2-3 Trees

- leaf nodes
  - contain data item
  - all leaves on same level
- nonleaf node
  - has 2 or 3 children
  - two search values
    * largest item in left subtree
    * largest item in middle subtree
- smallest 2-3 tree
  - only one node (leaf node)
- balanced, ordered tree

Example

```
10:47
|
5:8   21:32   63:69
|
5 8 10 21 32 47 63 69
```

Height of tree

- n data items
- all nodes have 2 children, height =
- all nodes have 3 children, height =

Time to Search for 1 element:

Time to List elements in sorted order:

Insertions in 2-3 trees
• Insert new leaf in appropriate place.
• Repeat until all nonleaf nodes have 2 or 3 children
  – if there is a node with 4 children, split the parent into two parent nodes, with 2 children each.
  – if split the root, then add a new root
• Adjust search values along insertion path

Example:

Insert 5

```
  5
```

Insert 21

```
  5 : 21
  /    \
 /      \n 5      21
```

Insert 8

```
  5 : 8
  /   \  
 /     \ 
 5      8
```

Insert 63

```
  5 : 8
  /   \  
 /     \ 
 5      8
```

Insert 69

```
  5 : 8
  /   \  
 /     \ 
 5      8
```

```
  8 : 63
  /   \  
 /     \ 
 5      8
```

```
  5 : 8
  /   \  
 /     \ 
 5      21
```

```
  21 : 63
  /   \  
 /     \ 
 5      8
```

```
  5 : 8
  /   \  
 /     \ 
 5      8
```

```
  5 : 21
  /    \
 /      \n 5      21
```

```
  8 : 63
  /   \  
 /     \ 
 5      8
```

```
  5 : 8
  /   \  
 /     \ 
 5      21
```

```
  21 : 63
  /   \  
 /     \ 
 5      8
```
Insert 32

Insert 7, 19, 25
Time to insert one element from 2-3 tree with n elements:

Deletions in 2-3 trees

Delete x from 2-3 tree

- let p be parent(x)
- if x is root, remove x
- else if p has 3 children
  then remove x
- else (p has 2 children)
  - if p is the root
    then remove x and p (see fig. 1)
  - else (p not the root)
    cases:
      * let s be the other child of p
        let l be the left sibling of p
        let r be the right sibling of p
        (note l nor r may not exist)
        if l has 3 children, replace x by one of l’s children (see fig 2)
        else if r has 3 children, replace x by one of r’s children.
        else (see fig. 3)
          - remove x
            combine p with l (or r)
            make s a child of l (or r)
            rename p to x
            recursively remove x

Figure 1:

Figure 2:
Figure 3:

Example

Delete 47

Delete 63
Example

Delete 47

Time to delete one element from 2-3 tree with n elements: