Lecture 13

Instructor: Karteek Sreenivasaiah

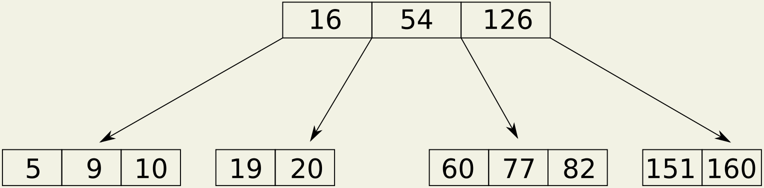
5th October 2018

BTree

A generalization of Binary Search Tree.

- Each node in a BTree can have $m \geq 1$ many elements.
- A node with m elements will have $m + 1$ children.
- Elements within a node are all sorted.
- Children's values follow the BST property.

Example



BTree

A BTree with *minimum degree* t:

- Number of elements in every node (except root) satisfies:

$$t - 1 \leq \# \text{ of elements} \leq 2t - 1$$

- Number of elements in root is at least one, and at most $2t - 1$.
- Elements in every node are sorted in non-decreasing order.
- A node with m elements has $m + 1$ children.
- All nodes satisfy the *generalized BST Property*.
- All leaves are at the same height.

BTree

A BTree with *minimum degree t*:

- Number of elements in every node (except root) satisfies:

$$t - 1 \leq \# \text{ of elements} \leq 2t - 1$$

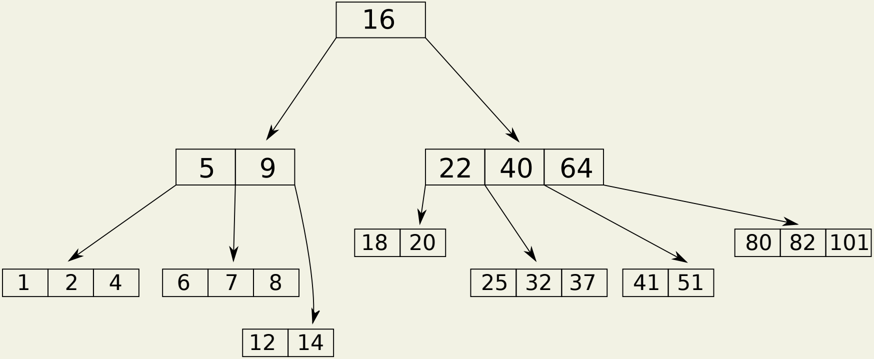
- Number of elements in root is at least one, and at most $2t - 1$.
- Elements in every node are sorted in non-decreasing order.
- A node with m elements has $m + 1$ children.
- All nodes satisfy the *generalized BST Property*
  - Generalized BST property:
    Let the elements in a node x be $x_1, x_2, \ldots x_\ell$.
    Let the children of x be $c_1, c_2, \ldots, c_{\ell+1}$.
    Then for every element m in the subtree under child c_i, we have:

$$x_{i-1} \leq m \leq x_i$$

- All leaves are at the same height.

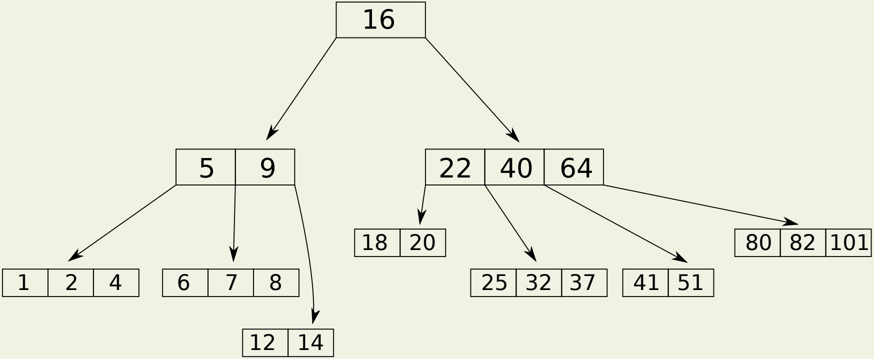
Example

A BTree with $t = 1$. Every node has between 1 and 3 elements.



Example

A BTree with $t = 1$. Every node has between 1 and 3 elements.



Implementation

Each node contains:

- n – Number of elements in the node.
- $n + 1$ many pointers to children.
- The elements $x_1, \ldots, x_n$ in non-decreasing order.

Height of a BTree

Theorem 1

A BTree with minimum degree t containing n elements has height $O(\log_t n)$

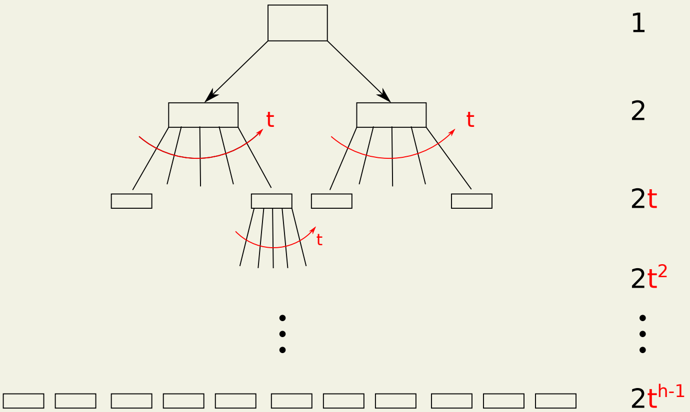
Proof

Take a BTree with min degree t of height h.

- Count minimum number of nodes in each level.
- Multiply with minimum number of elements in each node: $t - 1$.
- Conclude Theorem.

Height of a BTree

Counting number of nodes.



Height of a BTree

Total number of elements is thus:

$$
\begin{aligned}
n &= 1 + \sum_{i=1}^{h} 2t^{i-1}(t-1) \\
&= 1 + 2(t-1)\sum_{i=1}^{h} t^{i-1} \\
&= 1 + 2(t-1)\left(\frac{t^h-1}{t-1}\right) \\
&= 1 + 2(t^h - 1)
\end{aligned}
$$

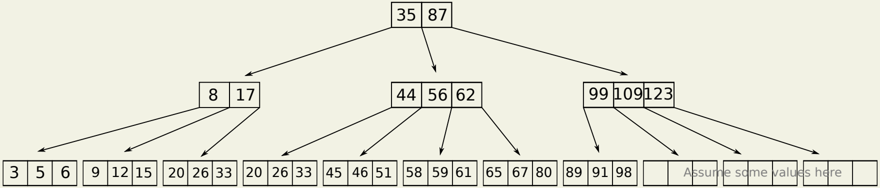
$$
\implies \frac{n+1}{2} = t^h
$$

$$
\implies h = \log_t \frac{n+1}{2}
$$

□

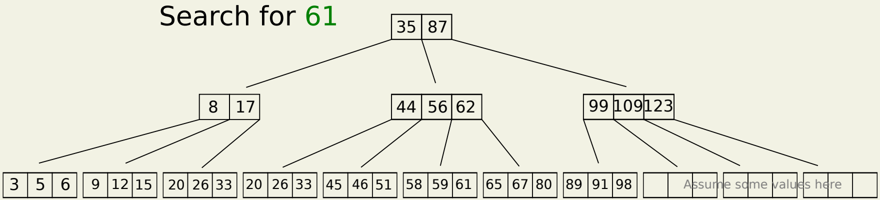
Search

BTree Search is a natural generalization of the procedure for BST.



