

CST8130: Data Structures

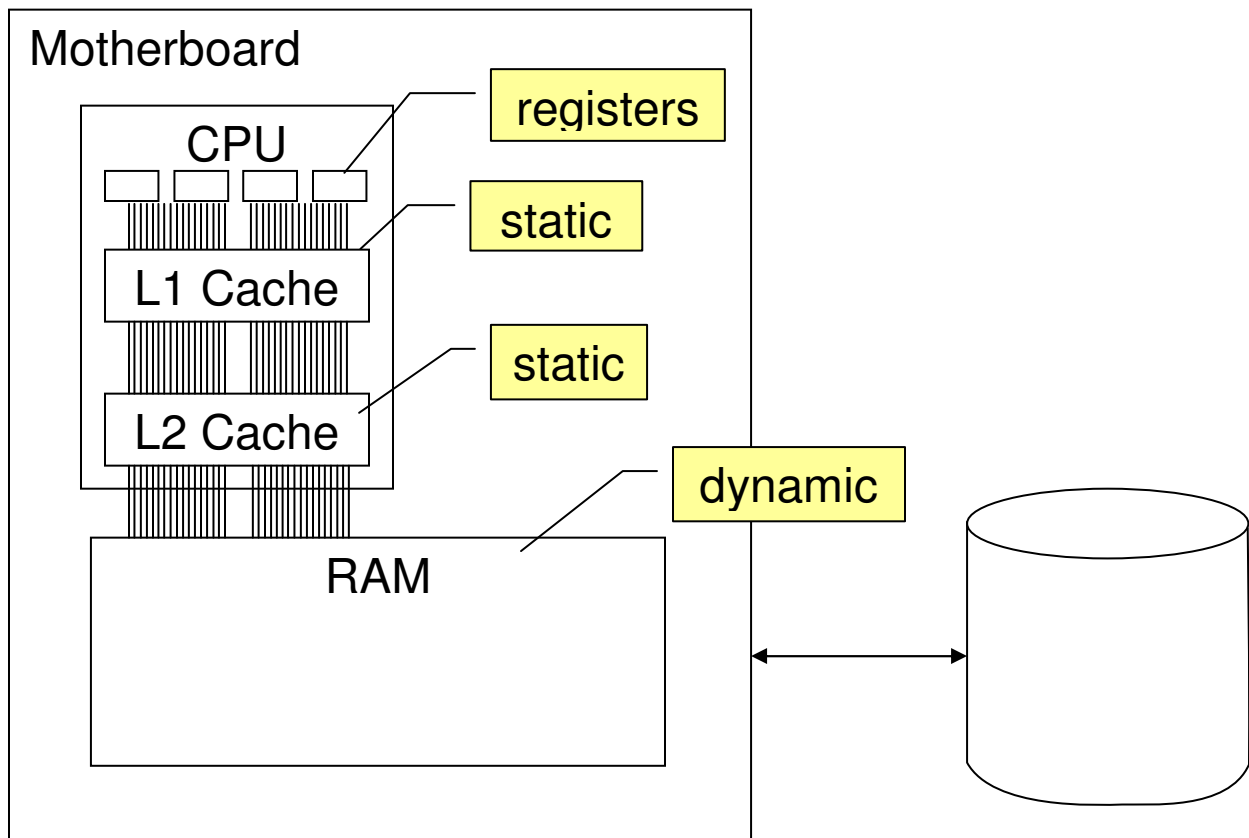
B-Trees

Binary Trees

- excellent efficiency: $O(\log_2 n)$
- assumes all access operations at the same speed

