B–Trees

[Bayer & McCreight, 1972]

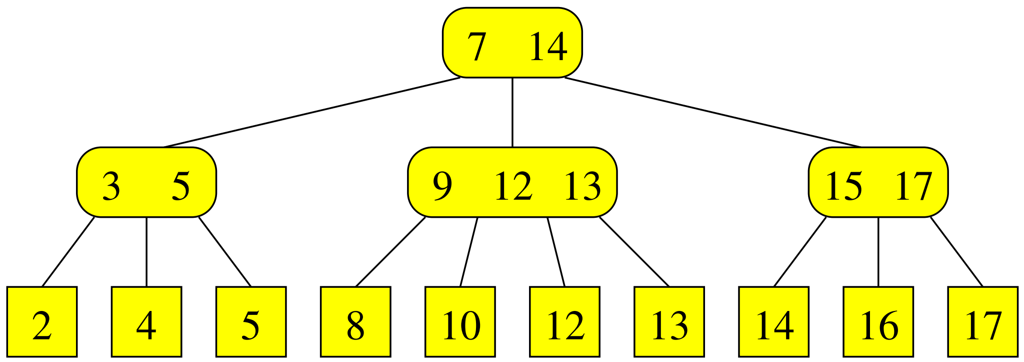
An Application of B–Trees

Core indexing data structure in many database management systems

TELSTRA, an Australian telecommunications company, maintains a customer database with 51.000.000.000 rows and 4.2 terabytes of data

(a, b)–Trees and B–trees

[Bayer & McCreight, 1972]

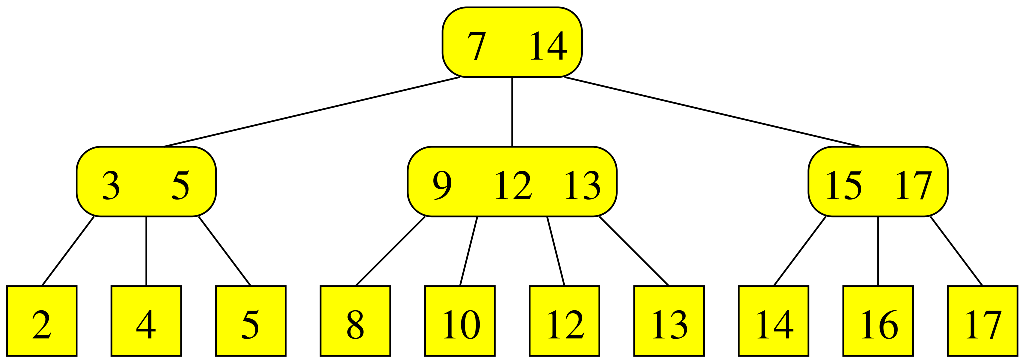


Definition A tree is an (a, b)–tree if $a \geq 2, \quad b \geq 2a - 1$ and

- All leaves have the same depth.
- All internal nodes have degree at most b.
- All internal nodes except the root have degree at least a.
- The root has degree at least two.