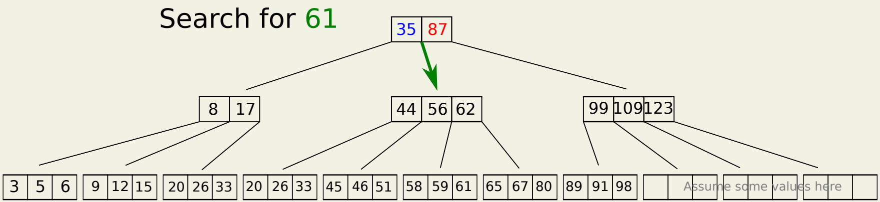
Search

BTree Search is a natural generalization of the procedure for BST.



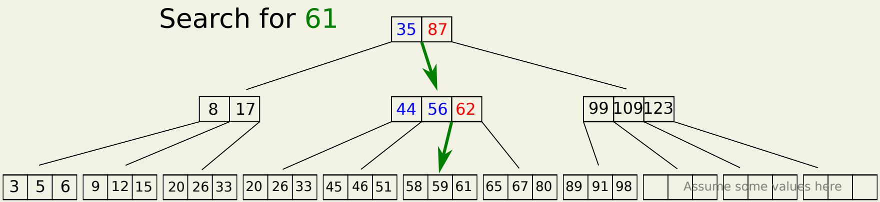
Search

BTree Search is a natural generalization of the procedure for BST.



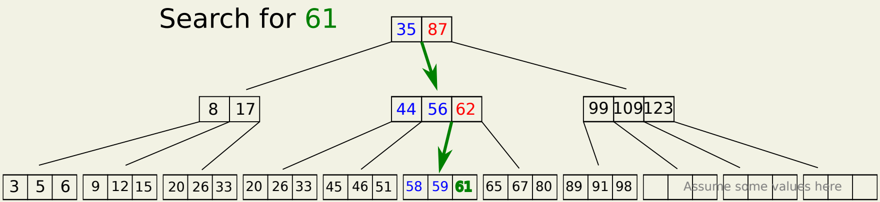
Search

BTree Search is a natural generalization of the procedure for BST.



Search

BTree Search is a natural generalization of the procedure for BST.



Insert procedure

BTree Insert

Intuitively, the insert procedure is very straightforward:

- Find the correct leaf to insert the new element.
- Try to insert the new element in to the leaf.

Insert procedure

BTree Insert

Intuitively, the insert procedure is very straightforward:

- Find the correct leaf to insert the new element.
- Try to insert the new element in to the leaf.

Note that unlike BST Insert, we are not creating a new node already! We try to fit the new element into an existing leaf.

Insert procedure – intuition

Min degree $t = 3$.

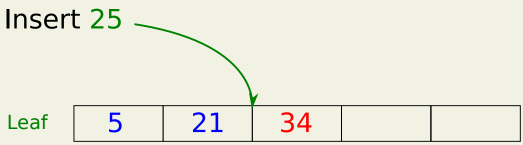
Insert 25

Leaf

5	21	34		

Insert procedure – intuition

Min degree $t = 3$.



Insert procedure – intuition

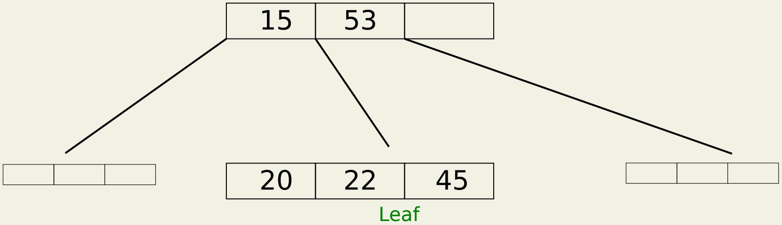
Min degree $t = 3$.

