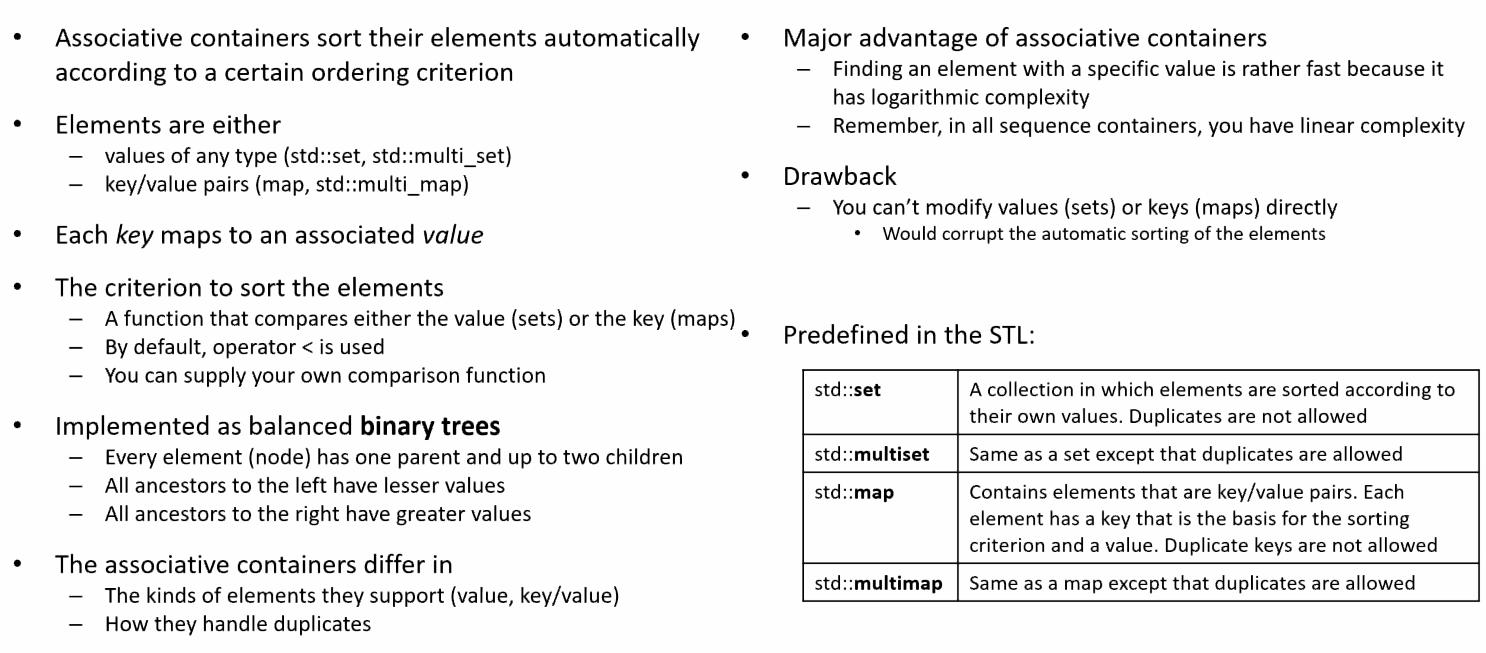
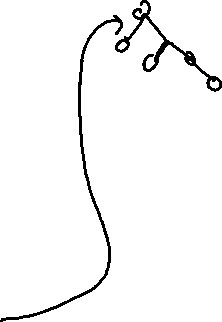
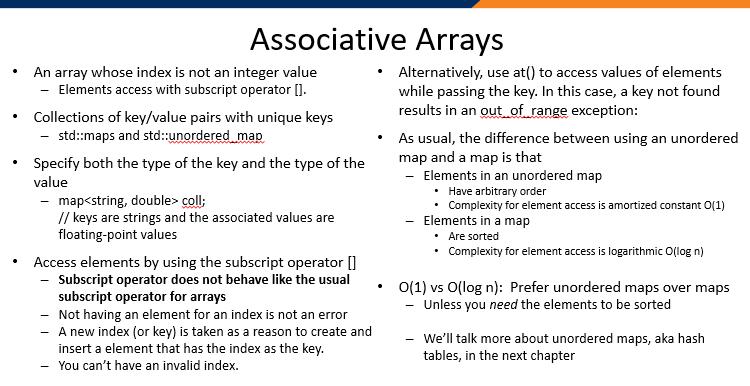
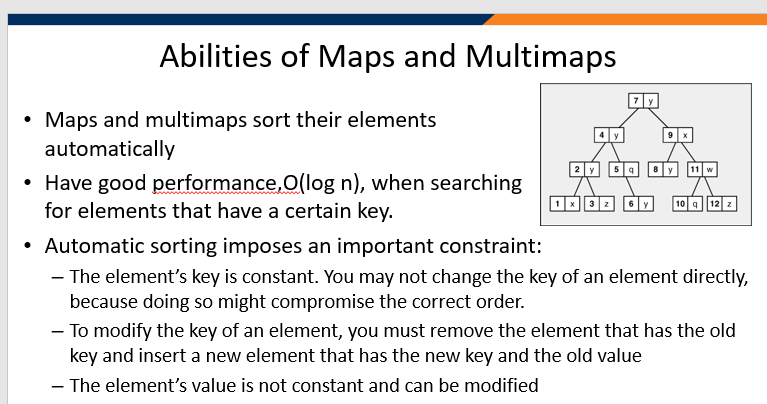
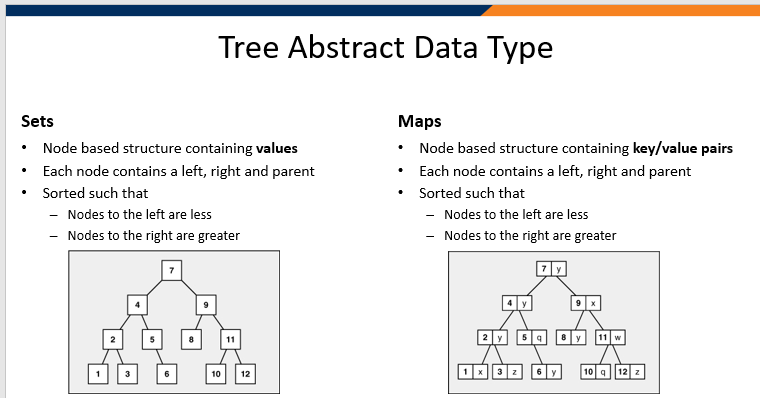
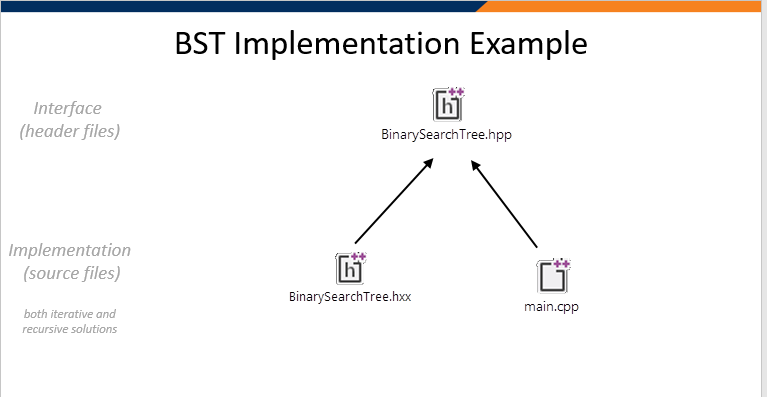