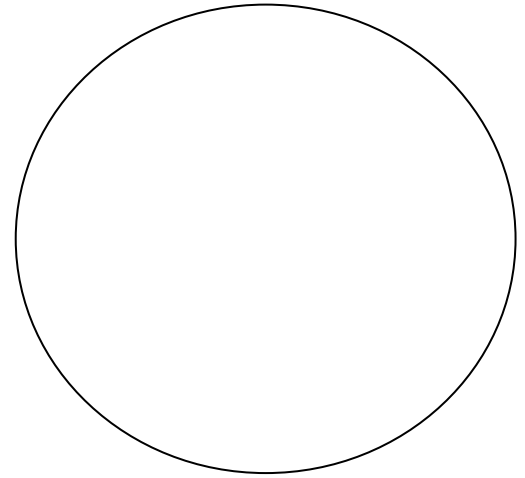
The Problem

- large databases may be primarily disk-based
- access times are dramatically different at different levels of hardware storage hierarchy
- too many access requests on slowest component



The Solution

- move larger chunks of data into RAM
- perform more searching in RAM, less on disk



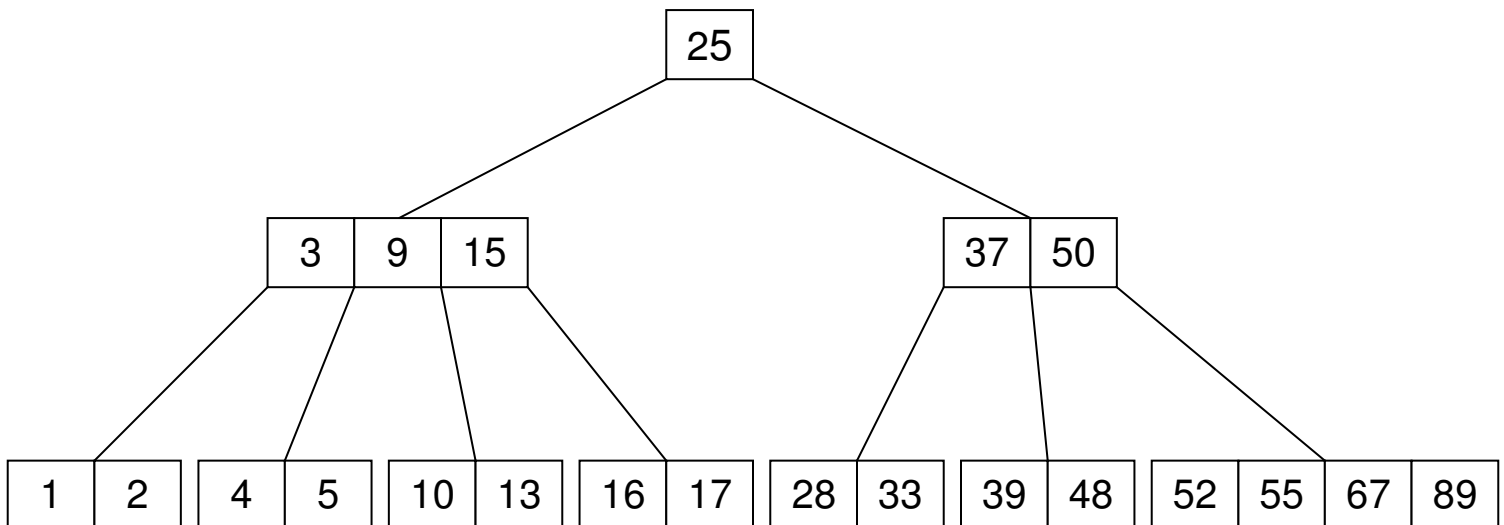
Optimize for Disk Organization

- sectors / clusters

B-Trees Rules

- root is either a leaf or has $2 \dots m$ subtrees.
- internal nodes (not root or leaf nodes) have at least $m/2$ subtrees and at most m non-null subtrees.
- all leaf nodes are at the same level (that is, the tree is perfectly balanced).
- leaf node has at least $m/2 - 1$ and at most $m - 1$ entries.