$(a, 2a - 1)$–trees are also denoted B–trees

Properties of (a, b)–Trees

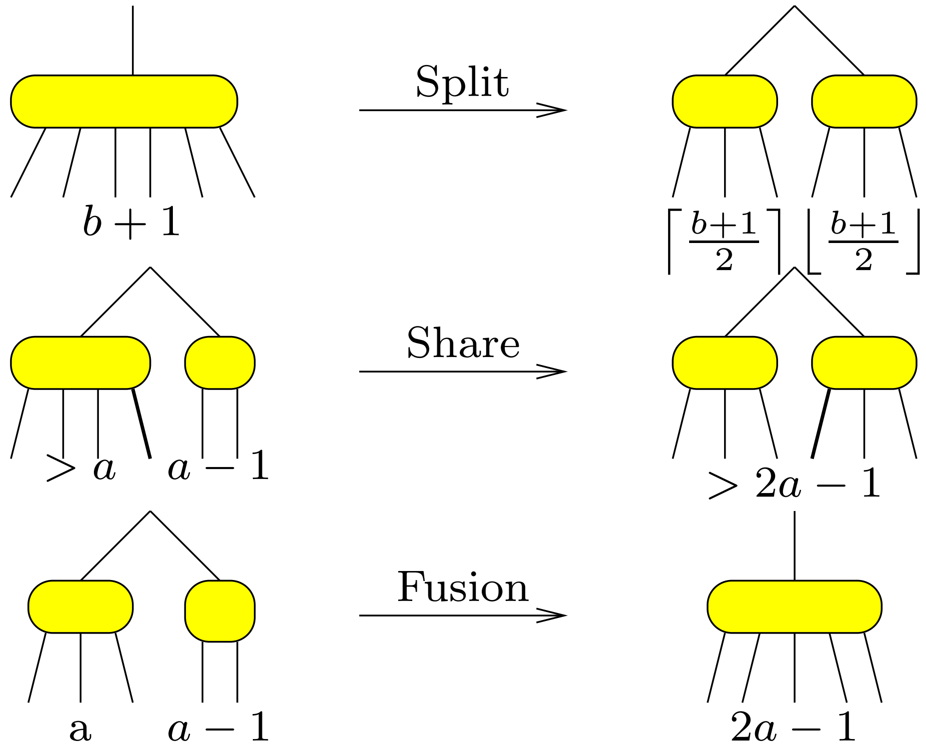


Lemma N leaves implies $\left\lceil \frac{\log n}{\log b} \right\rceil \leq \text{height} \leq \left\lfloor \frac{(\log n)-1}{\log a} \right\rfloor + 1$

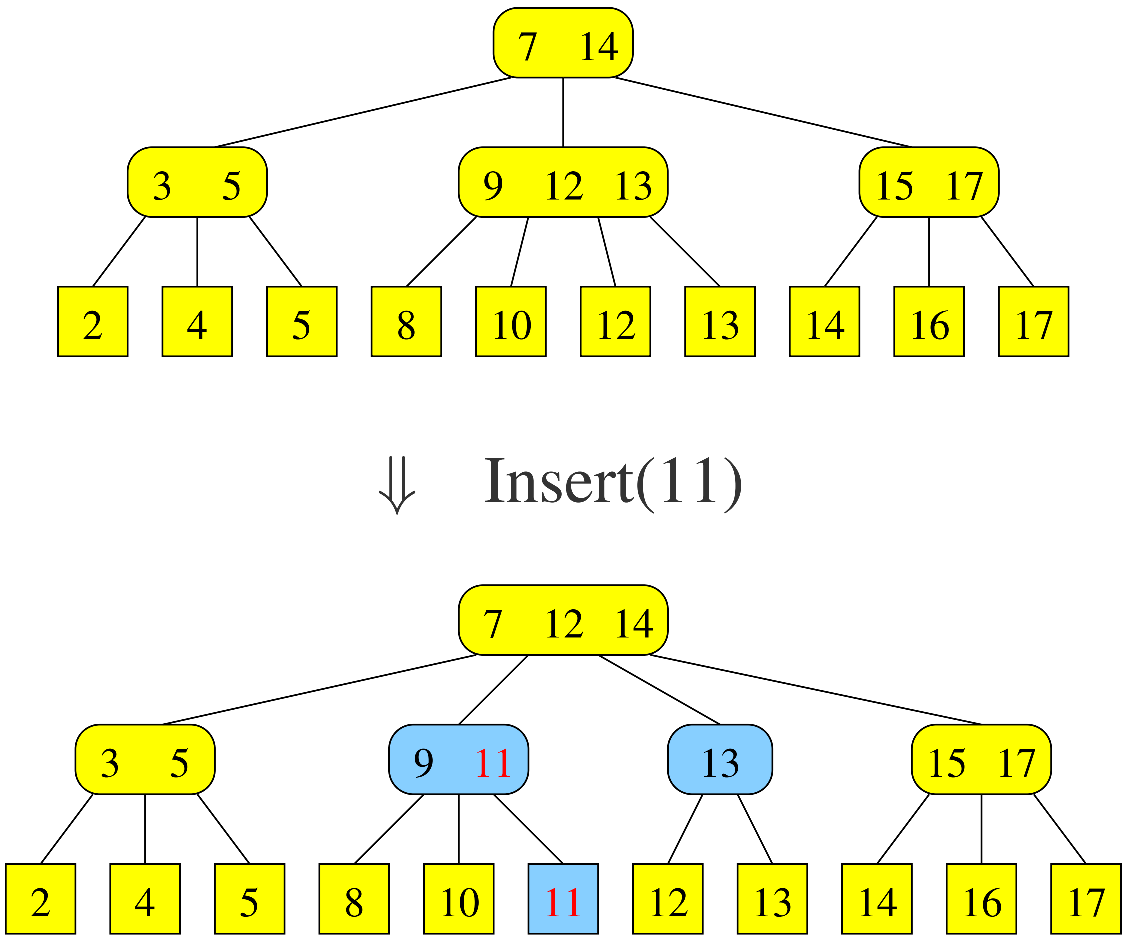
Lemma Searches require $O(\log_a n)$ I/Os if $b = O(B)$

