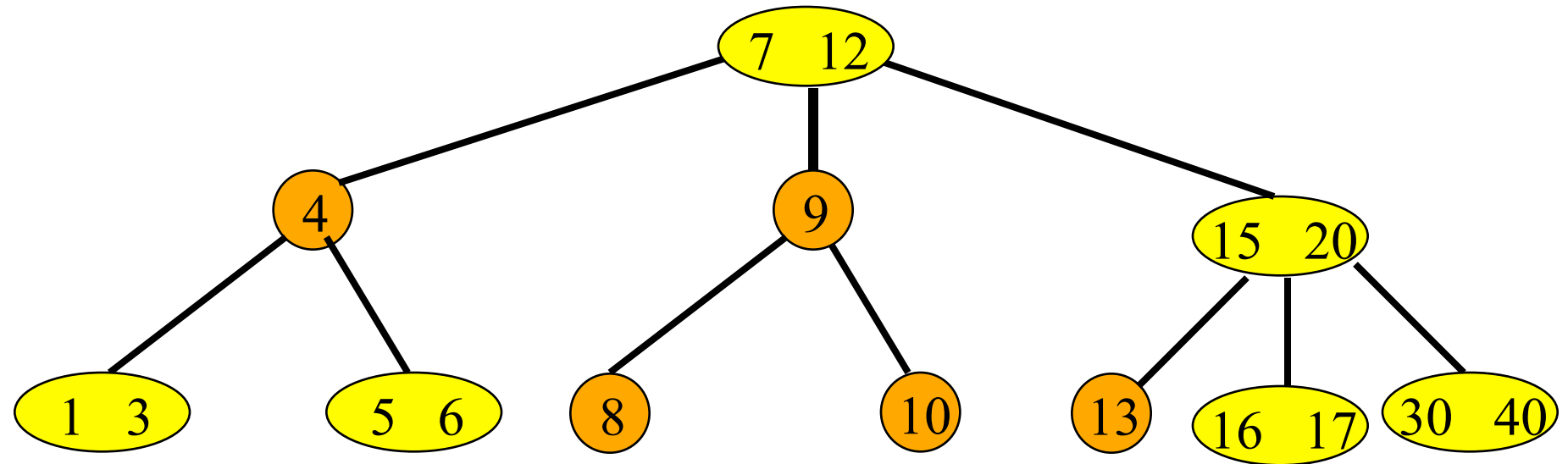


B-Tree – Inserts

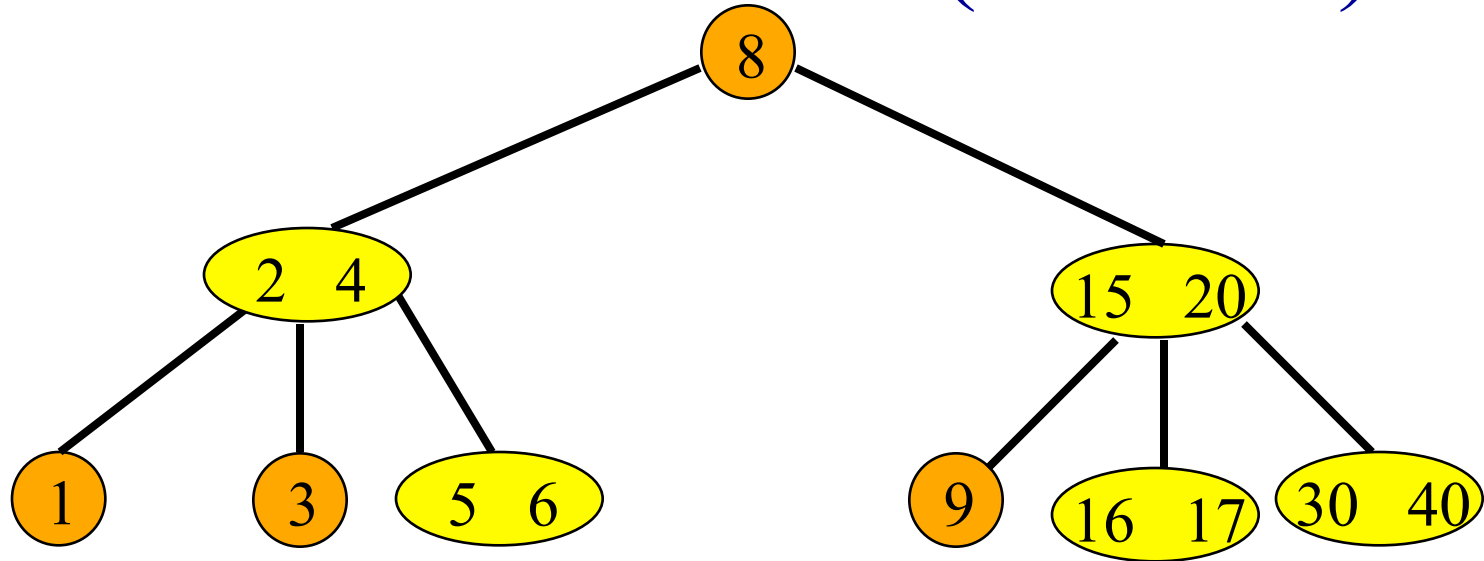


