CMPT 225 D100 LAB09

TA

TOPICS FOR TODAY

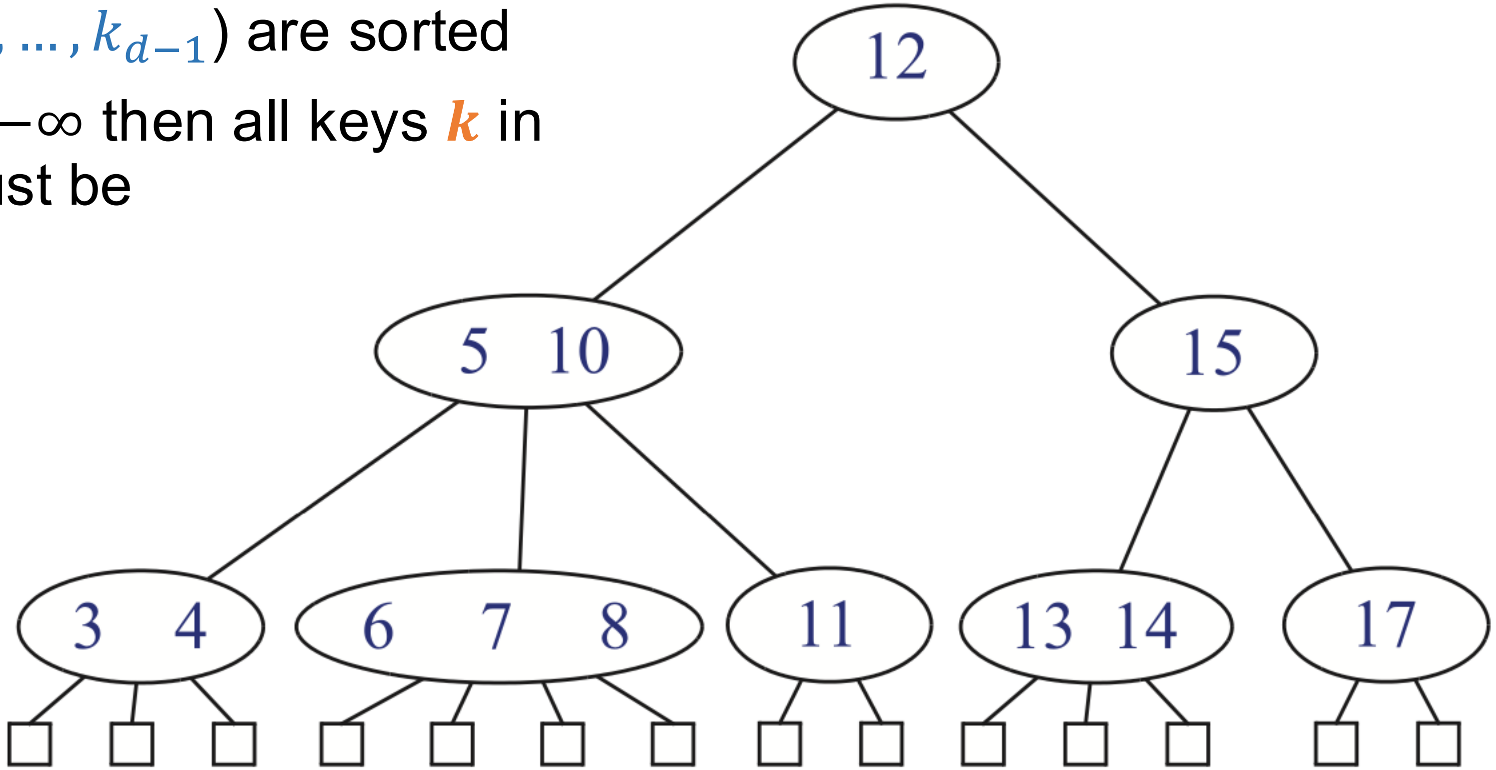
(2,4) trees

- Inserting keys
- Splitting nodes

Multi-way Search Trees

Multi-way Search Trees properties:

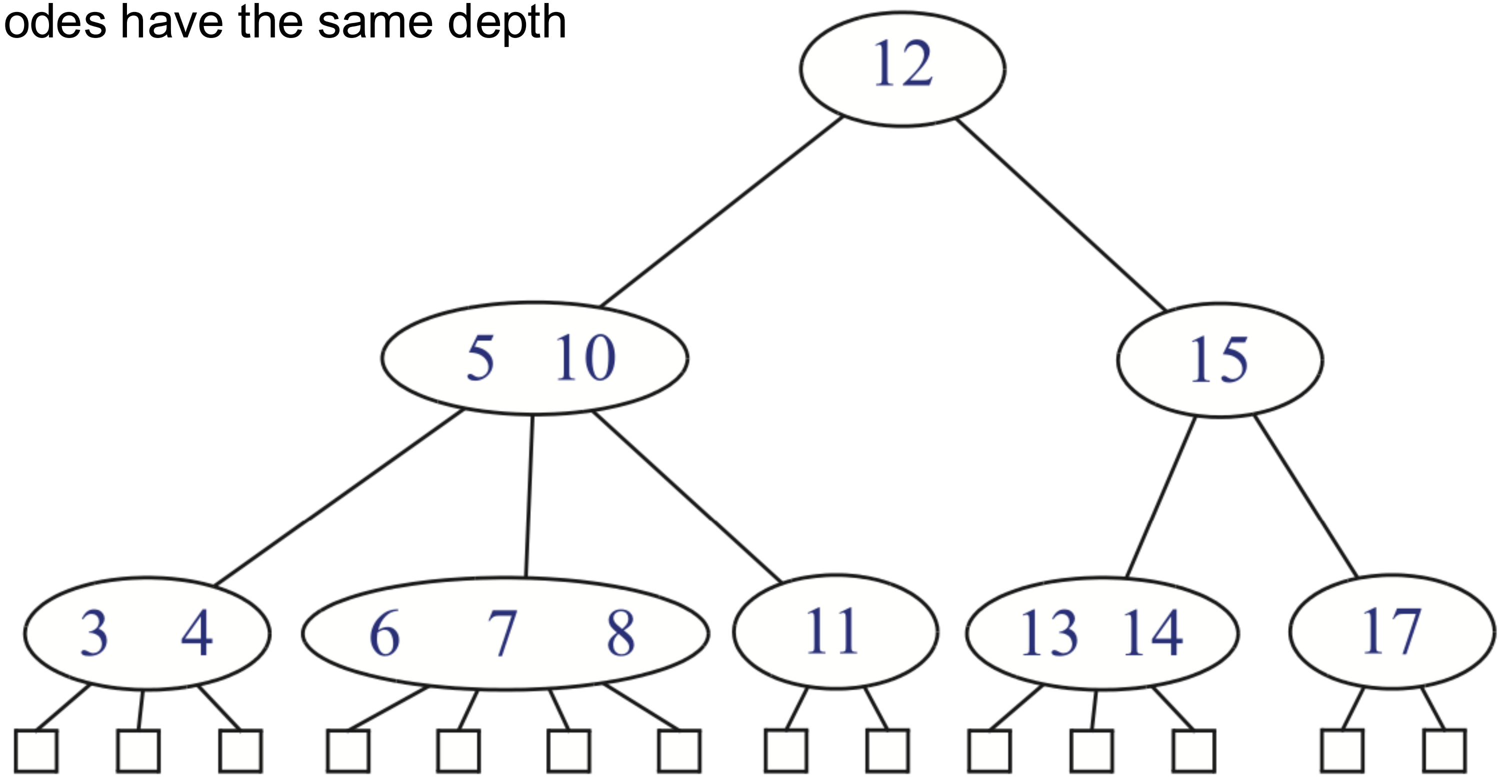
- Each internal node of T has at least two children
- All keys in each node $(k_1, k_2, \dots, k_{d-1})$ are sorted
- Assume $k_0 = -\infty$ and $k_d = -\infty$ then all keys $\boldsymbol{k}$ in child node $(\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_d})$ must be in the range $k_{i-1} < \boldsymbol{k} < k_i$



