Chapter 5, Binary search trees:

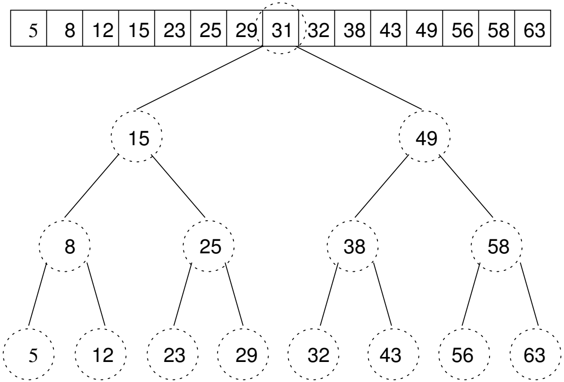
- Binary search trees; the balanced BST problem (spring-break eve)
- AVL trees (last week Monday and Wednesday)
- Traditional red-black trees (last week Friday, finished Monday)
- Left-leaning red-black trees (Monday, finish today)
- "Wrap-up" BSTs, B-trees (**Today**)
- Begin dynamic programming (Friday)
- Test 2 Wednesday, Apr 5

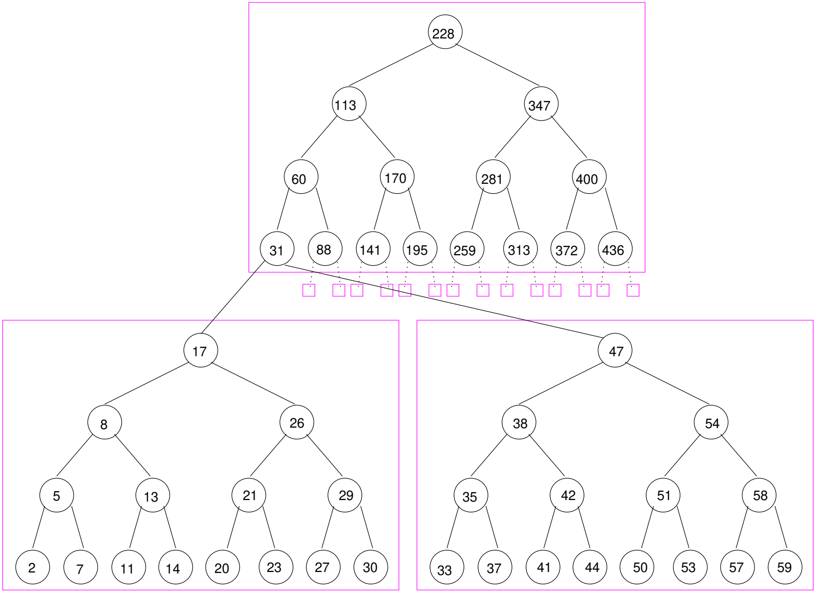
Today:

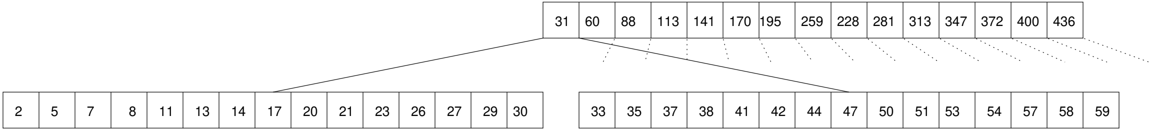
- Look ahead to Test 2
- Balanced tree comparisons
- Survey of B-trees

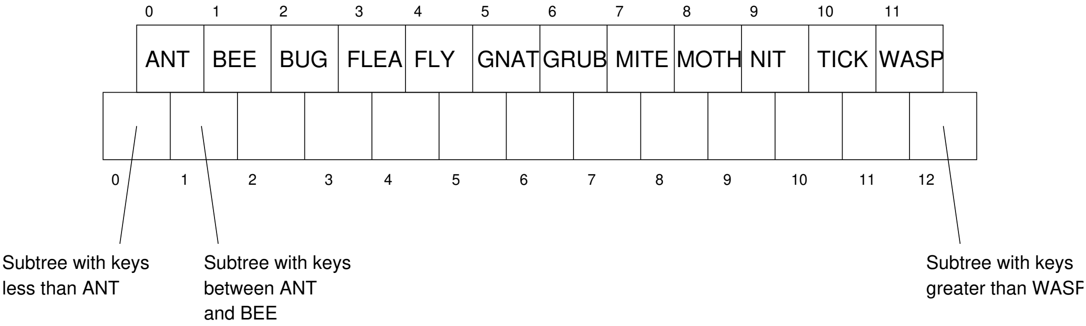
AVL trees	(Traditional) red-black trees
$h \leq 1.44 \lg n$	$h \leq 2 \lg(n+2) - 2$
The difference between the longest routes to leaves in the two subtrees is no greater than 1.	The longest route to any leaf is no greater than twice the shortest route to any leaf.
Stronger constraint, more aggressive re-balancing, more balanced tree, more work spent rebalancing.	Looser constraint, less aggressive rebal-ancing, less balanced tree, less work spent rebalancing.