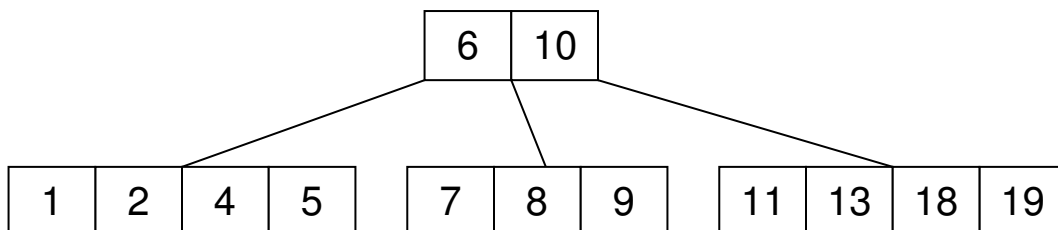
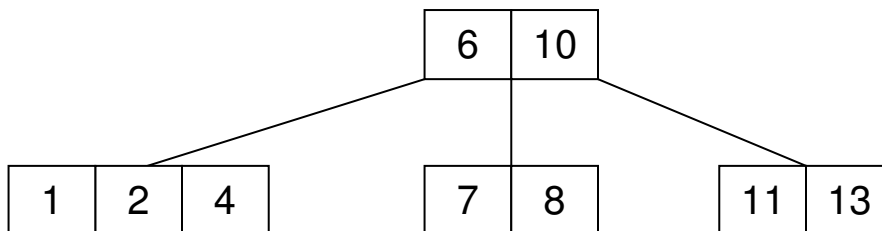
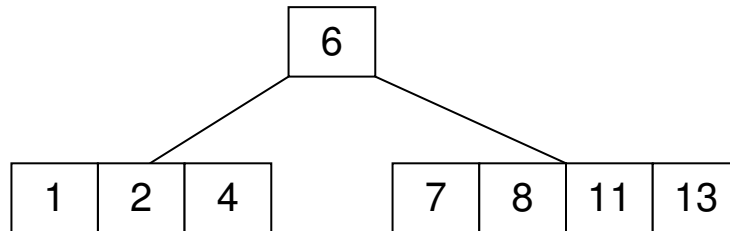
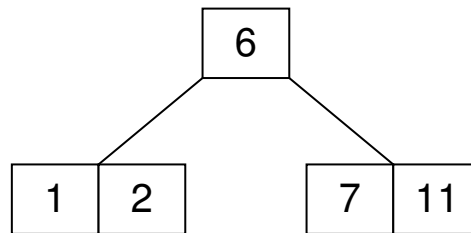
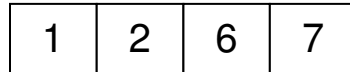
Sample B-Tree