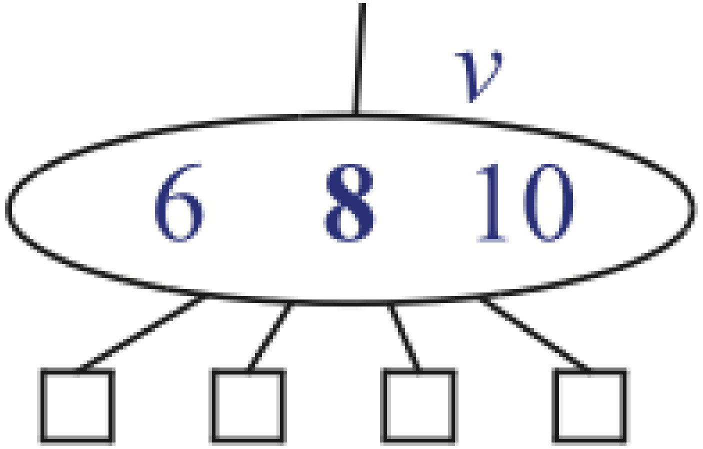
(2, 4) Tree

(2, 4) tree properties:

- ***Size Property*: **Every internal node has at most four children
- ***Depth Property*: **All the external nodes have the same depth



(2, 4) Tree Node

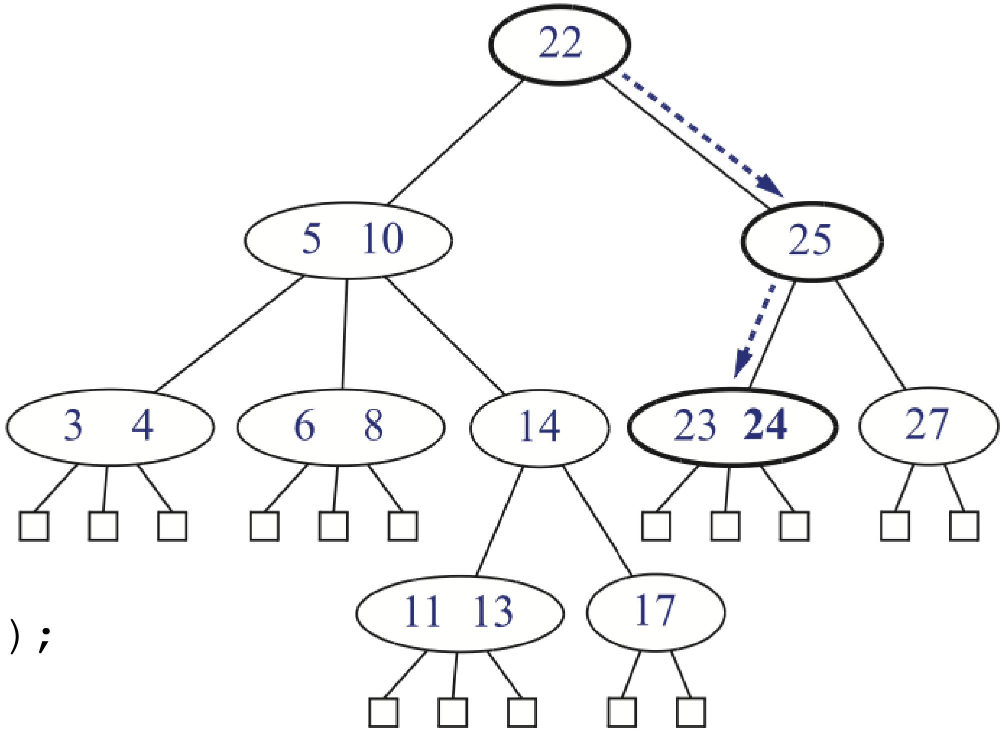


```
struct Node {
    vector<int> keys;
    vector<Node*> children;
    Node* parent;
};
```

- Vectors allow us to easily change the number of keys if needed to be 1, 2, or 3.
- We can also adjust the number of children to be 2, 3, or 4.
- The number of keys is easy to find in a vector since we can use: `v->keys.size();`