	After puts			After removals		
	Height	Leaf %	Total depth	Height	Leaf %	Total depth
Unbalanced	32	33.3%	134507	28	16.8%	61207
	31	33.2%	127865	26	17.0%	58171
	30	33.1%	129037	26	16.9%	58610
	28	33.5%	124463	26	17.3%	56086
	32	33.4%	136730	28	16.9%	62092
AVL	16	43.2%	100327	14	21.5%	46088
	15	42.9%	100395	14	21.1%	46028
	15	42.8%	100341	14	21.1%	46028
	15	42.8%	100282	14	21.3%	45973
	15	43.0%	100582	14	21.2%	46097
Traditional RB	16	42.8%	101948	16	21.5%	46729
	16	42.9%	101226	15	21.4%	46344
	16	43.1%	101525	15	21.5%	46462
	16	42.7%	101680	16	21.5%	46572
	16	42.9%	101292	15	21.4%	46338
Left-leaning RB	18	42.8%	102288	18	21.6%	46950
	19	42.9%	102860	16	21.3%	46774
	18	43.1%	101949	17	21.5%	46691
	18	42.7%	102011	17	21.6%	46938
	19	42.9%	102552	16	21.4%	46764


5 8 12 15 23 25 29 31 32 38 43 49 56 58 63
15
49
8
25
38
58
5
12
23
29
32
43
56
63

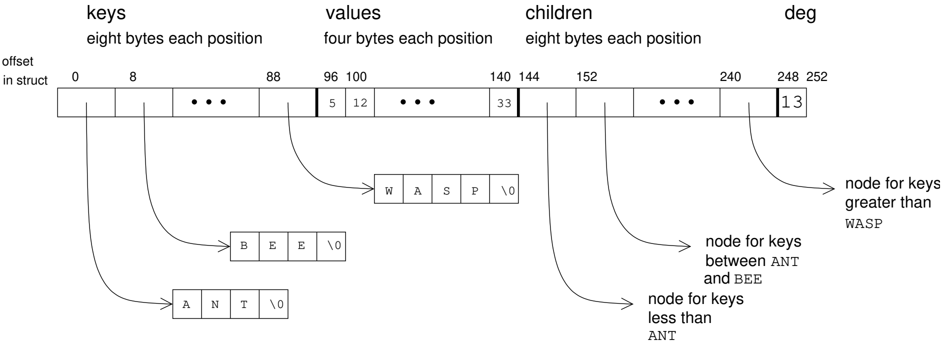

228
113
347
60
170
281
400
31
88
141
195
259
313
372
436
17
8
26
5
13
21
29
2
7
11
14
20
23
27
30
47
38
54
35
42
51
58
33
37
41
44
50
53
57
59

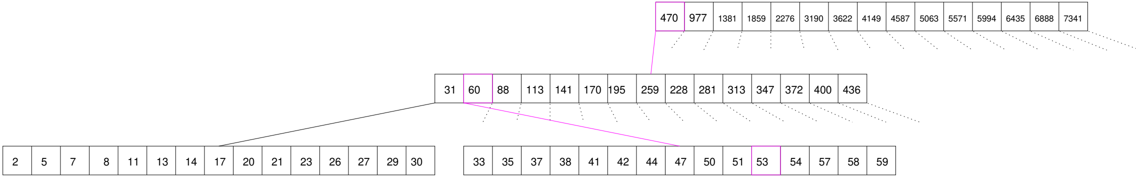

31 60 88 113 141 170 195 259 228 281 313 347 372 400 436
2 5 7 8 11 13 14 17 20 21 23 26 27 29 30
33 35 37 38 41 42 44 47 50 51 53 54 57 58 59