Insert 14.

Insert 2.

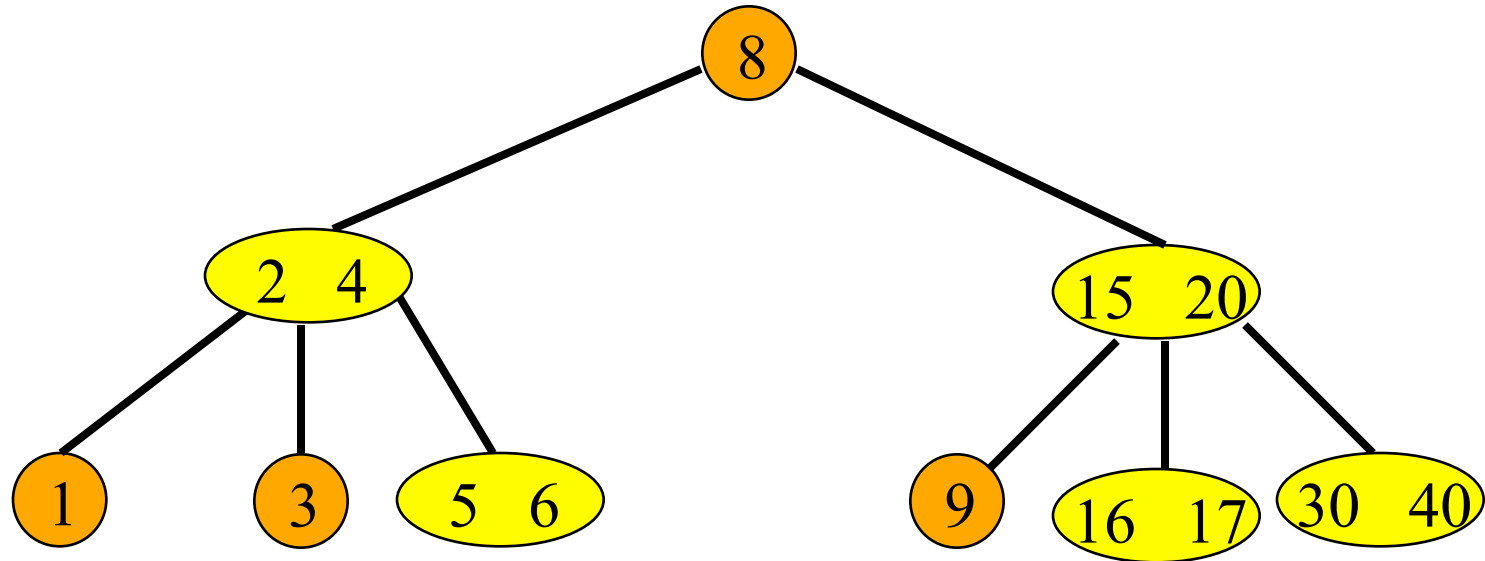
Insert 18.