Leaf	5	21	25	34	

Insert procedure – intuition

Min degree $t = 2$

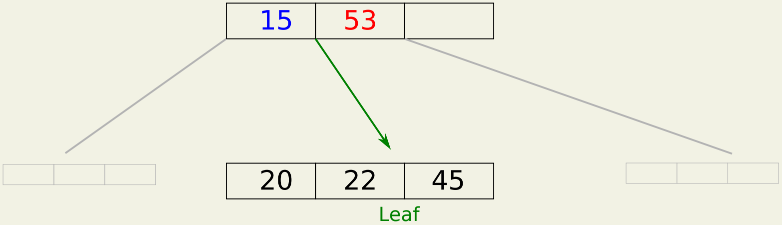
Insert 30



Insert procedure – intuition

Min degree $t = 2$

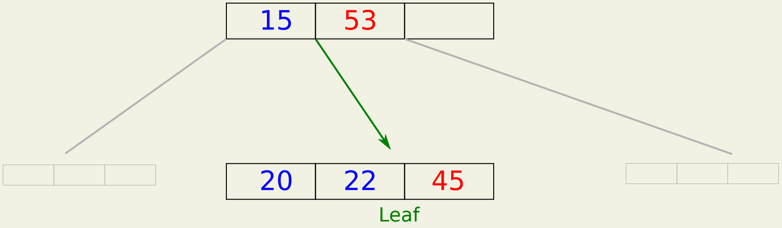
Insert 30



Insert procedure – intuition

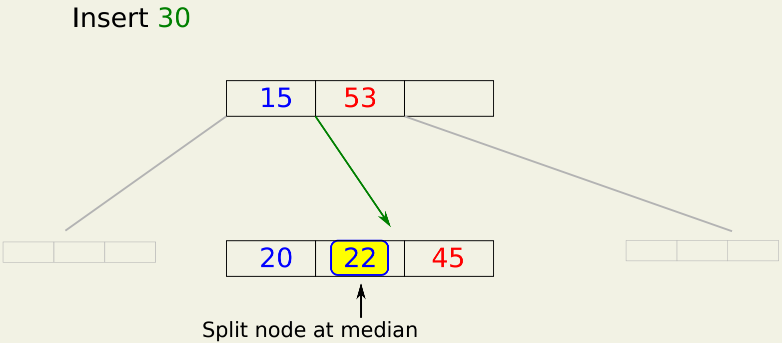
Min degree $t = 2$

Insert 30



Insert procedure – intuition

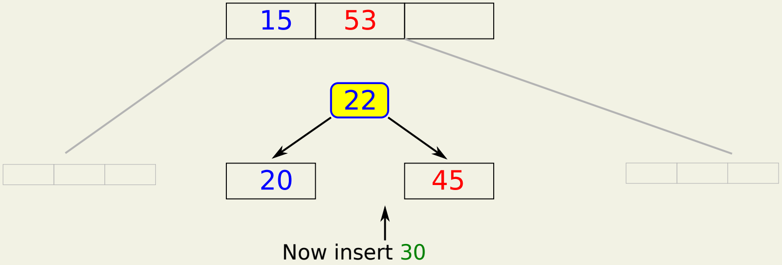
Min degree $t = 2$



Insert procedure – intuition

Min degree $t = 2$

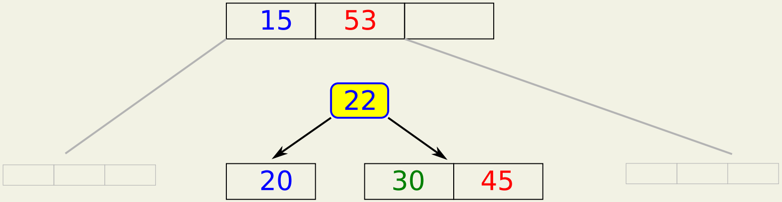
Insert 30



Insert procedure – intuition

Min degree $t = 2$

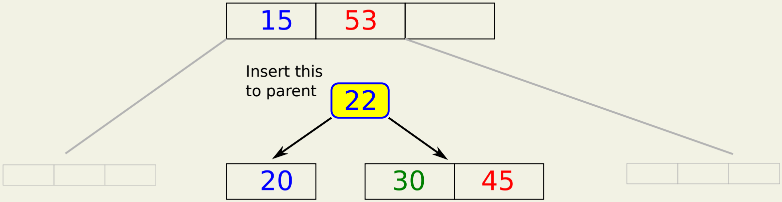
Insert 30



Insert procedure – intuition

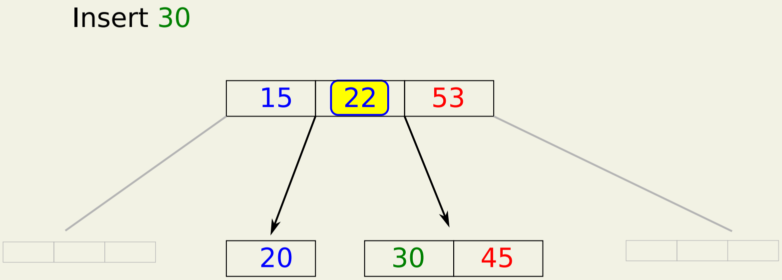
Min degree $t = 2$

Insert 30



Insert procedure – intuition

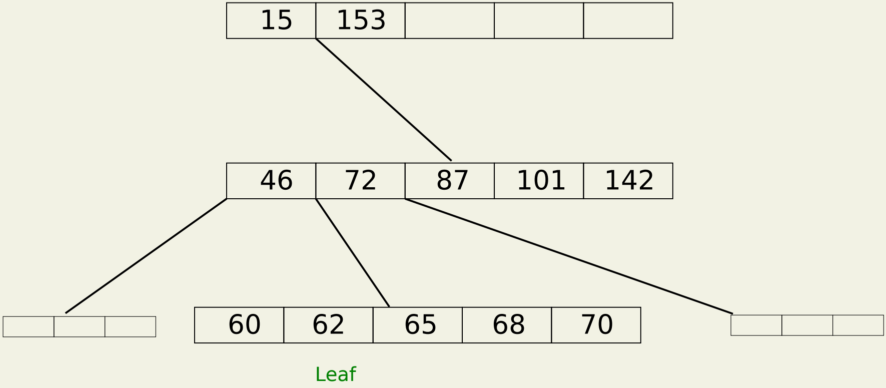
Min degree $t = 2$



Insert procedure – intuition

Min degree $t = 3$

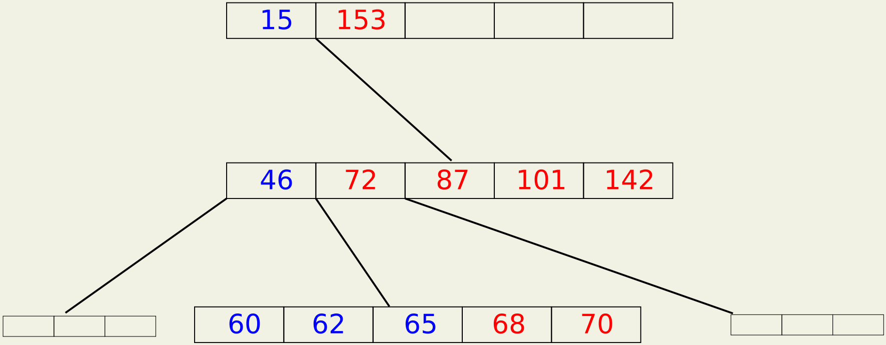
Insert 66



Insert procedure – intuition

Min degree $t = 3$

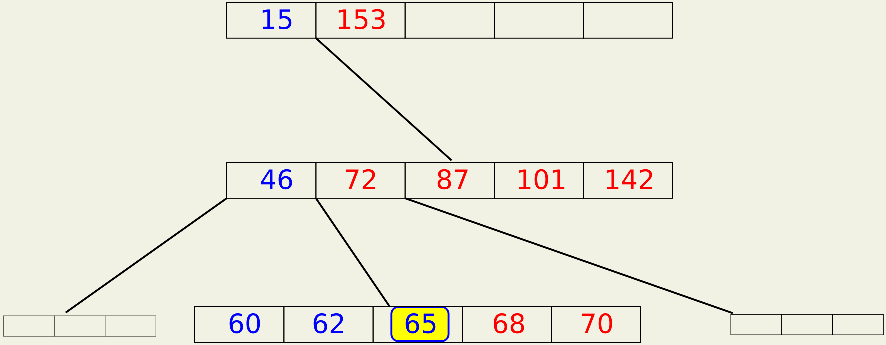
Insert 66



Insert procedure – intuition

Min degree $t = 3$

Insert 66

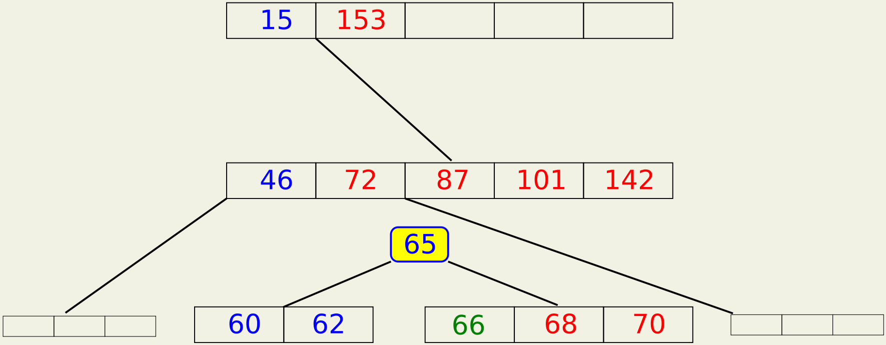


Split node at median

Insert procedure – intuition

Min degree $t = 3$

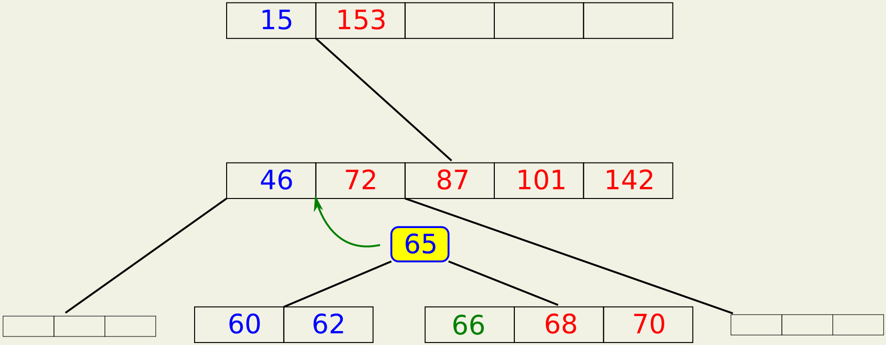
Insert 66



Insert procedure – intuition

Min degree $t = 3$

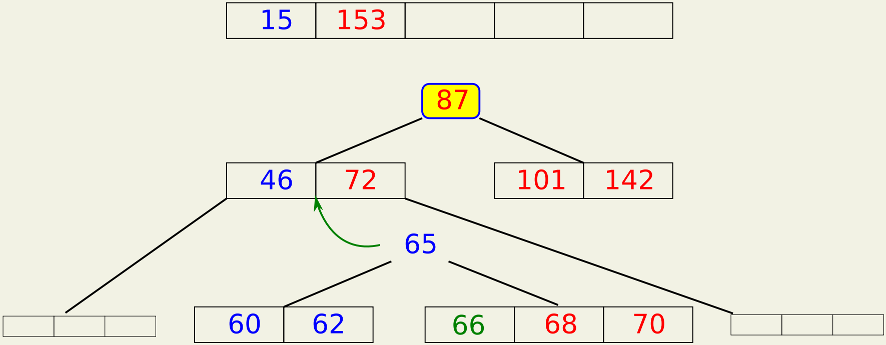
Insert 66



Insert procedure – intuition

Min degree $t = 3$

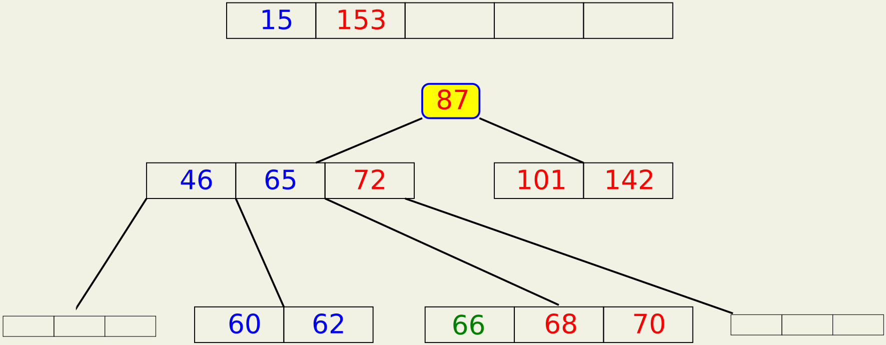
Insert 66



Insert procedure – intuition

Min degree $t = 3$