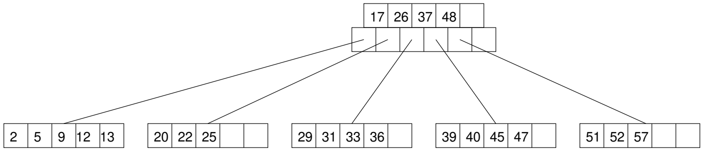

0 1 2 3 4 5 6 7 8 9 10 11
ANT BEE BUG FLEA FLY GNAT GRUB MITE MOTH NIT TICK WASP
0 1 2 3 4 5 6 7 8 9 10 11 12
Subtree with keys less than ANT
Subtree with keys between ANT and BEE
Subtree with keys greater than WASP

Formally, a B-tree with maximum degree M over some ordered key type is either

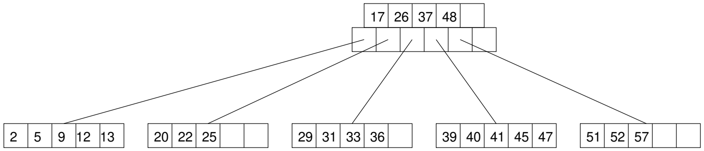
- empty, or
- a node with with $d - 1$ keys and d children, designated as lists `keys` and `children` such that
  - $\lceil M/2 \rceil \leq d \leq M$,
  - `children`[0] is a B-tree such that all of the keys in that tree are less than `keys`[0],
  - for all $i \in [1, d-1)$, `children`[i] is a B-tree such that all of the keys in that tree are greater than `keys`[$i-1$] and less than `keys`[i],
  - and `children`[$d-1$] is a B-tree such that all of the keys in that tree are greater than `keys`[$d-2$].


keys
eight bytes each position
values
four bytes each position
children
eight bytes each position
deg
offset
in struct
0
8
88
96
100
140
144
152
240
248
252
• • •
5
12
• • •
33
• • •
13
W A S P \0
B E E \0
A N T \0
node for keys
greater than
WASP
node for keys
between ANT
and BEE
node for keys
less than
ANT

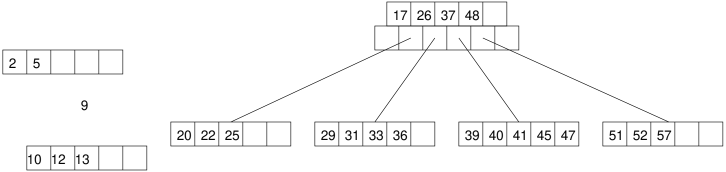

470 977 1381 1859 2276 3190 3622 4149 4587 5063 5571 5994 6435 6888 7341
31 60 88 113 141 170 195 259 228 281 313 347 372 400 436
2 5 7 8 11 13 14 17 20 21 23 26 27 29 30
33 35 37 38 41 42 44 47 50 51 53 54 57 58 59



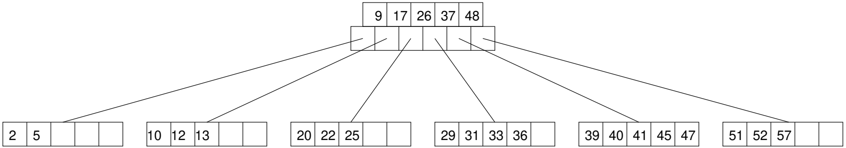
17 26 37 48
2 5 9 12 13
20 22 25
29 31 33 36
39 40 45 47
51 52 57

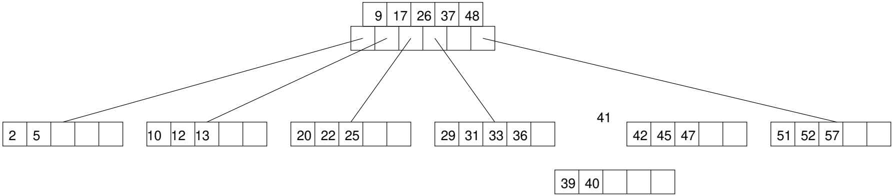


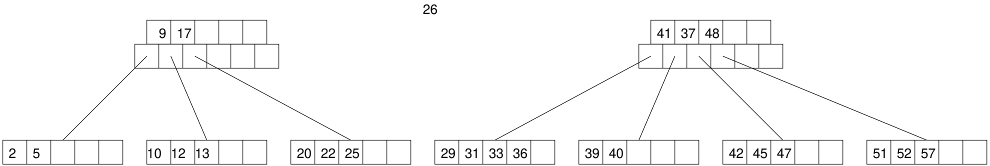
17 26 37 48
2 5 9 12 13
20 22 25
29 31 33 36
39 40 41 45 47
51 52 57

