (not necessary integers)
- s also number (not necessary integer)

  i) find i_1, i_2 s.t. $A[i_1]+A[i_2] = s$

  ii) find $i_1, i_2 \ldots i_k$ s.t. $A[i_1] + \ldots + A[i_k] = s$

- Hint: do better than $O(n^k)$

Palantir phone interview, 2007

Knapsack: Intuition

- Play with the examples on the right
- Build the intuition
- Figure it out

k = 2, S = 13

1	1	2	3	**5**	**8**	13	21

k = 2, S = 2

1	**1**	2	3	5	8	13	21

k = 4, S = 8

1	1	2	3	**5**	8	13	21

k = 4, S = 18

1	1	**2**	**3**	**5**	**8**	13	21

Knapsack: Solution 1

- build a dictionary d = {A[i] : i for i in 1...n} (maps numbers in A to their positions)
- for j in 1 ... n
  - if d contains s’ = s - A[j], then we have a solution
  - obtain the solution as i_1, i_2 = j, d[s’]

Knapsack: Solution II

- let $h = \lfloor k / 2 \rfloor$ (h stands for half)
- build a dictionary $d = \{A[i_1] + \ldots + A[i_h] : [i_1, i_2 .. i_h]$ for $i_1 \ldots i_h$ in $1 \ldots n\}$ (maps sums of h-tuples in a to the positions of the numbers)
- for $j_1, j_2 \ldots j_{k-h}$ in $1 \ldots n$
  - if d contains $s' = s - (A[j_1] + \ldots + A[j_k])$, solution $i_1, i_2 \ldots i_k = [j_1, j_2, \ldots j_{k-h}] + d[s']$

Problem 3: Segments

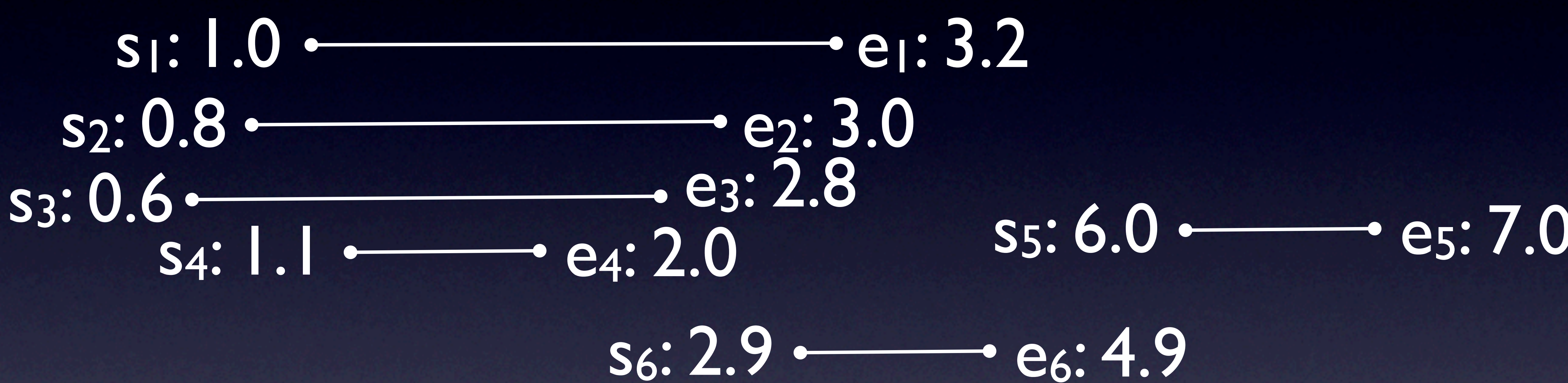
- Given n 1-D line segments $[s_i, e_i]$
  - all endpoints s_i, e_i distinct real numbers
- Want:

i) detect if there are any intersections

ii) count the number of intersections

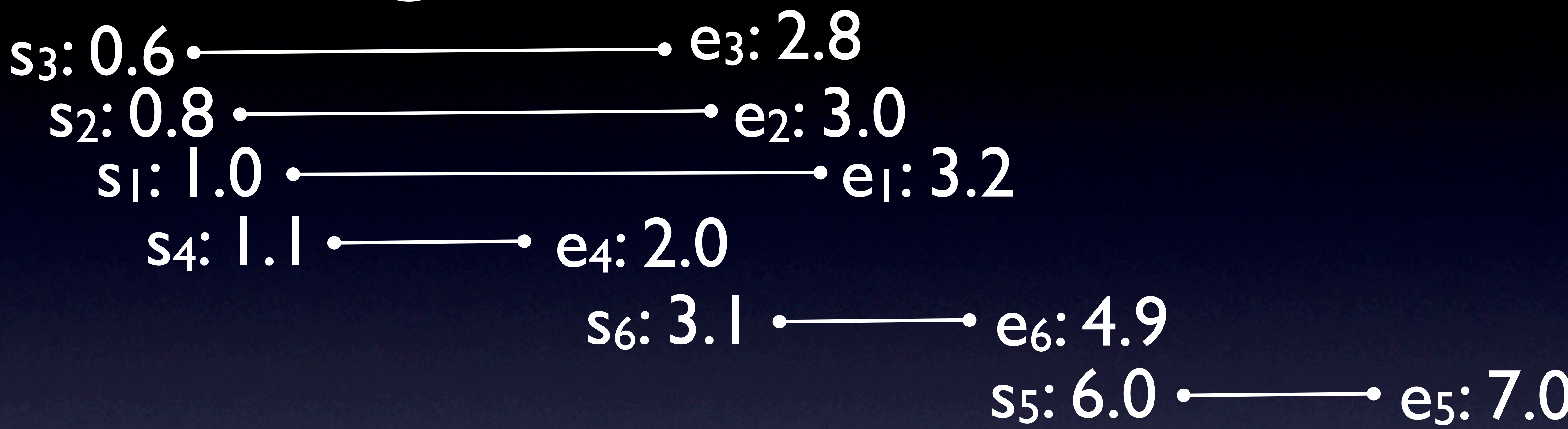
iii) report the intersecting segments

Segments: Intuition



- Yes: 7 intersections
- (1, 2) (1, 3) (1, 4) (2, 3) (2, 4) (3, 4) (1, 6)

Segments: Solution 1



- Sort segments by starting point:
  (0.6, 2.8) (0.8, 3.0) (1.0, 3.2)
  (1.1, 2.0) (2.9, 4.9) (6.0, 7.0)
- Only check adjacent segments ($s_{i+1} < e_i$)

Segments: Solution II

- For each segment, generate two events
  - start, end
- Process events sorted by their coordinate
  - start of segment:
    inters += open_segs
    open_segs += 1
  - end of segment:
    open_segs -= 1

0.6	s_3	**0** → 1	+0
0.8	s_2	**1** → 2	+1
1.0	s_1	**2** → 3	+2
1.1	s_4	**3** → 4	+3
2.0	e_4	4 → 3	
2.8	e_3	3 → 2	
3.0	e_2	3 → 2	
3.1	s_6	2 → 1	
3.2	e_1	**1** → 2	+1
4.9	e_6	1 → 0	
6.0	s_5	0 → 1	+0
7.0	e_5	1 → 0	