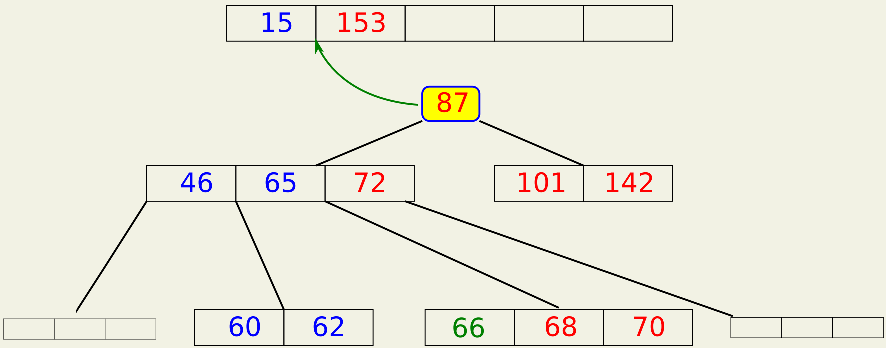
Insert 66



Insert procedure – intuition

Min degree $t = 3$

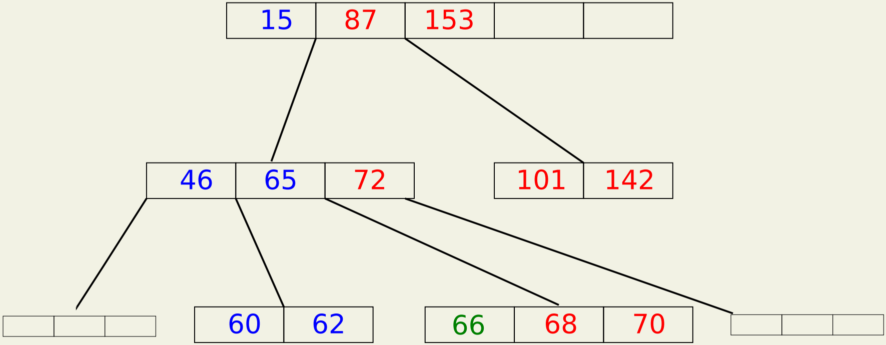
Insert 66



Insert procedure – intuition

Min degree $t = 3$

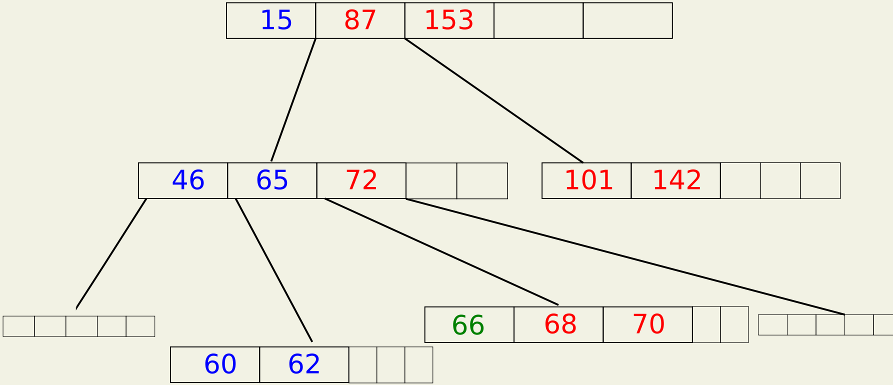
Insert 66



Insert procedure – intuition

Min degree $t = 3$

Insert 66



Insert into BTree

Intuitively, the insert procedure is very straightforward:

- Find the correct leaf to insert the new element.
- If node is not full, insert the new element in it.
- If node is full, split node at median and insert new element.
- Try to insert the median to the parent.
- Recurse upwards splitting nodes to fit all elements.

Note: A BTree increases in height only when the root is split.

Insert procedure from CLRS

The BTree Insert procedure in CLRS avoids backward recursion:

- During the search for the correct leaf to insert the new element:
  - Split every full node along the search path.

Insert procedure from CLRS

The BTree Insert procedure in CLRS avoids backward recursion:

- During the search for the correct leaf to insert the new element:
  - Split every full node along the search path.

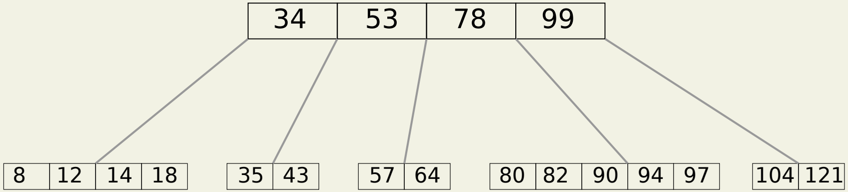
i.e., we split all full nodes along the search path as a pre-emptive action.
So if the leaf needs to be split to insert the new element, the median will certainly fit into parent.
This completely avoids the backward recursion seen in the examples.

Exercise 1

Read the pseudo code from CLRS.

Exercise 2

Min degree $t = 3$.