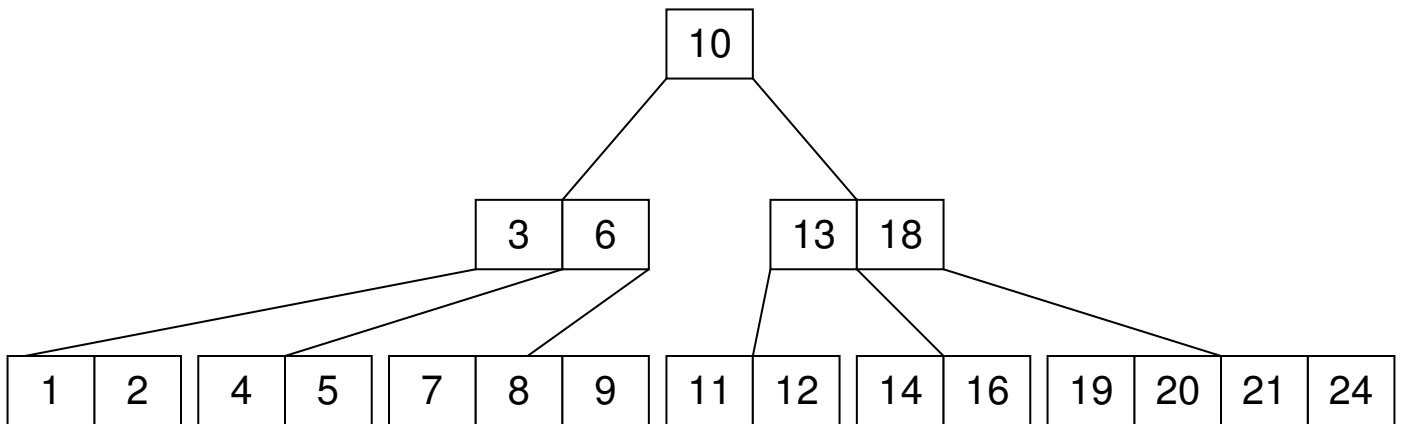
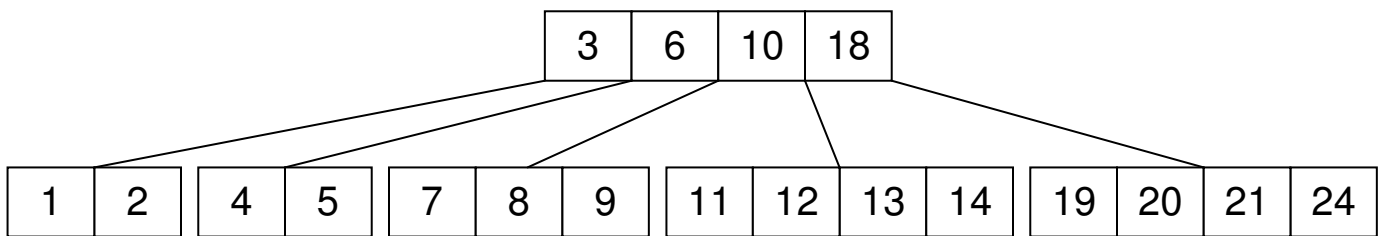
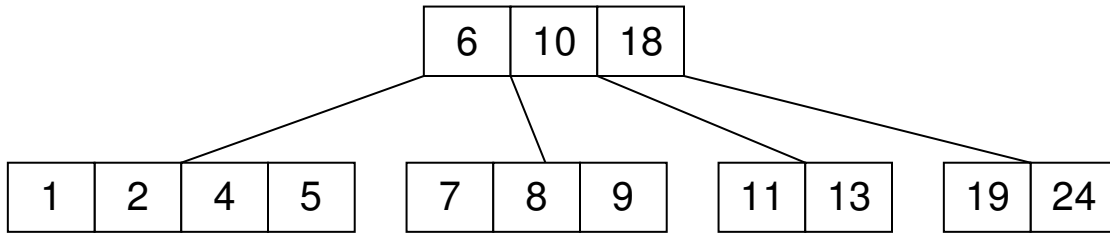
Building a B-Tree

Input Stream:

1	7	6	2	11	4	8	13	10	5	19	9	18	24	3	12	14	20	21	16
---	---	---	---	----	---	---	----	----	---	----	---	----	----	---	----	----	----	----	----



Input Stream: 1 7 6 2 11 4 8 13 10 5 19 9 18 24 3 12 14 20 21 16



B-Tree: Issues of Efficiency

- accessing data on disk
- height of tree represents number of disk accesses
- tree order (that is, number of paths out of each node) determines number of items stored in each node

Maximum Number of Items In B-Trees

<i>height</i>	<i>m = 2</i>	<i>m = 4</i>	<i>m = 10</i>	<i>m = 100</i>
1	1	3	9	99
2	3	15	99	9999
3	7	63	999	999999
4	15	255	9999	99999999

- best performance: values of m in range 50 to 400

2. Given the following sequence of numbers, show how they would be organized in a B-tree with order 3.

12, 24, 7, 4, 30, 13, 19, 27, 8, 10, 2, 18, 21, 14, 28, 1, 9