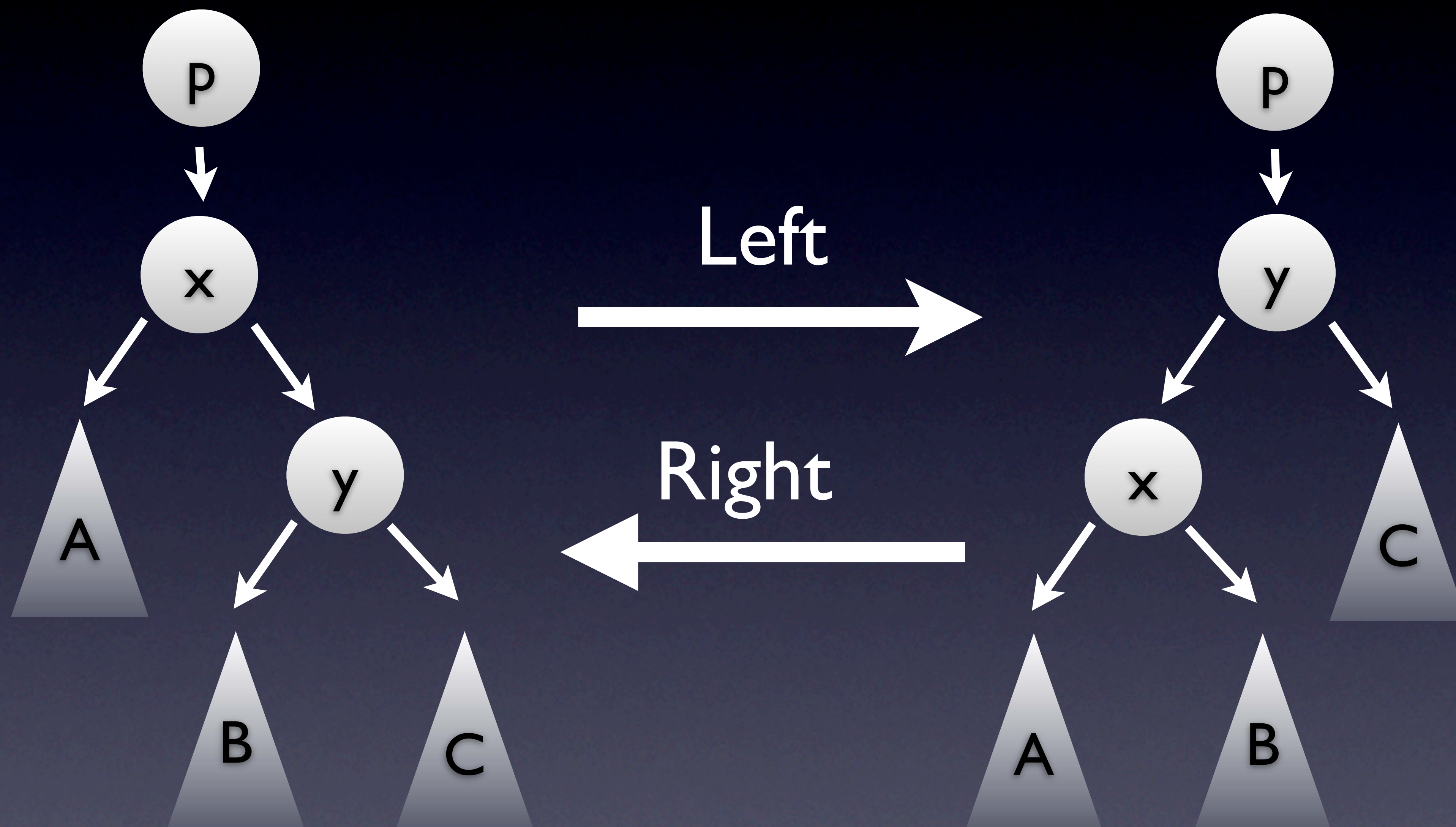
Segments: Solution III

0.6	s_3	**0** → 1	+0	s_3	{3}	
0.8	s_2	**1** → 2	+1	s_2	{**2**, 3}	(2, 3)
1.0	s_1	**2** → 3	+2	s_1	{**1**, 2, 3}	(1, 2) (1, 3)
1.1	s_4	**3** → 4	+3	s_4	{1, 2, 3, **4**}	(4, 1) (4, 2) (4, 3)
2.0	e_4	4 → 3		e_4	{1, 2, 3}	
2.8	e_3	3 → 2		e_3	{1, 2}	
3.0	e_2	3 → 2		e_2	{1}	
3.1	s_6	2 → 1	+1	s_6	{1, **6**}	(6, 1)
3.2	e_1	**1** → 2		e_1	{6}	
4.9	e_6	1 → 0		e_6	{ }	
6.0	s_5	0 → 1	+0	s_5	{**5**}	
7.0	e_5	1 → 0		e_5	{ }	