B-Tree – Delete (2-3 tree)



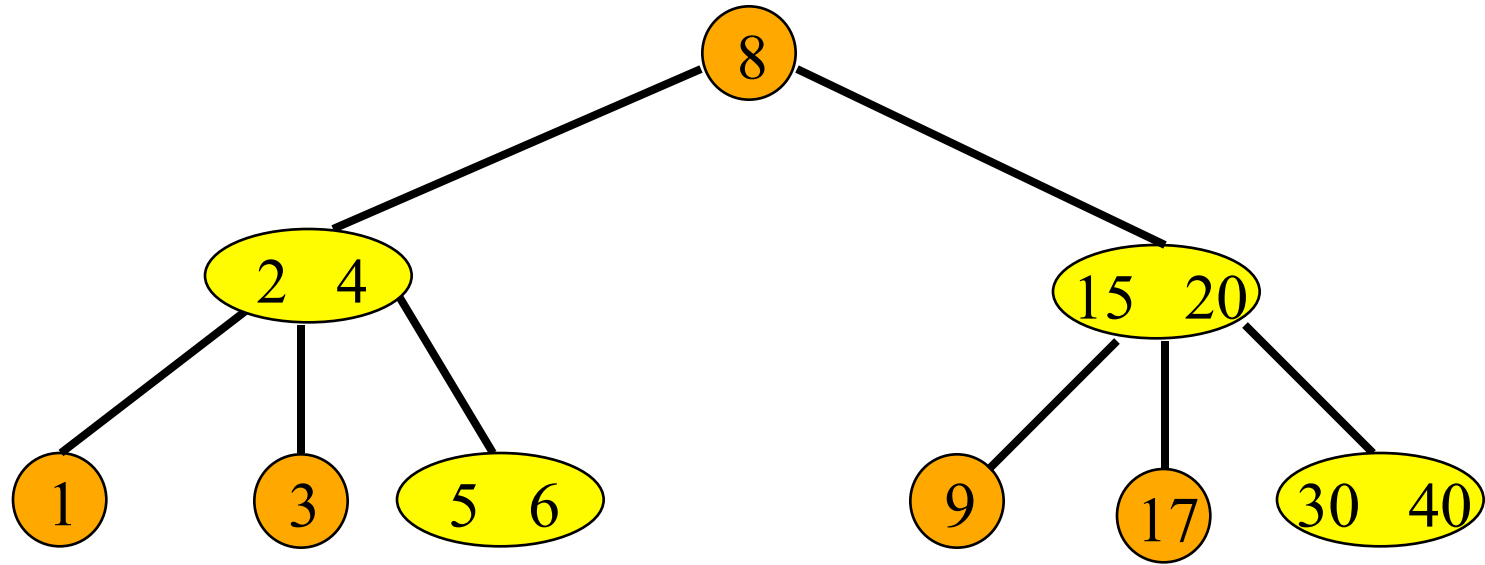
- Delete the pair with key = 8.
- Transform deletion from interior into deletion from a leaf.
- Replace by largest in left subtree.

Delete From A Leaf



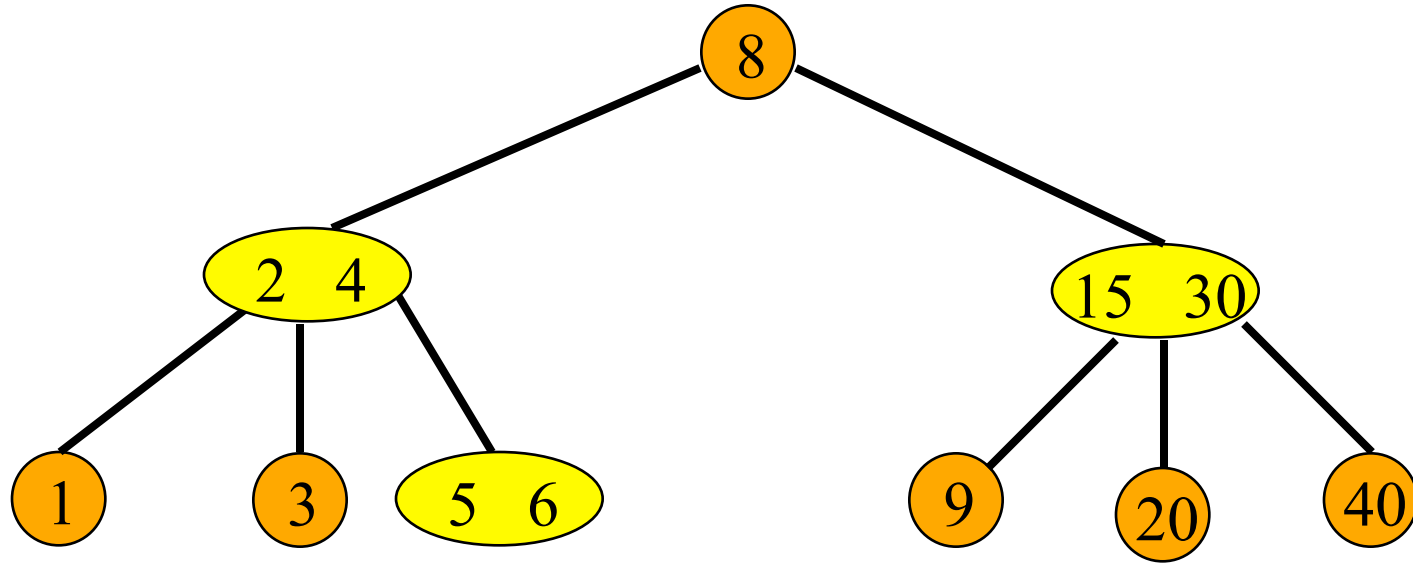
- Delete the pair with key = 16.
- 3-node becomes 2-node.