Insert elements: 10, 79, 47, 29

Delete in one pass

BTree Delete

The idea of Insert in one pass:

- Split all full nodes encountered during search

The idea of Delete in one pass is:

- Merge all nodes that have min number of elements.

Delete from BTree

One pass procedure to delete x from a BTree:
During search for the element x:
If all nodes encountered have more than the min required number of elements:

- Case 1: x is in a leaf.
  Simply delete the element.
- Case 2: x is in an internal node N. Left and right child of x are L and R.
  - 2a: L has more than min number of elements.
    Replace x with the predecessor.
    Delete predecessor.
  - 2b: Symmetric case with R. (Replace with successor and delete)
  - 2c: Neither L nor R has more than min number of elements.
    Merge L, x and R. Delete x.

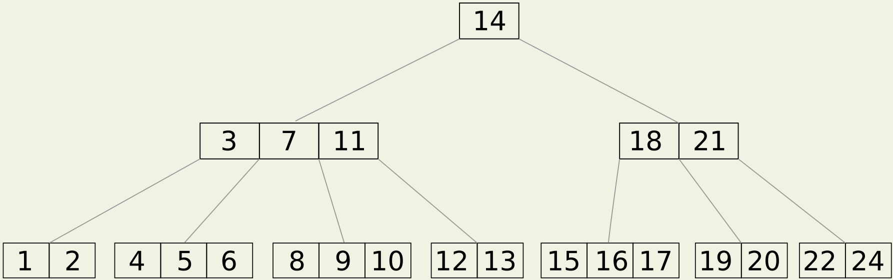
Delete from BTree

During search for the element x:
Else If a node N does not have more than min number of elements.

- Case 3a: An immediate sibling of N has more than min elements.
  Borrow an element from that sibling like so:
  - Move an extra element from sibling of N to parent.
  - Move an element from parent to N.
- Case 3b: Both siblings of N have exactly min elements.
  - Merge N with a sibling.
  - Bring down an element from parent to make median.

Delete from BTree

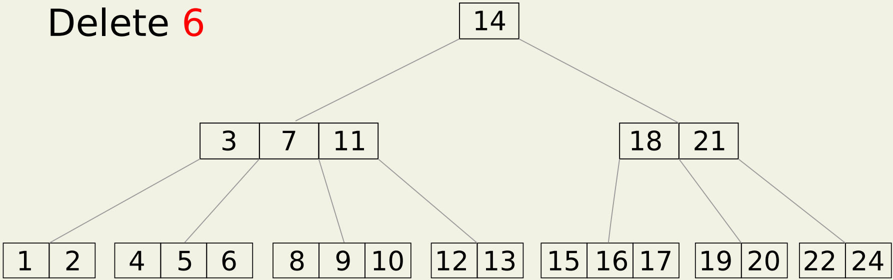
Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

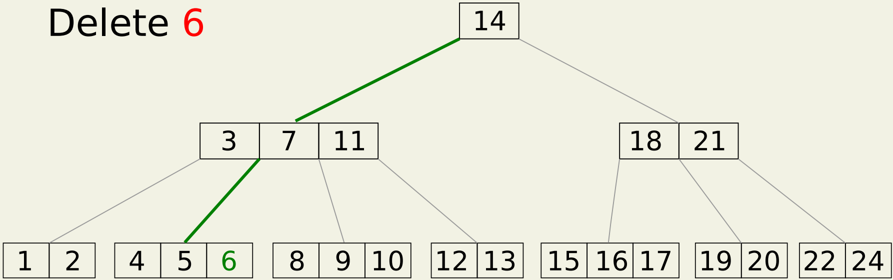
Min degree $t = 3$. So min number of elements is 2.