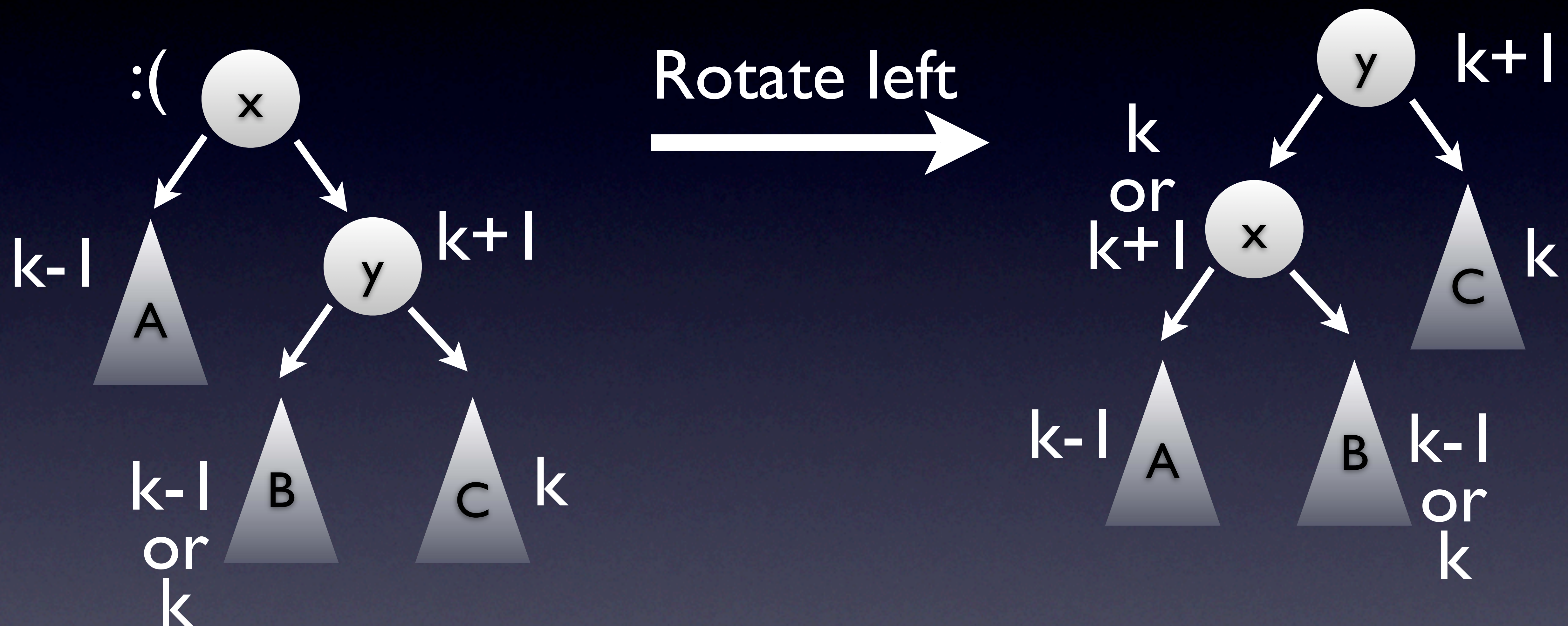
Bonus: AVL review

- Rotations