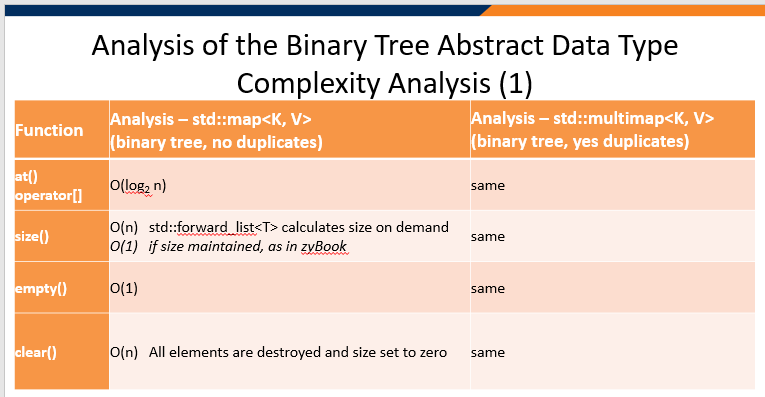
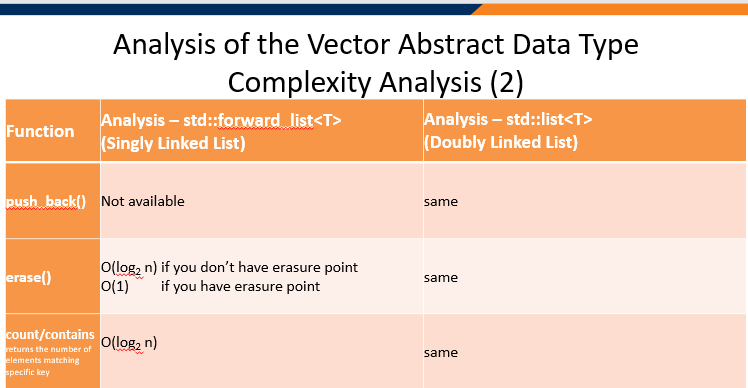
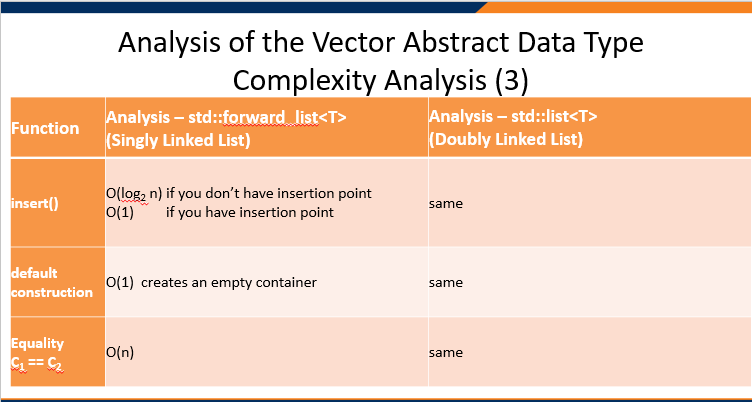
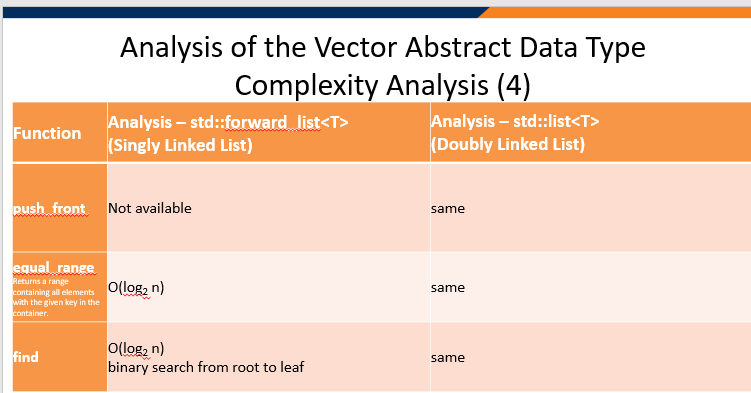
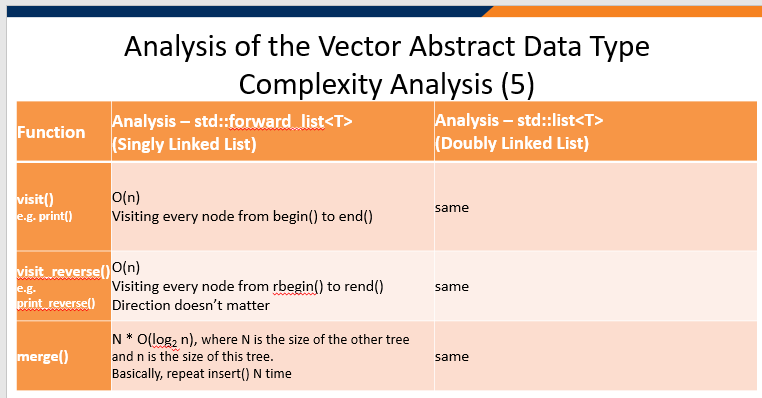
Lecture