Delete From A Leaf



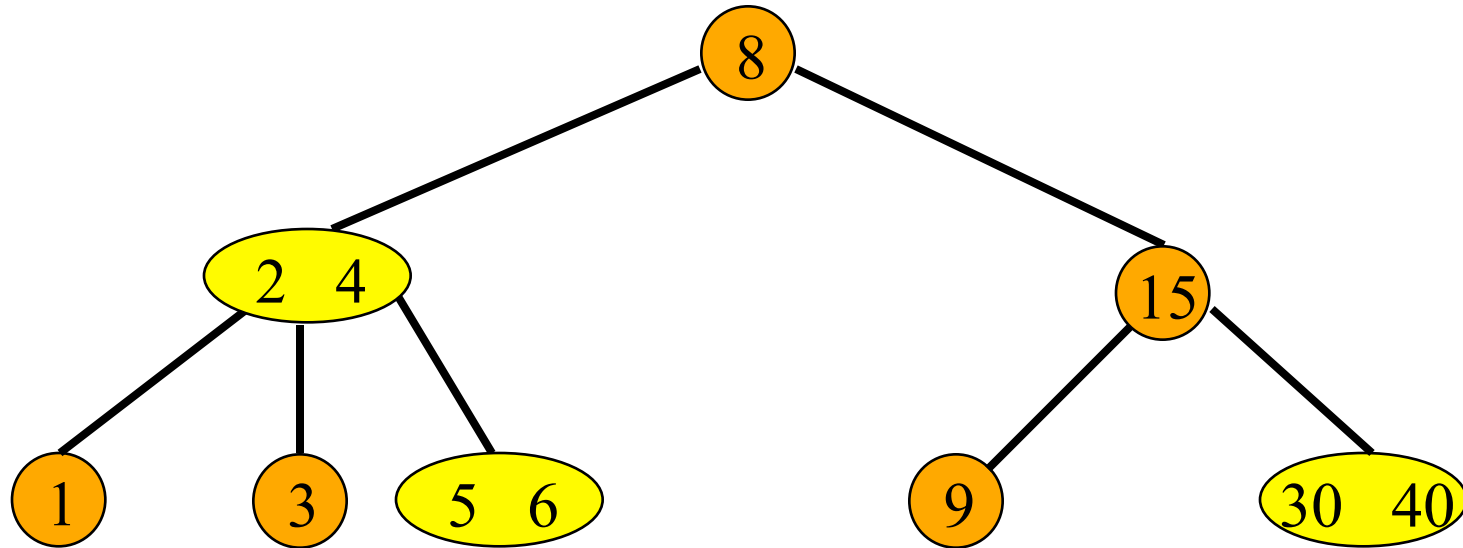
- Delete the pair with key = 17.
- Deletion from a 2-node.
- Check one sibling and determine if it is a 3-node.
- If so borrow a pair and a subtree via parent node.