- Using rotations to rebalance

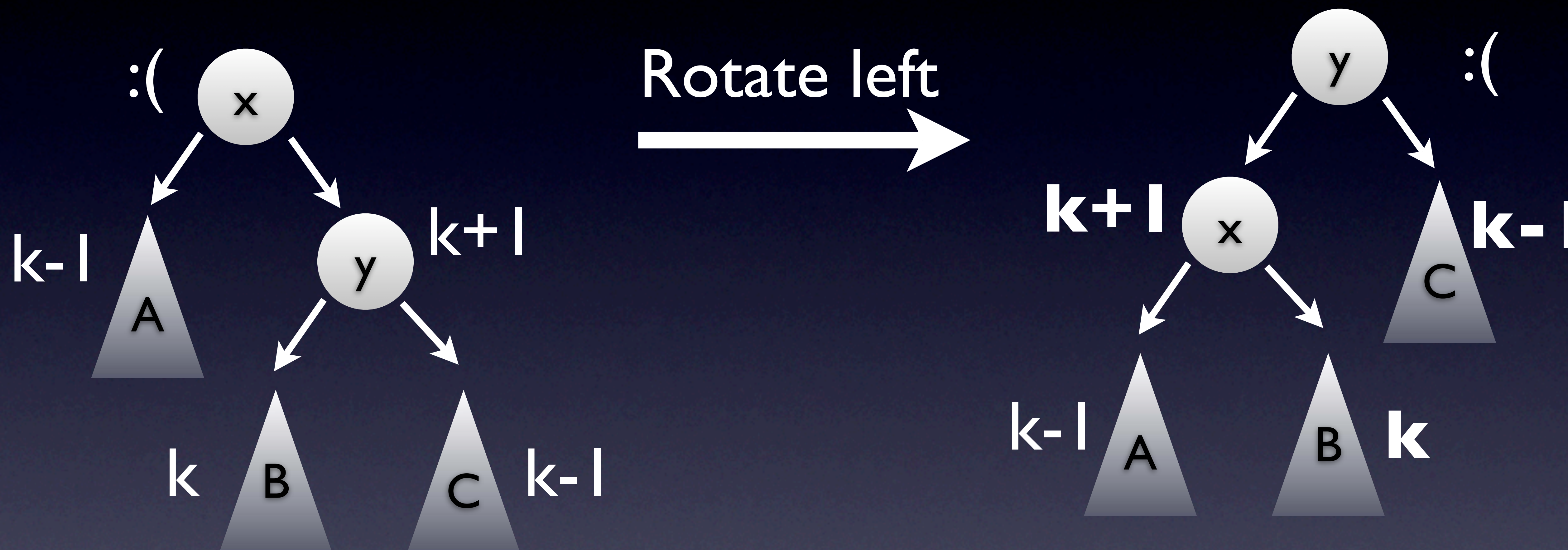
Uberpoke (rotations)



