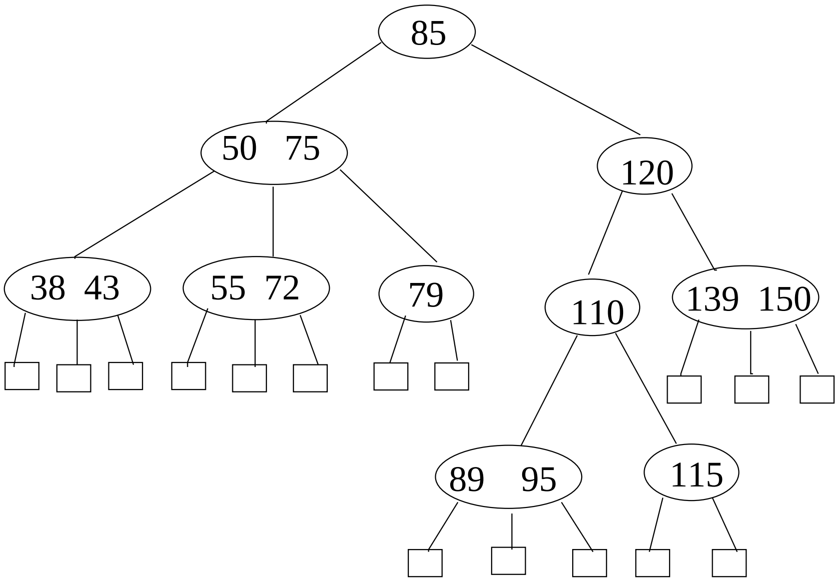
Multi-way search trees

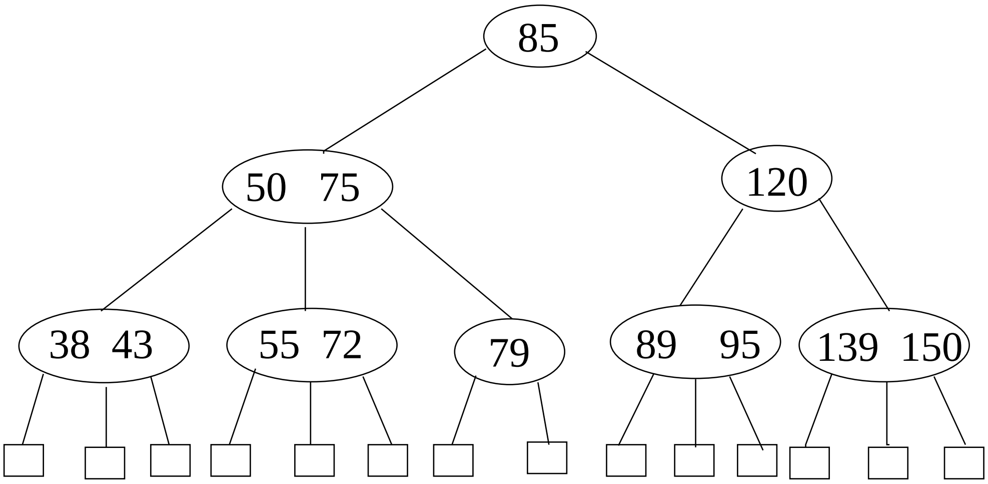
- A node is referred to as a m-node if it has m children
- A m-way search tree T is defined as an ordered tree having following properties.
  - every internal node n is a m-node $v_1, v_2, \ldots, v_m$ with $m \geq 2$
  - n stores m-1 items of type (k_i, x_i), for $1 \leq i \leq m-1$, where k is a key and x_i is an element.
  - n store two additional keys $k_0 = -\infty$ and $k_m = \infty$
  - the keys are ascending, i.e., $k_i \leq k_{i+1}$ for $1 \leq i \leq m-2$.
  - items (k, x) belonging to m-way tree rooted at child node v_i for $1 \leq i \leq m$ are such that such that $k_{i-1} \leq k \leq k_i$.

Example of an m-way tree



Balanced multi-way search trees

- Consider multi-way search trees with following additional constraints
  - m = 4 and all the external nodes having same depth.
- It means that each internal node must have between 2, 3, or 4 children and the tree is balanced. Following figure shows an example.



B-trees

- Balanced multi-way search tree with additional constraints are also known as B-trees.
- A B-tree of order m is a (a, b) tree with a=$\lceil m/2 \rceil$ and b=m.
- Usually B-trees are used for indexing data bases.
- The main problem is disk I/O.
- The value of m must be such that all m children references and $m-1$ keys at a node can fit into a single disk block.

Insertion into (2,4) tree

- B-trees insertion, deletion. and membership search follows the same pattern as discussed for (2,4) tree.
- Insertion can increase the height of the tree if all internal nodes at every level are fully populated.
- Insertion result in a split of some fully populated nodes.
- The split operation is as follows:
  - Proceed like binary search tree to locate the external node where new key should point.
  - Suppose the internal node where new key should be inserted, is already a 4 node, i.e., it has 3 keys.
  - Since no vacant position exist for insertion of the new key, split the old node n by two nodes n' and n''
  - node n' gets two keys k_1, k_2 and n'' gets the remaining key k_4.
  - key k_3 being propagated up to the parent node.