Delete From A Leaf



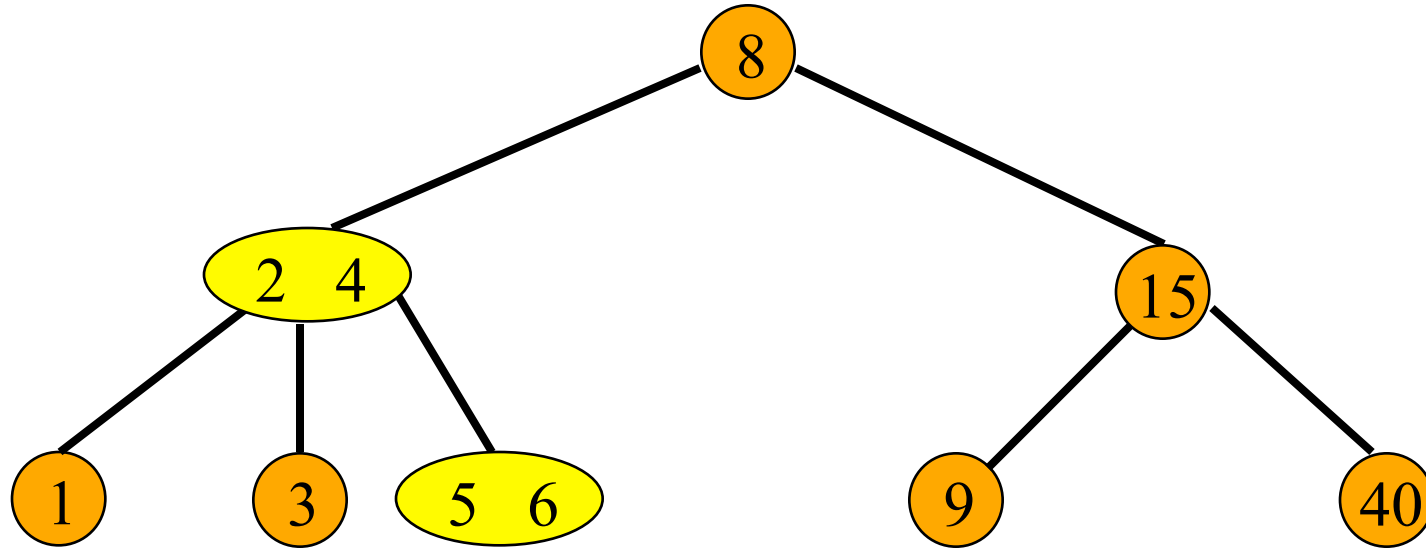
- Delete the pair with key = 20.
- Deletion from a 2-node.
- Check one sibling and determine if it is a 3-node.
- If not, combine with sibling and parent pair.

Delete From A Leaf



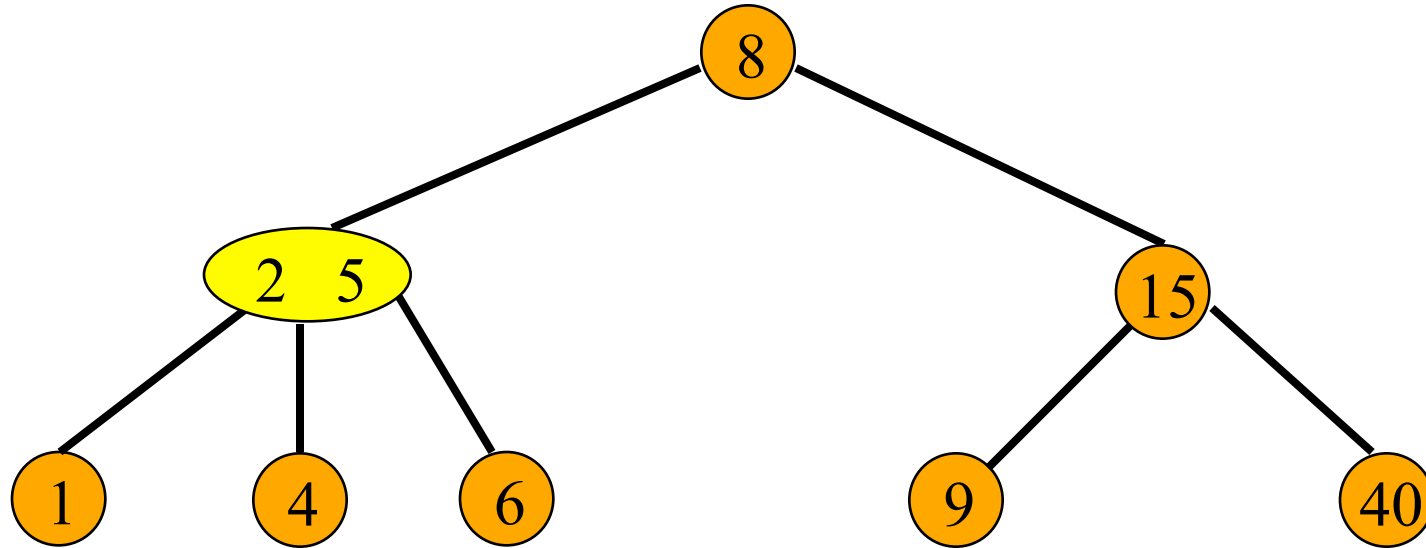
- Delete the pair with key = 30.
- Deletion from a 3-node.
- 3-node becomes 2-node.