Updates in (a, b)–Trees

- Search for location to insert or delete a leaf
- Create/delete leaf and search key at the parent node
- Rebalance using the following transformations



Example : Insert into a (2,4)–Tree



Analysis of (a, b)–Trees – Insertions Only

Theorem

$$n \text{ insertions imply } n/\left\lfloor (b+1)/2 \right\rfloor^h \text{ splits at height } h$$

$$\text{i.e. in total } O(n/b) \text{ splits}$$

Proof

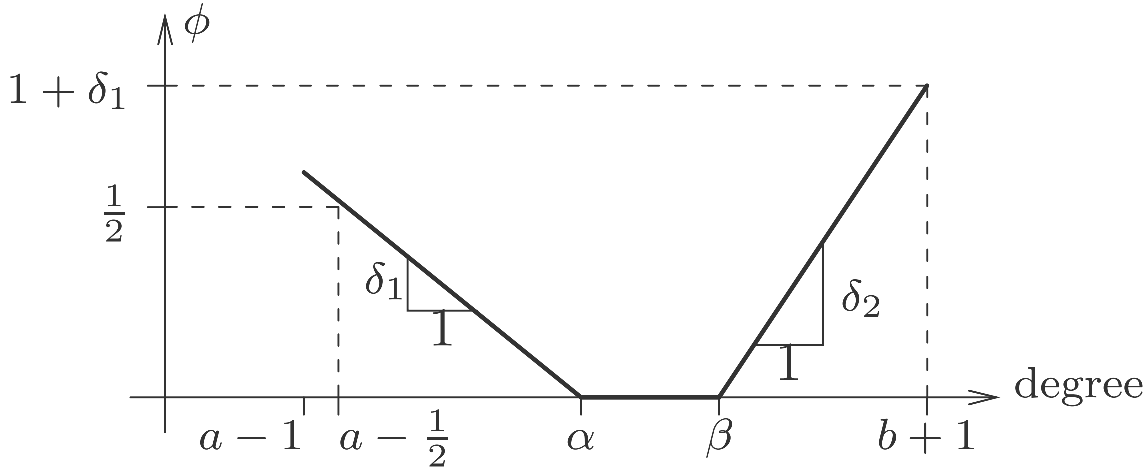
- Nodes are created due to splits
- All nodes except the root has degree at least $\left\lfloor (b+1)/2 \right\rfloor^h$
- The number of nodes in the lowest level dominates all other levels

$\square$

Analysis of (a, b)–Trees

Theorem If $b \geq 2a$, then i insertions and d deletions perform at most $O(\delta^h(i + d))$ splits and fusions at height h, where $\delta < 1$ depends on a and b

Proof (sketch) Amortization argument, each node has a potential ϕ (= measure of unbalancedness)



$\square$

Theorem If $b \geq 2a$, then the total # splits and # fusions is $O(i + d)$. If $b \geq (2 + \varepsilon)a$, for some $\varepsilon > 0$, the number of node splittings and node fusions is $O(\frac{1}{a}(i + d))$

Analysis of (a, b)–Trees

Theorem

$(B/3, B)$–trees perform $\Theta(1/B)$ rebalancing per update

Theorem

$(\lfloor B/2 \rfloor, B)$–trees perform $\Theta(1)$ rebalancing per update

Theorem

$(\lceil B/2 \rceil, B)$–trees perform $\Theta(\log_B N)$ rebalancing per update if B odd

Lower Bound for Searching

Theorem Searching for an element among N elements in external memory requires $\Omega(\log_B N)$ I/Os

Proof (sketch)

- Adversary argument
- Algorithm knows total order of stored elements
- Initially all elements are candidates for being the query element
- If prior to an I/O there are C candidate elements left, then there exists anwers leaving $\left\lceil \frac{C-B}{B+1} \right\rceil$ candidates after reading B elements

$\square$

Note The lower bound holds even if an I/O can read B arbitrary elements from memory