Delete 6



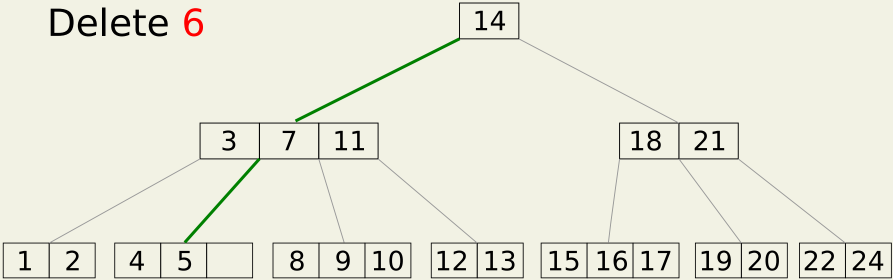
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



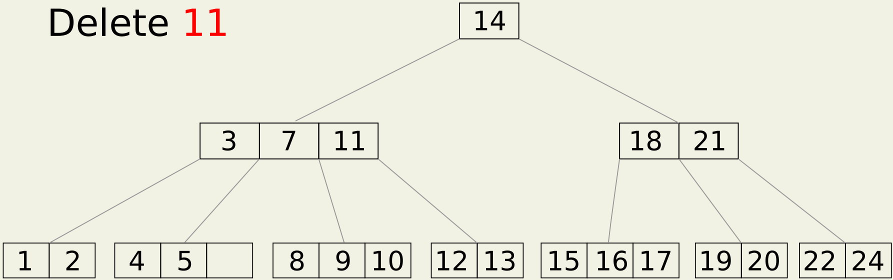
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



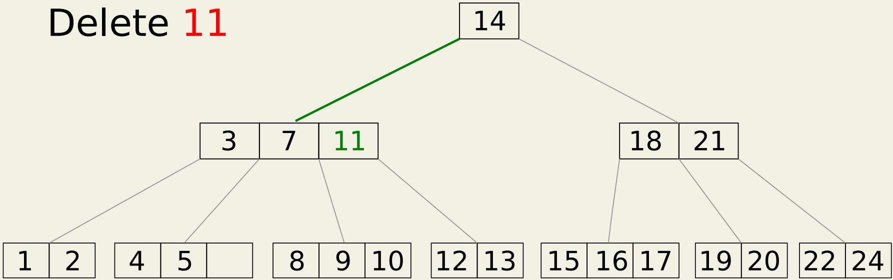
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

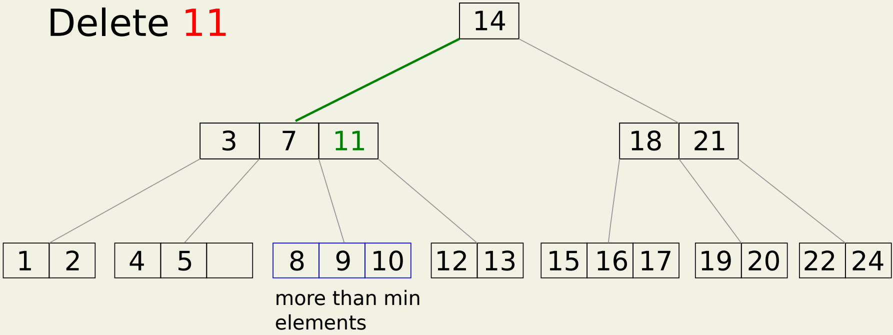
Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

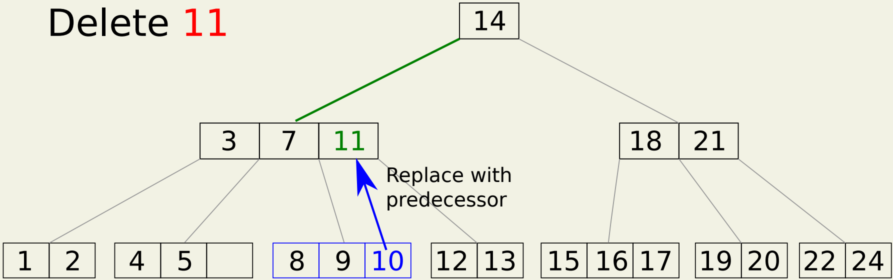
Min degree $t = 3$. So min number of elements is 2.

Delete 11



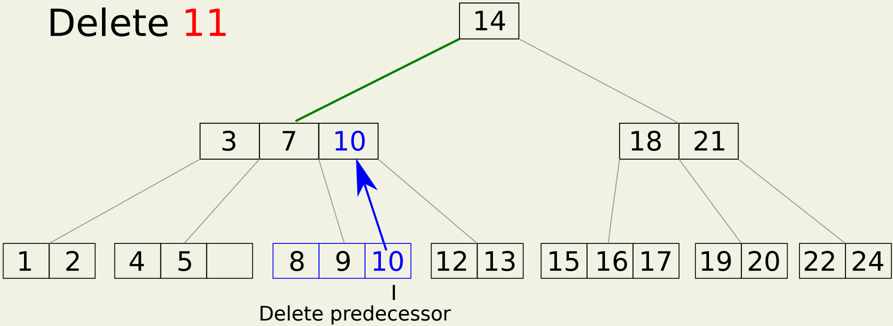
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

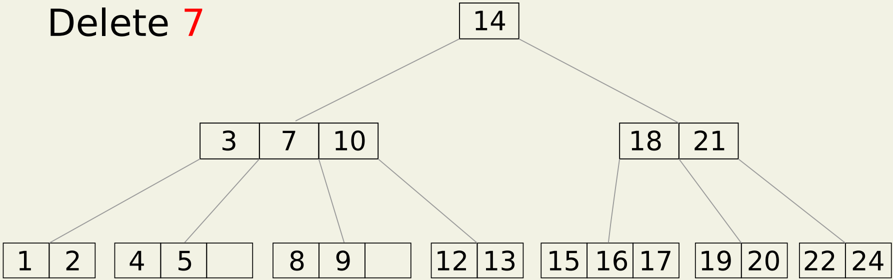
Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

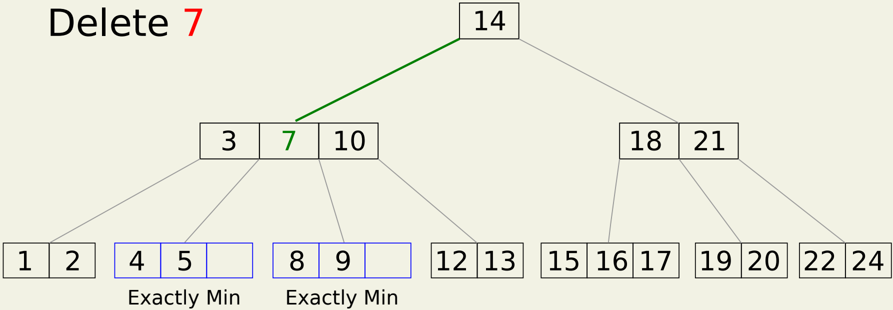
Min degree $t = 3$. So min number of elements is 2.

Delete 7



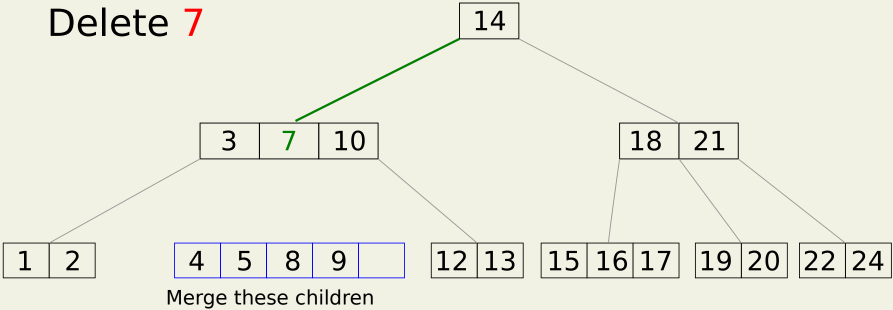
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



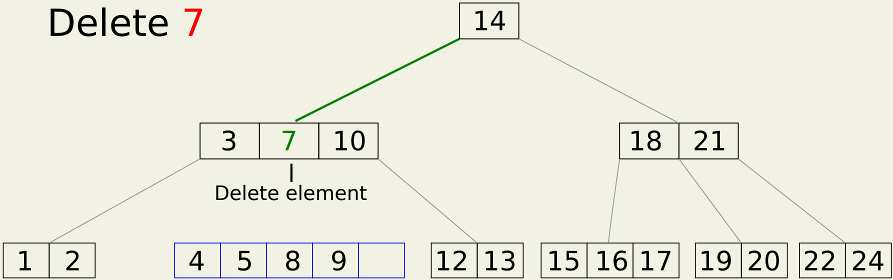
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



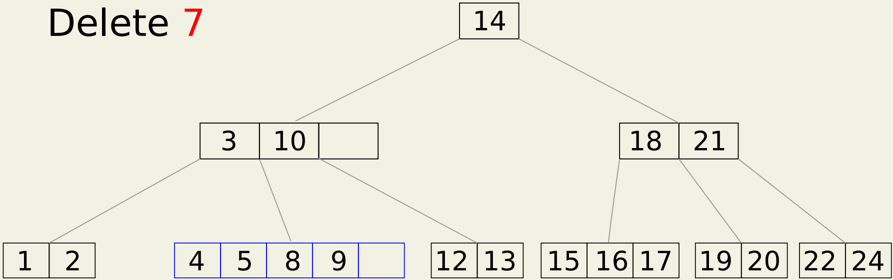
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



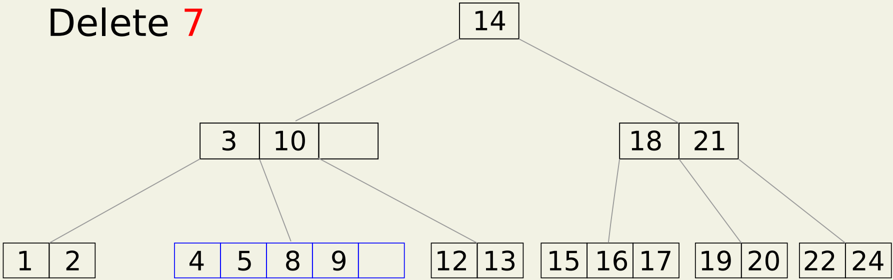
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

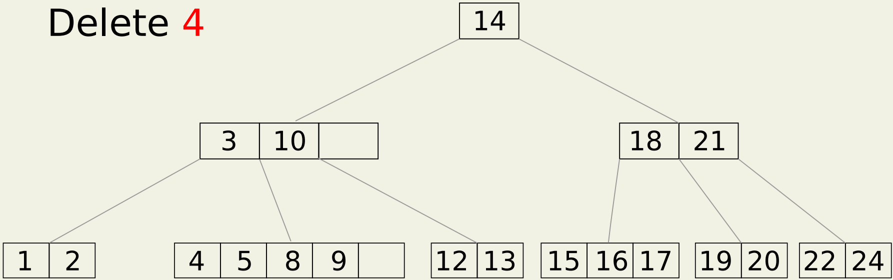
Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

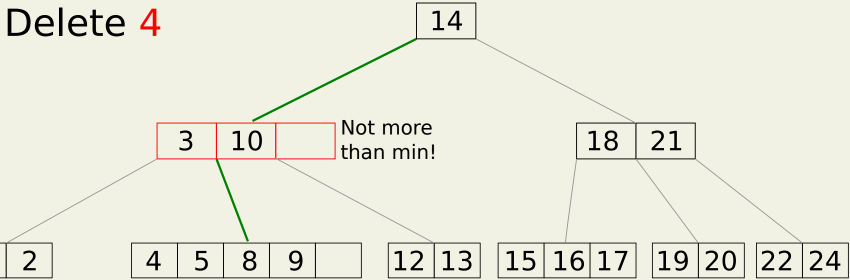
Min degree $t = 3$. So min number of elements is 2.

Delete 4



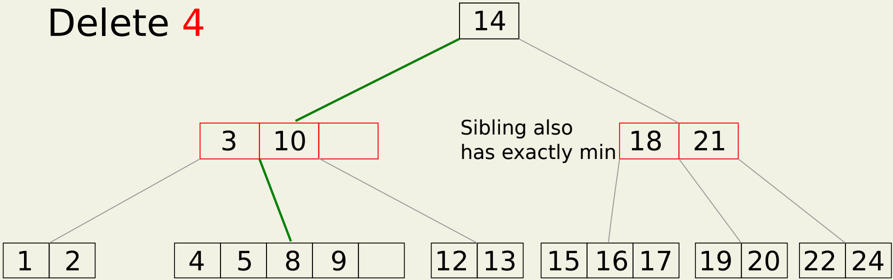
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



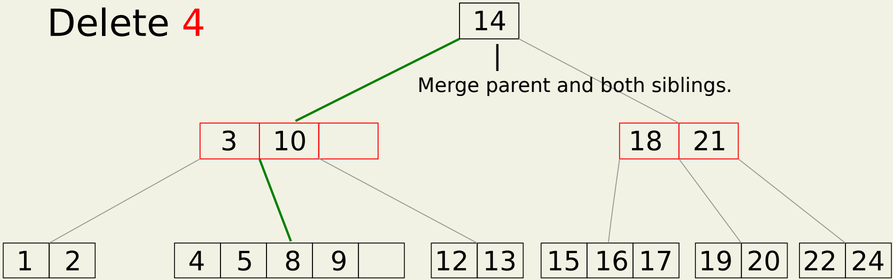
Delete from BTree

Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

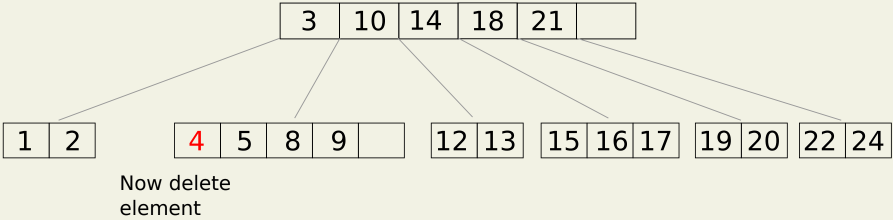
Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

Min degree $t = 3$. So min number of elements is 2.

