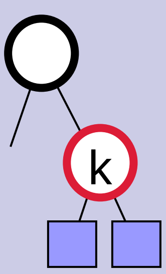
- Insertion in red-black trees
- a-b trees
  - What are they?
  - Insertion and deletion in a-b trees

Insertion in red-black trees

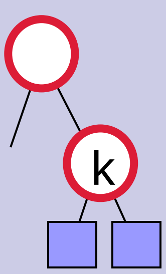
- Let k be the key being inserted
- As in the case of a BST we first search for k; this gives us the place where we have to insert k.
- We create a new node with key k and insert it at this place.
- The new node is colored red.

Insertion(2)

- Since inserted node is colored red, the black height of the tree remains unchanged.
- However, if the parent of inserted node is also red then we have a double red problem.



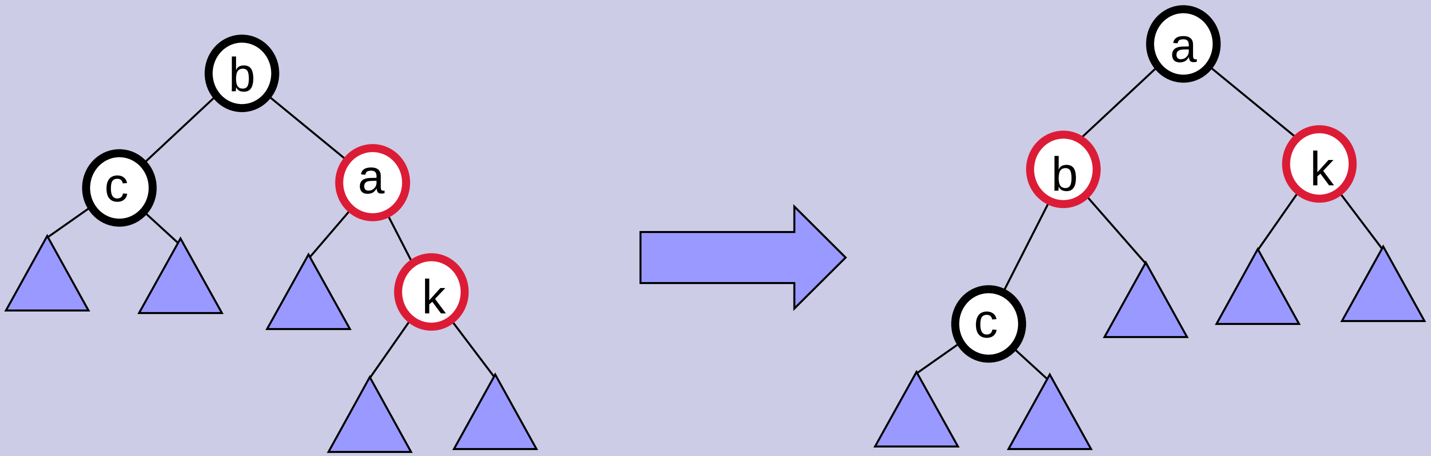
No problem



Double red problem

Insertion: case 1

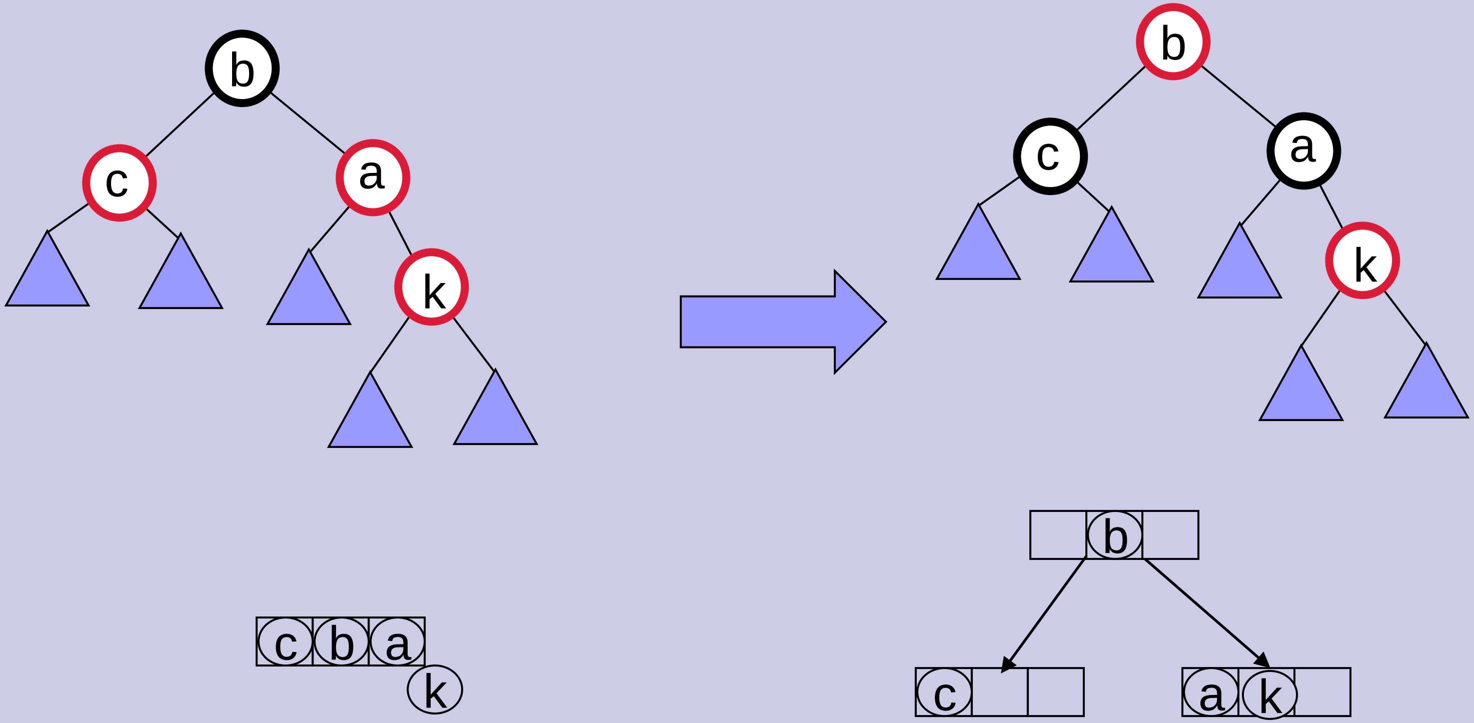
- Parent of inserted node (a) is red.
- Parent of (a) must be black (b)
- The other child of (b) is black (c).



The 2-4 tree node contains {b,a,k} and is malformed.
The rotation corrects the defect.

Insertion: Case 2

- Parent of inserted node (a) is red.
- Parent of (a) must be black (b)
- The other child of (b) is also red (c).



Insertion: case 2 (contd)

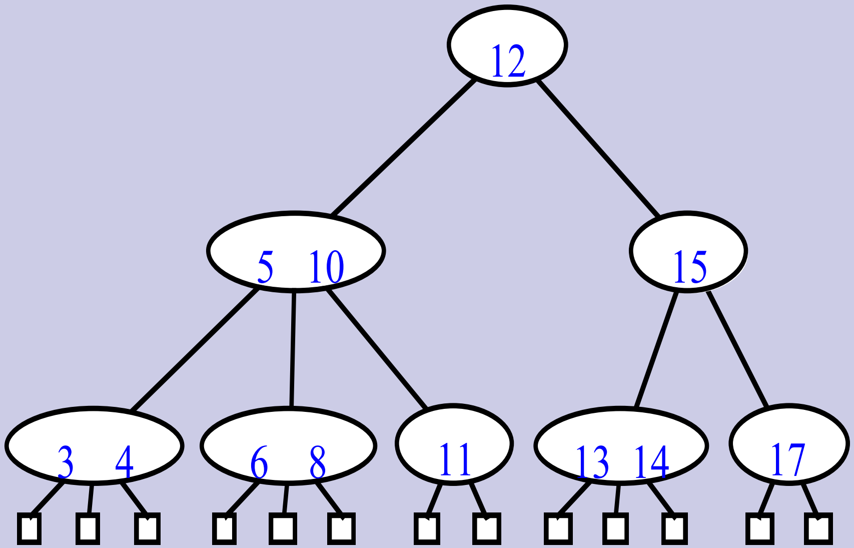
- The parent of b could also be red. In that case, the double red problem moves up a level.
- We repeat this process at the next level.
- Eventually, we might color the root red.
- In this case we recolor the root black. This increases the black depth of every external node by 1.
- In the 2-4 tree this corresponds to splitting the root.

Insertion and Deletion: Summary

- In both insertion and deletion we need to make at most one rotation.
- We might have to move up the tree but in doing so we only recolor nodes.
- Time taken is O(log n)

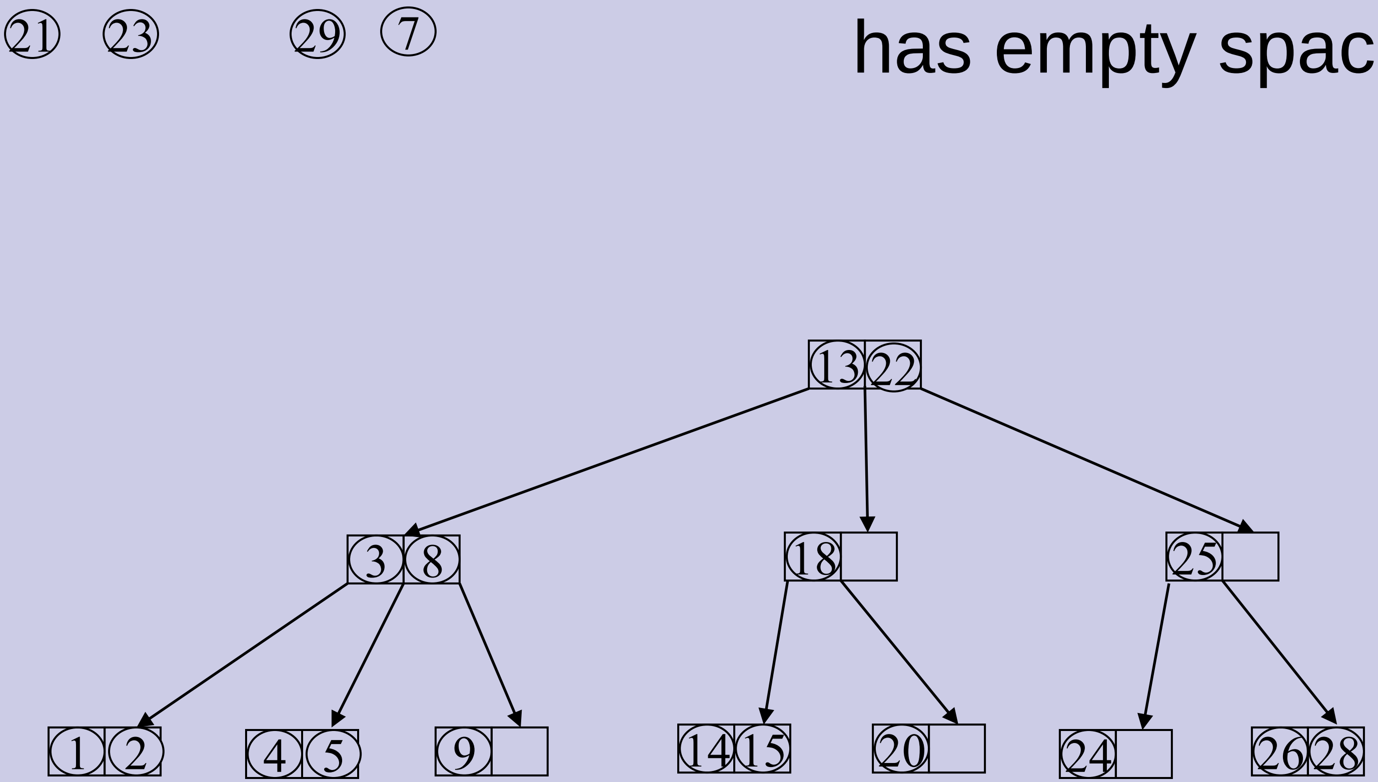
(a,b) Trees

- A multiway search tree.
- Each node has at least *a* and at most *b* children.
- Root can have less than *a* children but it has at least 2 children.
- All leaf nodes are at the same level.
- Height *h* of (a,b) tree is at least $\log_b n$ and at most $\log_a n$.



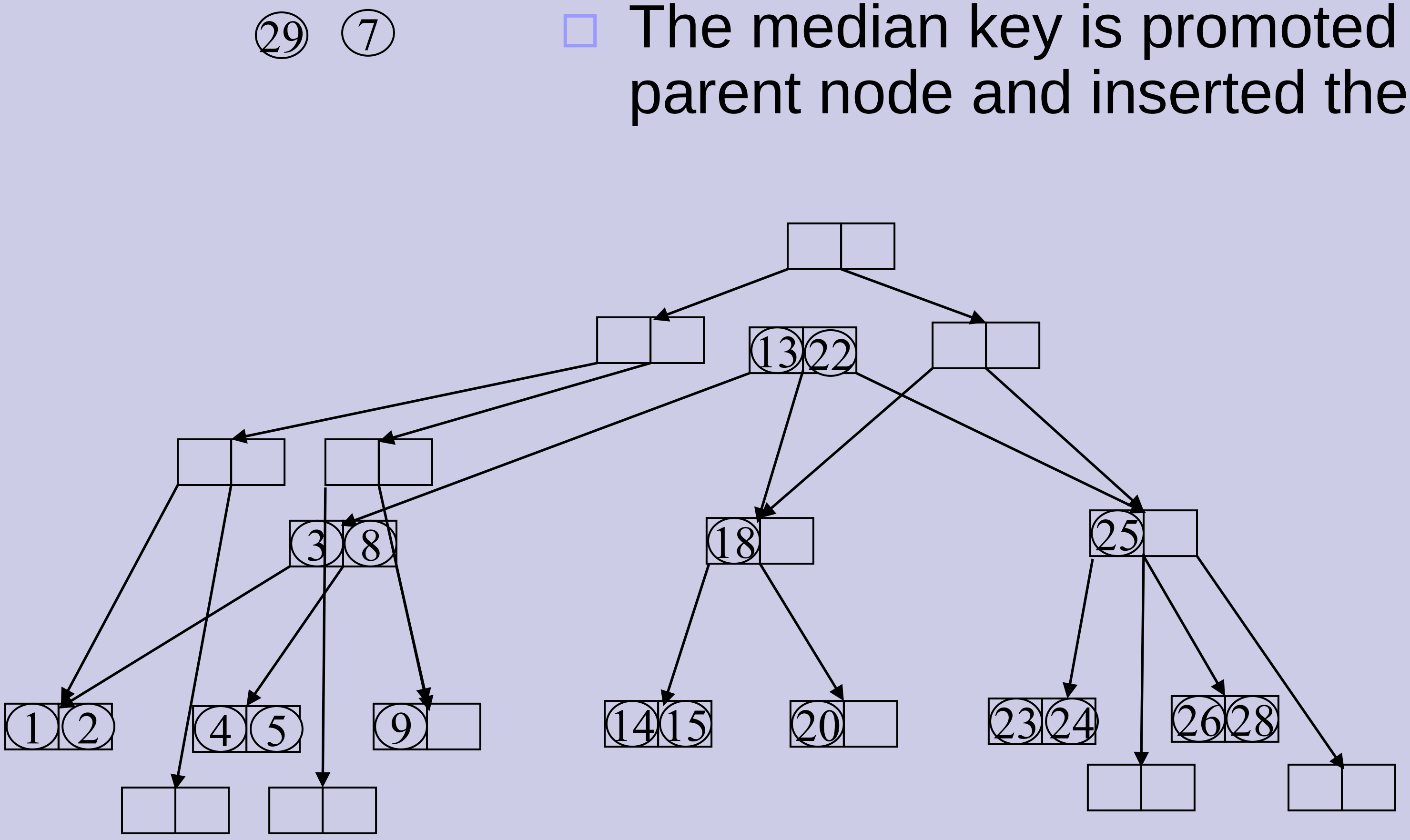
Insertion

- No problem if the node has empty space



Insertion(2)

- Nodes get split if there is insufficient space.
- The median key is promoted to the parent node and inserted there

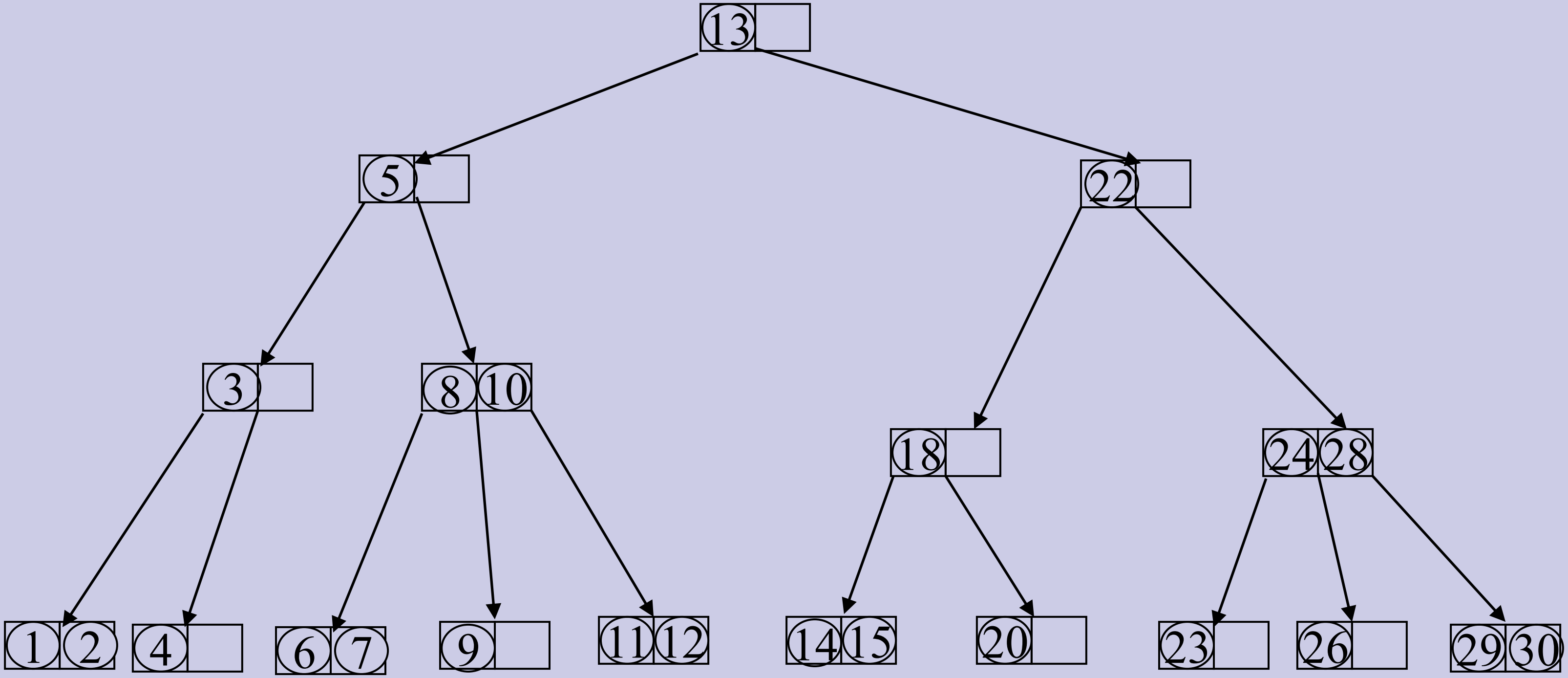


Insertion(3)

- A node is split when it has exactly *b* keys.
- One of these is promoted to the parent and the remaining are split between two nodes.
- Thus one node gets $\lceil \frac{b-1}{2} \rceil$ and the other $\lfloor \frac{b-1}{2} \rfloor$ keys.
- This implies that a-1 >= $\lfloor \frac{b-1}{2} \rfloor$

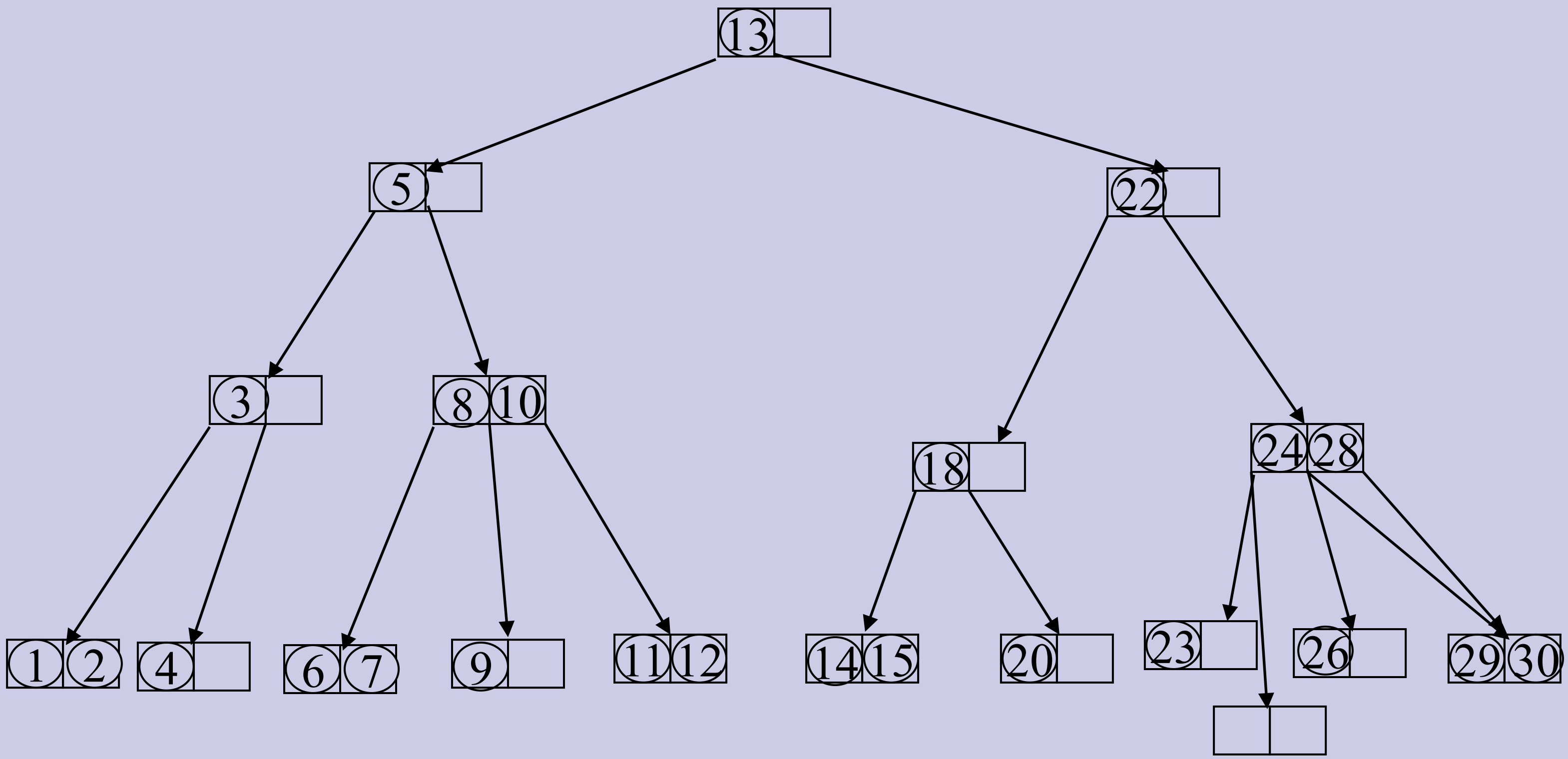
Deletion

- If after deleting a key a node becomes empty then we borrow a key from its sibling.
- Delete 20



Deletion(2)

- If sibling has only one key then we merge with it.
- The key in the parent node separating these two siblings moves down into the merged node.
- Delete 23



Deletion(3)

- In an (a,b) tree we will merge a node with its sibling if the node has *a*-2 keys and its sibling has *a*-1 keys.
- Thus the merged node has 2(*a*-1) keys.
- This implies that 2(*a*-1) <= *b*-1 which is equivalent to *a*-1 <= $\left\lfloor \frac{b-1}{2} \right\rfloor$.
- Earlier too we argued that *a*-1 <= $\left\lfloor \frac{b-1}{2} \right\rfloor$
- This implies *b* >= 2*a*-1
- For a=2, *b* >= 3

Conclusion

- The height of a (a,b) tree is $O(\log n)$.
- $b >= 2a-1$.
- For insertion and deletion we take time proportional to the height.