Delete From A Leaf



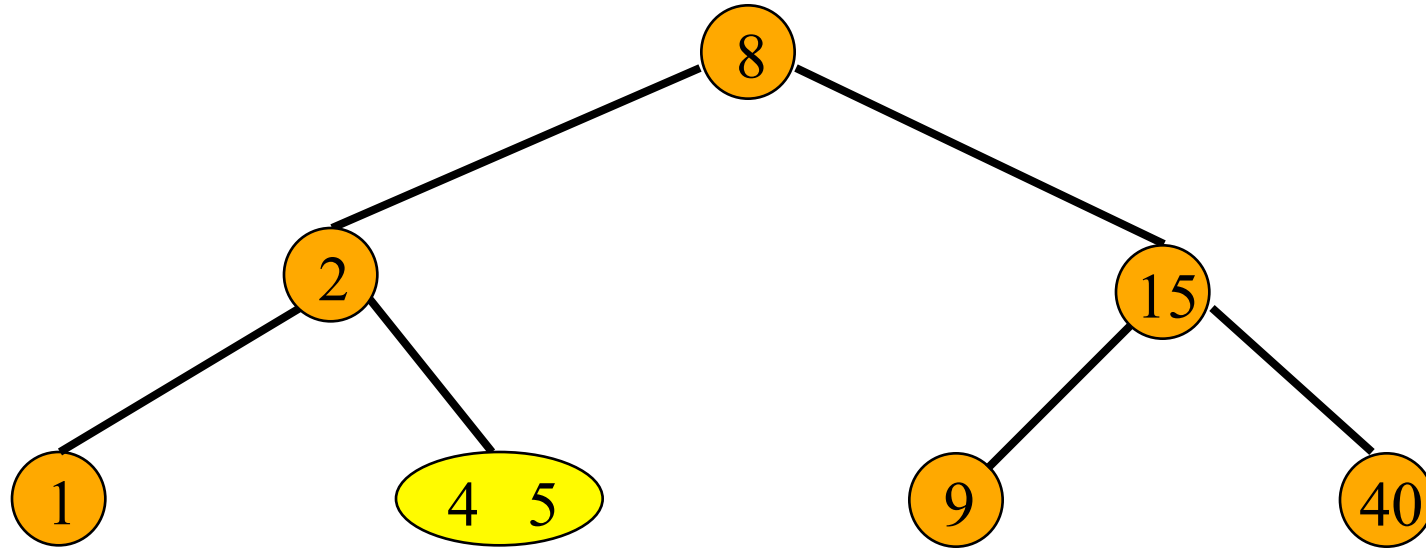
- Delete the pair with key = 3.
- Deletion from a 2-node.
- Check one sibling and determine if it is a 3-node.
- If so borrow a pair and a subtree via parent node.