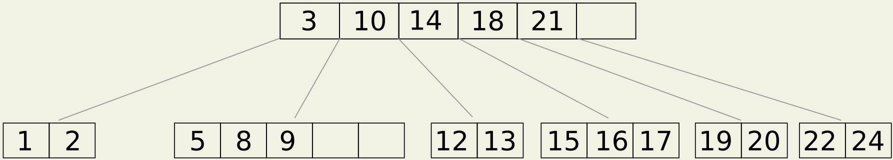
Delete 4



Delete from BTree

Min degree $t = 3$. So min number of elements is 2.

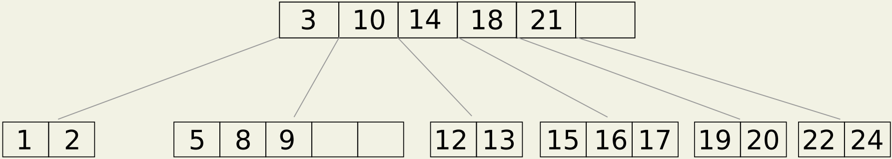
Delete 4



Delete from BTree

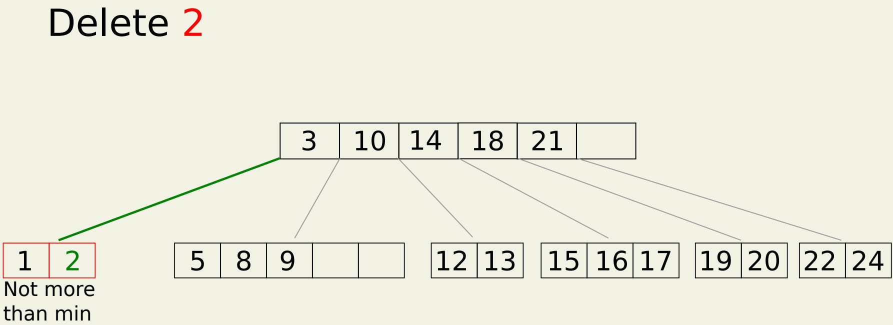
Min degree $t = 3$. So min number of elements is 2.

Delete 2



Delete from BTree

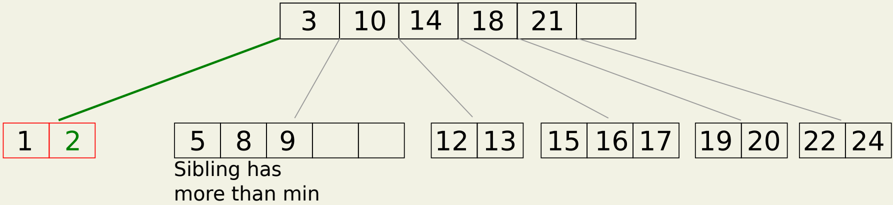
Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

Min degree $t = 3$. So min number of elements is 2.

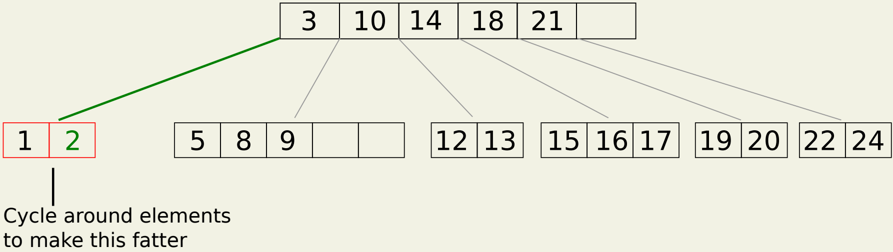
Delete 2



Delete from BTree

Min degree $t = 3$. So min number of elements is 2.

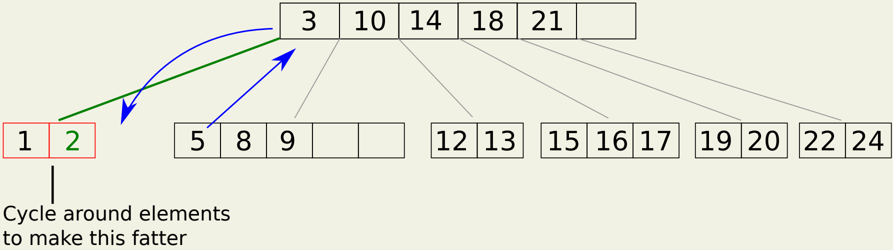
Delete 2



Delete from BTree

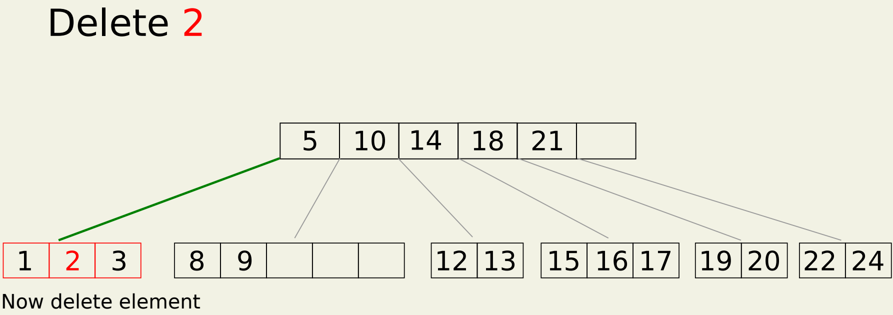
Min degree $t = 3$. So min number of elements is 2.

Delete 2



Delete from BTree

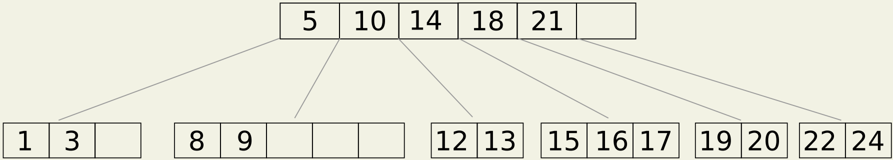
Min degree $t = 3$. So min number of elements is 2.



Delete from BTree

Min degree $t = 3$. So min number of elements is 2.

Delete 2



Now delete element