Rebalancing: Easy

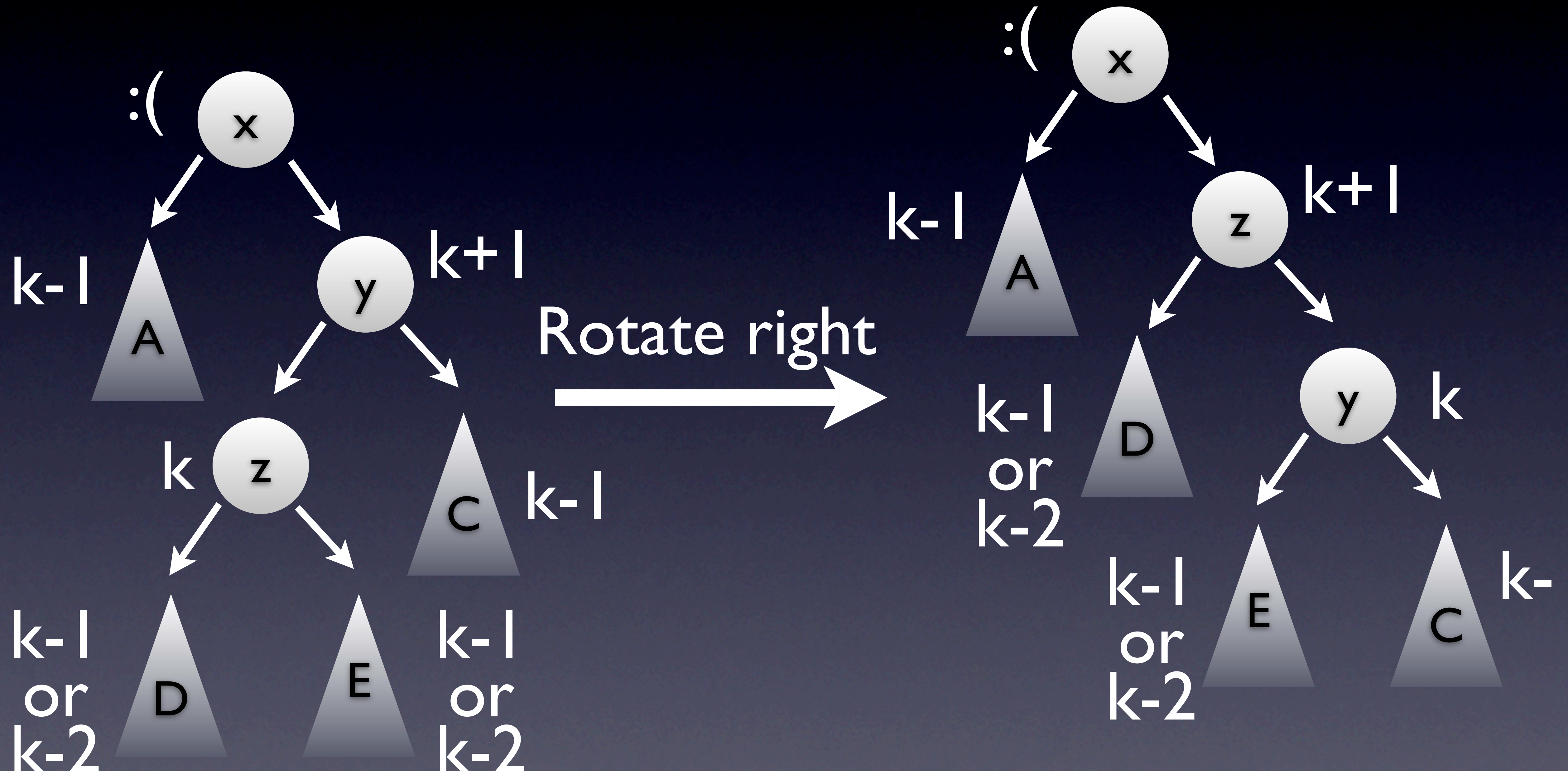


Rebalancing: Hard

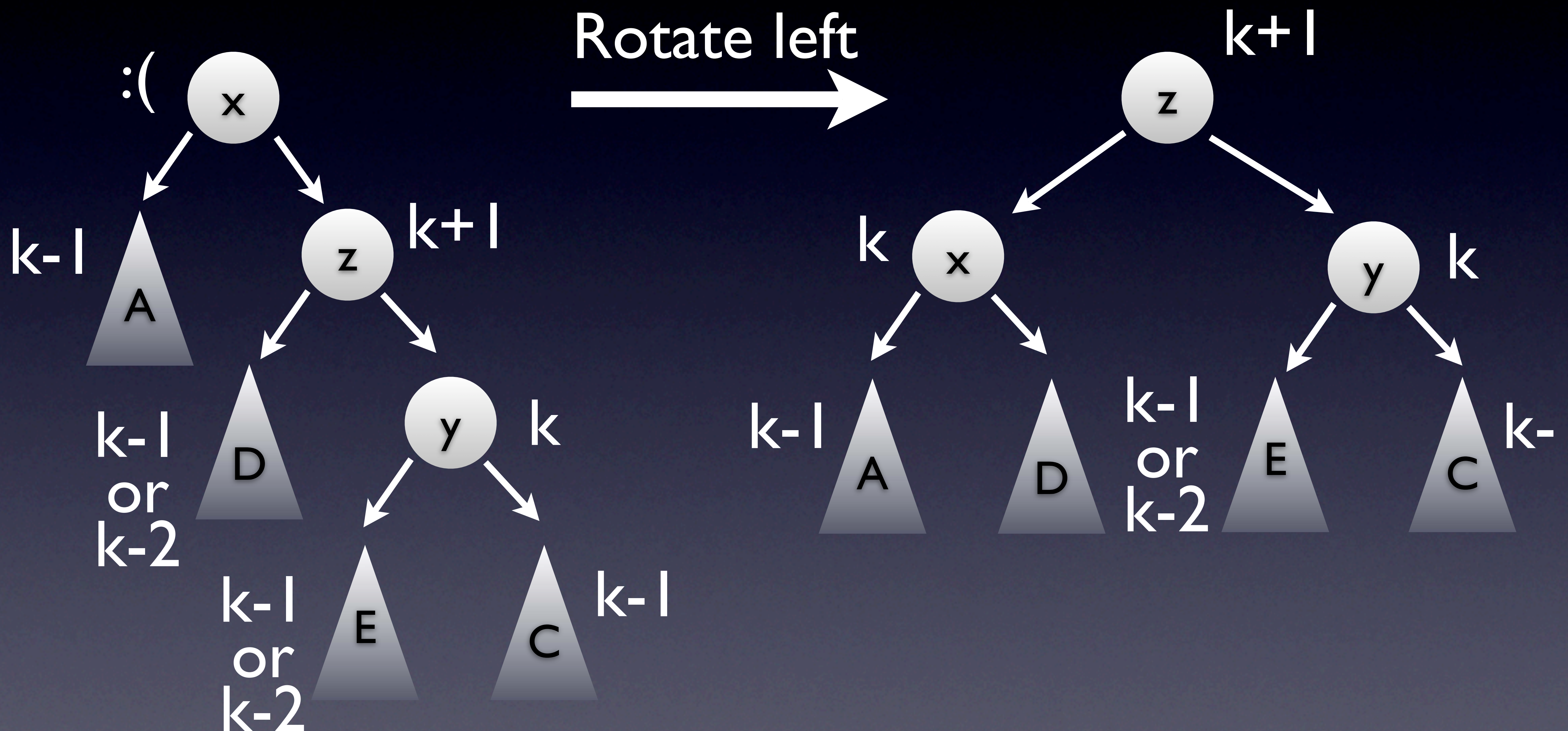


B cannot be taller than C

Hard Rebalancing: Right, then Left



Hard Rebalancing: Right, then Left



Problem 4: BSTing it Up

- BST with n nodes
- Obtaining keys in sorted order:
  - Call **minimum**: node w/ minimum key
  - Call **successor** n-1 times: obtain the other n-1 keys, in sorted order
- Prove that method above takes O(n) time

BSTing it Up: Solution

- Amortized analysis: aggregate analysis
  - key observation: each edge visited twice
    - parent → child, then child → parent
  - n - 1 edges, O(n) edge and node visits

Problem 5: Bunker Hill

- Satellite imagery: rectangular arrays of pixels, 4 levels of gray / pixel, no noise
- WxH image of hill, w x h image of bunker
- Want: locate all bunkers on the hill