Delete From A Leaf



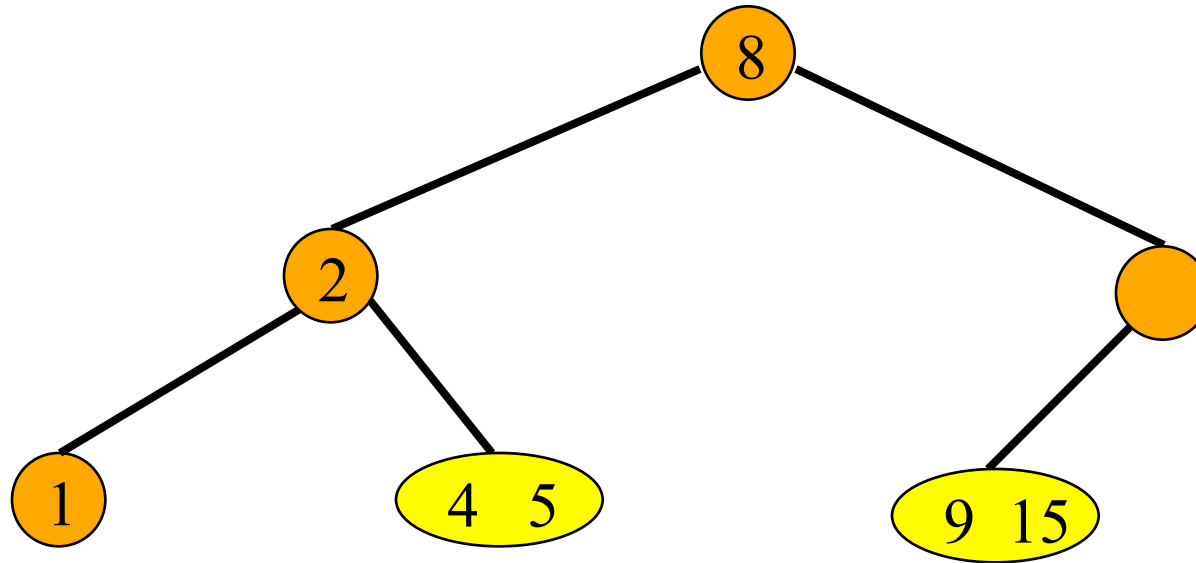
- Delete the pair with key = 6.
- Deletion from a 2-node.
- Check one sibling and determine if it is a 3-node.
- If not, combine with sibling and parent pair.

Delete From A Leaf



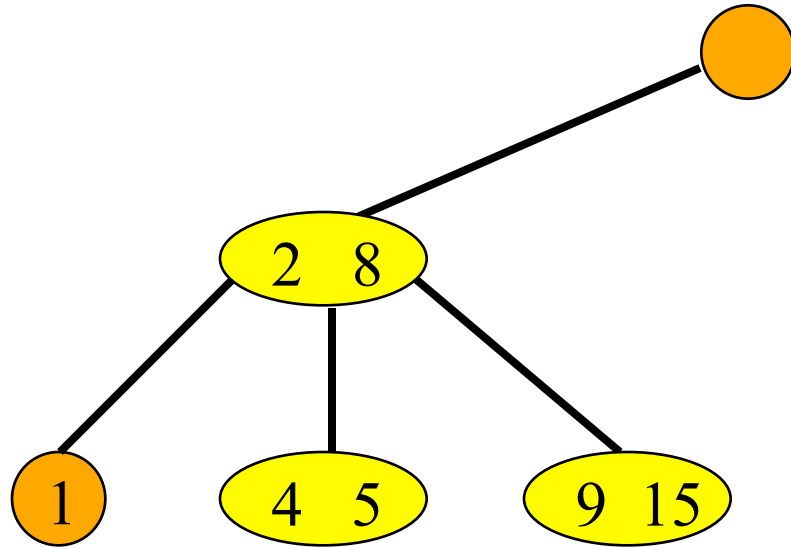
- Delete the pair with key = 40.
- Deletion from a 2-node.
- Check one sibling and determine if it is a 3-node.
- If not, combine with sibling and parent pair.

Delete From A Leaf



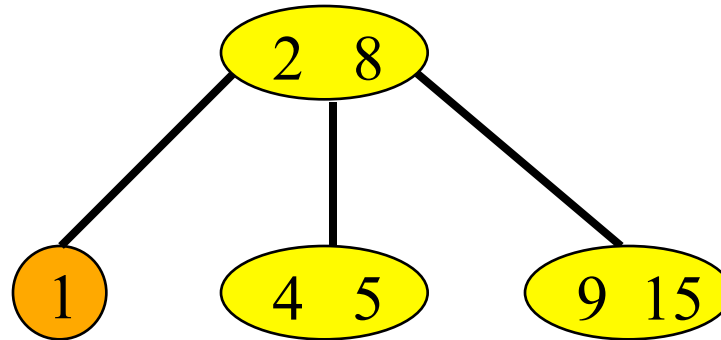
- Parent pair was from a 2-node.
- Check one sibling and determine if it is a 3-node.
- If not, combine with sibling and parent pair.

