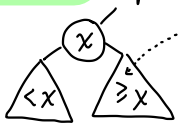
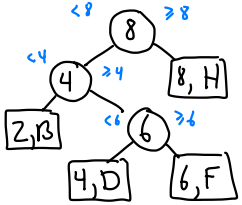
Extended Binary Search Trees

- **Internal nodes** - "Splitting" keys

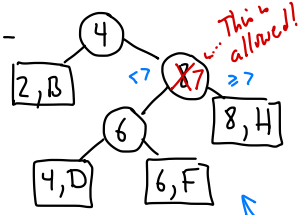


- **External nodes** - Store key + value (but no child links)

E.g. Store (K,V) pairs
{(2,B),(4,D),(6,F),(8,H)}



-or-



This is allowed!!

Dictionary operations?

- Straightforward generalizations over standard BSTs but "twists"

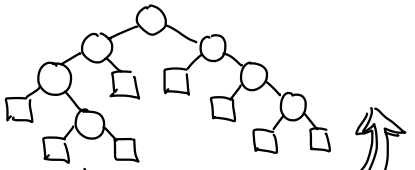
Find: Same but need to descend to leaf level

find(8) → "H"
find(7) → null
→ splitters not in dictionary!

Recall: Extended Binary Trees

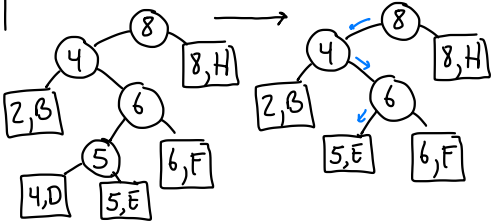
- **Internal nodes** - have non-null left + right
- **External nodes** - leaves

E.g.

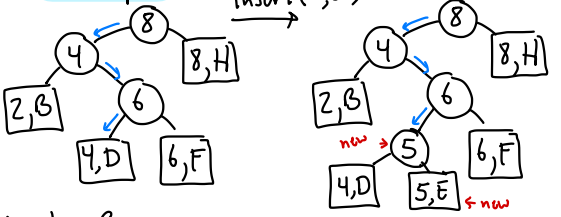


Supplemental: Extended Binary Search Trees I

Example:



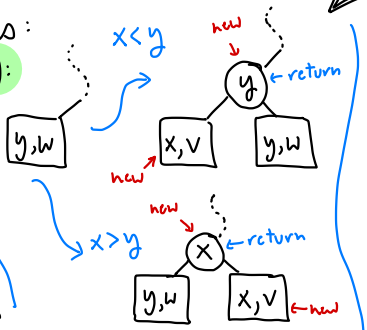
Example:



Insert: Find appropriate leaf.

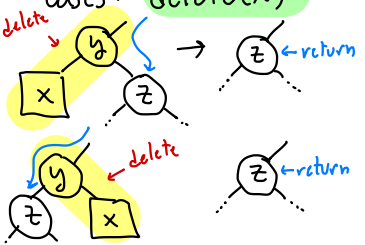
- keys match? → Error!
- else cases:

insert(x,v):



Delete: Find leaf

Cases: delete(x)



[z may also be extern]

Parent: Node

```
abstract class Node {
    Key key;
    abstract Value find(Key x);
    abstract Node
        insert(Key x, Value v);
    abstract Node
        delete(Key x)
}
```

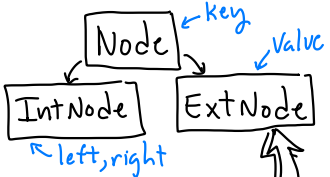
"local" utilities defined in subclasses

Java - Implementation Notes

```
class XBSTree<Key, Value> {
    ⋮
}
```

extends Comparable<Key>

Node types:



Supplemental: Extended Binary Search Trees II

Internal Node:

```
class IntNode extends Node {
    Node left, right;
    Value find(...)
          insert(...)
          delete(...)
}
```

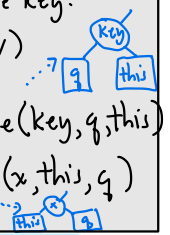
as above not abstract

External Node:

```
class ExtNode extends Node {
    Value value;
    Value find(...)
          insert(...)
          delete(...)
}
```

as before

```
ExtNode: insert(Key x, Value v) {
    if (x == key) Error! Duplicate key!
    else { q = new ExtNode(x,v)
        if (x < key)
            return new IntNode(key, q, this)
        else return new IntNode(x, this, q)
    }
```




```
IntNode: insert(Key x, Value v) {
    if (x < key) left = left.insert(x,v)
    else /* x ≥ key */ ← equality allowed!
        { right = right.insert(x,v)
    return this ← return pointer to updated tree
```

XBSTree:

init: null

```
private data: Node root;
public methods:

Value find(Key x) → root.find(x)
insert(Key x, Value v)
        → root = root.insert(x,v)
delete(Key x) → root = root.delete(x)
```