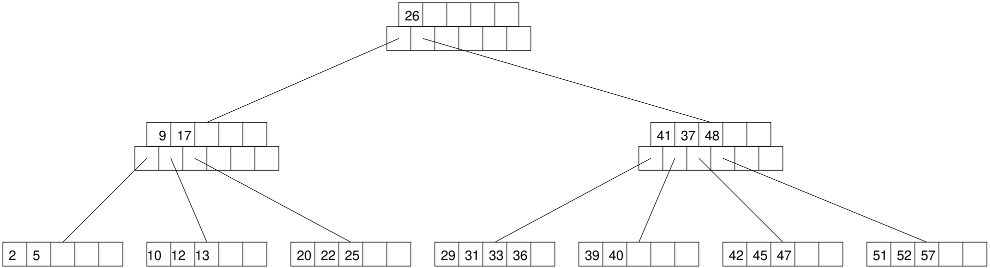


17 26 37 48
2 5
9
20 22 25
29 31 33 36
39 40 41 45 47
51 52 57
10 12 13


9 17 26 37 48
2 5
10 12 13
20 22 25
29 31 33 36
39 40 41 45 47
51 52 57


9 17 26 37 48
2 5
10 12 13
20 22 25
29 31 33 36
41
42 45 47
51 52 57
39 40


26
9 17
41 37 48
2 5
10 12 13
20 22 25
29 31 33 36
39 40
42 45 47
51 52 57


26
9 17
41 37 48
2 5
10 12 13
20 22 25
29 31 33 36
39 40
42 45 47
51 52 57

$$\underbrace{(M-1)}_{\text{keys per node}} \underbrace{\sum_{i=0}^{h-1} M^i}_{\text{sum of nodes at each level}} = (M-1)\frac{M^h-1}{M-1} = M^h - 1$$

$$\begin{aligned} n &= M^h - 1 \\ M^h &= n + 1 \\ h &= \log_M(n+1) \end{aligned}$$

$$
\begin{aligned}
n &= M^h - 1 \\
M^h &= n + 1 \\
h &= \log_M(n+1)
\end{aligned}
$$

$$
h = \log_{\frac{M}{2}}(n+1) = \frac{\log_M(n+1)}{1 - \log_M 2}
$$

Cost of a search:

$$
\begin{aligned}
\lg M \cdot h &= \lg M \cdot \frac{\log_M(n+1)}{1-\log_M 2} \\
&= \lg M \frac{\frac{\lg(n+1)}{\lg M}}{1-\frac{\lg 2}{\lg M}} \\
&= \frac{\lg(n+1)}{1-\frac{1}{\lg M}} \\
&= \frac{\lg M}{\lg M-1} \lg(n+1)
\end{aligned}
$$

Compare: $1.44 \lg n$ for AVL trees, $2 \lg n$ for RB trees.

Let c_0 be the cost of searching at a node (proportional to $\lg M$) and c_1 be the cost of reading a node from memory. The the cost of an entire search is

$$(c_0 + c_1)\frac{\log_M(n+1)}{1 - \log_M 2}$$

Now, consolidate the constants by letting $d = \frac{c_0+c_1}{1-\log_M 2}$, and we have

$$d \log_M(n+1)$$

Coming up:

Do **Traditional RB** *project (due Fri, Mar 22)*
(Recommended: Do **Left-leaning RB** *project for your own practice)*

Due **Wed, Mar 20 (today** *(end of day) (but hopefully you've spread it out)*
Read Sections 5.(4-6) [some parts carefully, some parts skim, some parts optional—see Canvas]
Do Exercise 5.13
Take quiz

Due **Mon, Mar 25** *(class time)*
Read Section 6.(1&2)
Do Exercises 6.(5–7)
Take quiz

Due **Tues, Mar 26** *(end of day)*
Read Section 6.3
Do Exercises 6.(16, 19, 23, 33)
Take quiz