(2, 4) Tree Search

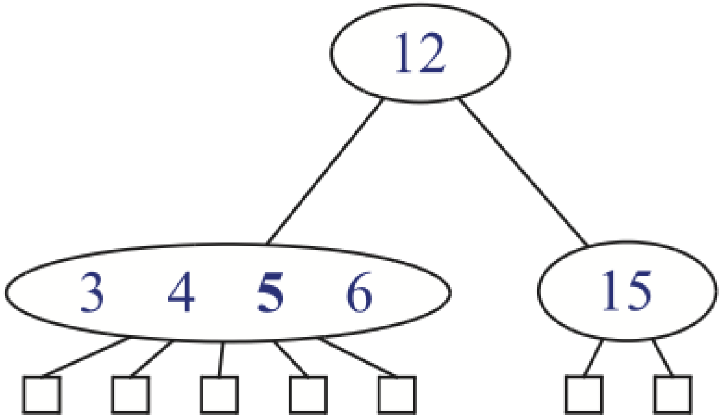
```
Node* treeSearch(int k, Node* v){
   if(isExternal(v))
      return v;

   for (int i = 0; i < (int)v->keys.size(); i++) {
      if (k == v->keys[i])
         return v;
      if (k < v->keys[i])
         return treeSearch(k, v->children[i]);
   }
   return treeSearch(k, v->children[v->keys.size()]);
}
```



Searching is a bit harder than a binary tree. Instead of only choosing left or right, we must choose between options 1, 2, 3 or 4

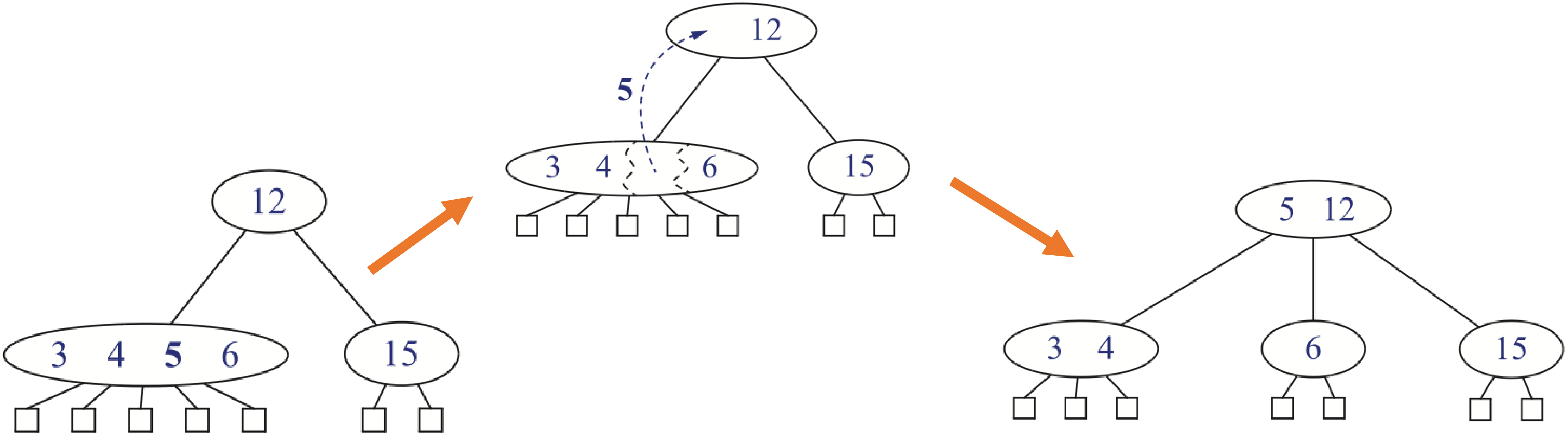
(2, 4) Tree Overflow

- When added keys to the tree, we add it to the lowest internal level of the tree
- This may exceed the max capacity of that node
- This is called overflow
- Example: adding 5