CPSC 131  
10/26/20  
924 6951 7663

1. Associated Containers
   1. 



* 1. Keys = a pair of values as opposed to just one

1. Associative Arrays  
   
2. Abilities of Maps and Multimaps  
   
3. Tree Abstract Data Type  
   
4. BST Implementation Example  
   
5. Analysis of Binary Tree  
   
   1. Note, the at() operator[] is O(log2 n) regardless of duplicates. Why? Because the size is what the container maintains and at() becomes a matter of complexity as opposed to size.
   2. We have to go to every branch in the tree
6. Analysis of Vector Abstract Complexity  
   
7. Vector Part 2  
   
   1. Have to go through the process of the tree to get to where you need to be (again another reason why it’s log2n) if you don’t have an insertion point
   2. IF you traverse the trees the same way, the trees will be the same if the values are the same. They don’t need to have the same topology.
8. 
   1. When we say that n < log2n in terms of speed in this context, it’s because of the sheer amount of nodes we’re dealing with.
9.   
   1. Unlike a doubly-linked list in which we take the tail and hook it to the head, in order to merge two trees, we really have to visit a node in one tree and insert them in the right spot in the other.
   2. Merging two trees is no different from merging a list of multiple keys.
   3. There really is no advantage to merging two trees than inserting a 1000 times.