Delete From A Leaf



- Parent pair was from a **2**-node.
- Check one sibling and determine if it is a **3**-node.
- No sibling, so must be the root.
- Discard root. Left child becomes new root.

Delete From A Leaf

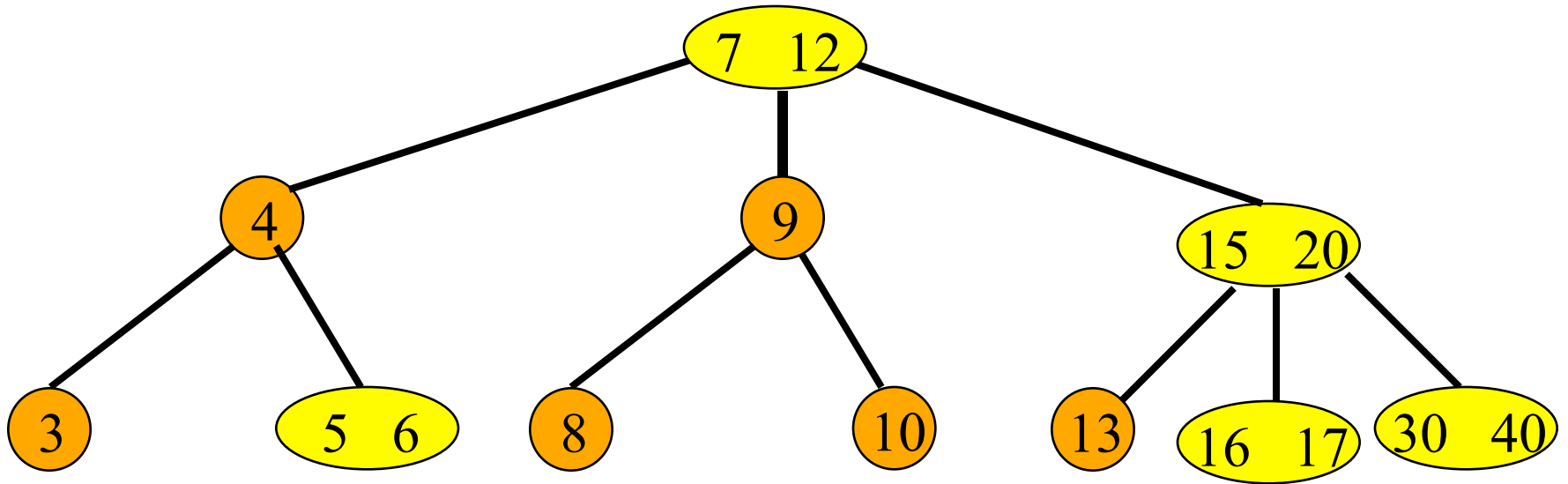


- Height reduces by 1.

Delete A Pair

- Deletion from interior node is transformed into a deletion from a leaf node.
- Deficient leaf triggers bottom-up borrowing and node combining pass.
- Deficient node is combined with an adjacent sibling who has exactly $\text{ceil}(m/2) - 1$ pairs.
- After combining, the node has $[\text{ceil}(m/2) - 2]$ (original pairs) + $[\text{ceil}(m/2) - 1]$ (sibling pairs) + 1 (from parent) $\leq m - 1$ pairs.

B-Tree – Delete



Delete 7.

Delete 3.

Delete 8.

Bool BT-Delete(x, k)

1. If leaf[x]
2. if In(x, k) then BT-Delete-leaf(x,k)
3. return #key > Ceil(m/2)-2?
4. false:true

Bool BT-Delete(x, k)

1. If not leaf[x]
2. if In(x,k)
3. then Select&Replace(x,k, k')
4. flag \leftarrow BT-Delete(x,k')
5. If flag
6. then Borrow/Merge
7. return #key > Ceil(m/2)-2?
8. false:true

Bool BT-Delete(x, k)

1. if not leaf[x] && not In(x,k)
2. then flag \leftarrow BT-Delete(Ci[x],k)
3. If flag
4. then Borrow/Merge
5. return #key > Ceil(m/2)-2?
6. false:true

- **Exercises:** P623-2, 4