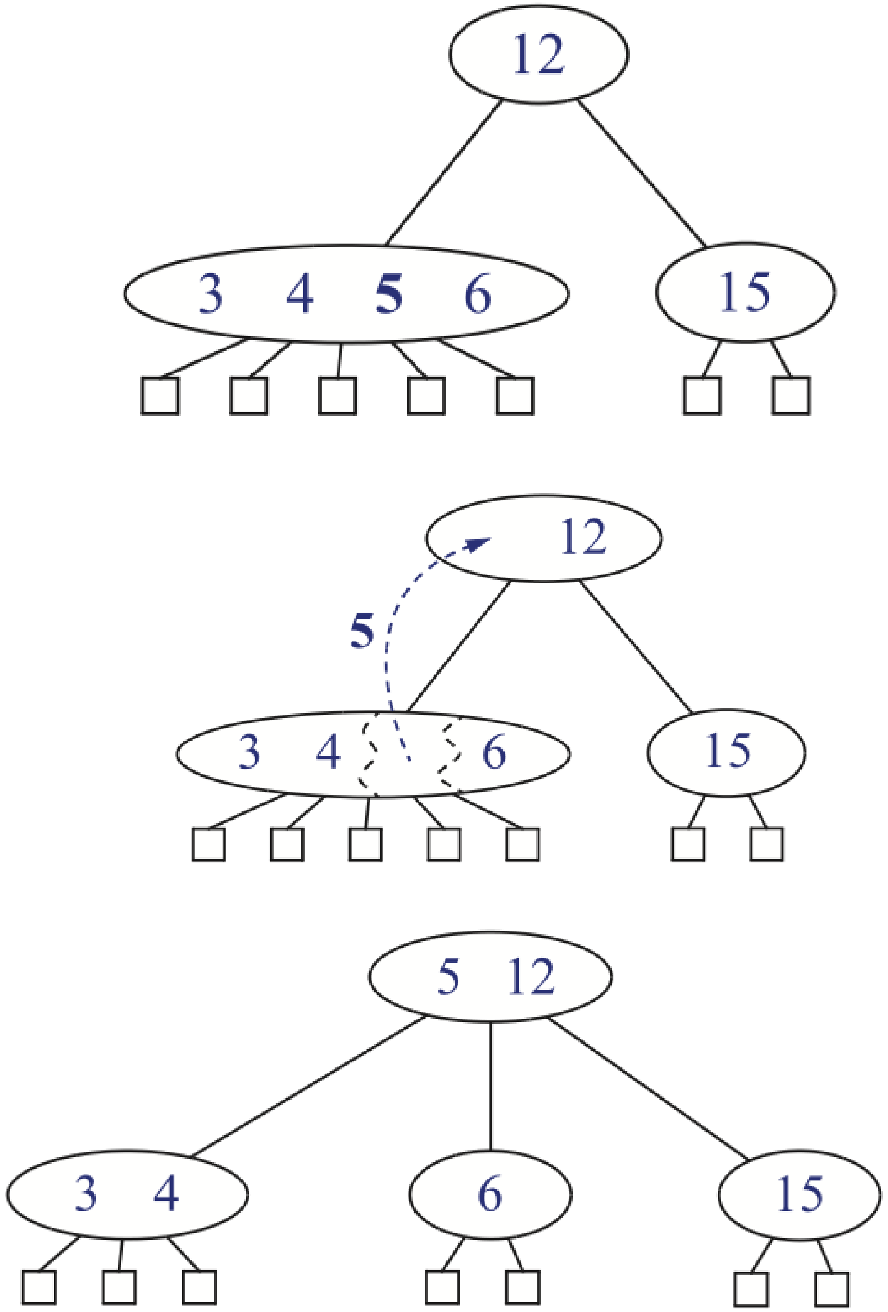
(2, 4) tree overflow

- When a node overflows, it must split.



Split

- Actions of a split:
  - Create a new node containing k_4 and children v_4 and v_5
  - Add the k_3 to the parent
  - Shrink the original node to contain only k_1 and k_2 with children v_1, v_2 and v_3



Exercise

- The goal today is to complete a (2, 4) tree by writing the spilt function

- Download the **TwoFour.cpp** file
  - Read the other functions to understand how they work, especially the `addKey` and `treeInsert` functions
  - Complete the code in **TwoFour.cpp**
    - Look for the comment: //YOUR CODE HERE
  - You will have to complete the functions :
    - `void splitChild(Node* parent, Node* child)` – make sure it will work when the child is internal and external

- After it is complete, it will run and create a (2,4) tree of 25 “random” items.