  - no noise, perspective, lighting (no AI)
  - pixel-by-pixel comparison works

Bunker Hill: Intuition

The Bunker (3x3)

1	**2**	**1**
2	**1**	**2**
1	**2**	**1**

The Hill (10x10)

1	**2**	**1**	2	3	0	0	2	1	3
2	**1**	**2**	1	0	0	**1**	**2**	**1**	2
1	**2**	**1**	2	1	3	**2**	**1**	**2**	3
3	3	3	1	2	1	**1**	**2**	**1**	1
0	0	0	2	1	2	0	1	0	3
0	3	1	3	2	1	3	2	1	0
0	**1**	**2**	**1**	3	3	**2**	**1**	**2**	3
0	**2**	**1**	**2**	1	0	**1**	**2**	**1**	2
3	**1**	**2**	**1**	2	3	**2**	**1**	**2**	1
1	3	2	1	1	0	3	0	1	0

Bunker Hill: Solution

- Known as 2-D Rabin Karp
- Reduce bunker to $H_B[1...h]$, where $H_B[i]$ is the hash of $B[1...w][i]$
- Create rolling hashes $R_H[1...H]$ (one for every row of pixels in the hill), roll in parallel so they hash a block of w pixels
- 1-D Rabin Karp for $H_B[1...h]$ in $R_H[1...H]$