Time complexity Average:

Sorted List over Dynamic Vector

Function isEmpty(): O(1)

Function insert(): O(n)

Function getSize(): O(1)

Function deleteElement(): O(n/2)

Function getElement(): O(1)

Function getIterator(): O(1)

Sorted List over Binary Search Tree:

Function isEmpty(): O(1)

Function insert(): O(log n)

Function getSize(): O(1)

Function deleteElement(): O(log n)

Function getElement(): O(1)

Function getIterator(): O(1)

Pseudocode for 5 most important algorithms:

Subalgorithm SortedListOverDV<T>::insert(T element)

p <- dynamicVector->getSize()

dynamicVector->addLast(element)

while dynamicVector->getElement( p - 1) > element && p >0) then //we go backwards

//to check for perfect position

dynamicVector->update(p, dynamicVector->getElement( p -1));

p--;

endWhile

dynamicVector->update(p, element);

endSubalgorithm

Private Subalgorithm SortedListOverBST<K, E>::setOrderedNodes() //in-order traversing and saving

//the elements in a list

Node<K, E>\* node ->bst->GetRoot();

while not parentsStack.empty() or node != null

if node != null then

parrentsStack.push(node)

node = node->getLeft();

else

node = parrentsStack.top();

parentsStack.pop()

orderedNodes.push\_back(node) // vector where store it

node = node->getRight();

endWhile

endSubalgorithm

Subalgorithm Vector<T>::remove( position)

For i= position in size-1 then

elements[i] = elements[i+1]

eldFor

size --;

endSubalgorithm

Subalgorithm Vector<T>::insert( position, newElement)

If size < capacity then

for i=size, I > position; ii then

elements[i] = elements[i-1]

endFor

size++

else

reallocate()

endIf

endSubalgorithm

Subalgorithm BST<KEY, ELEMENT>::AddNode(KEY k, Node<KEY, ELEMENT>\* leaf)

If k <= leaf->GetKey())

If leaf->GetLeft() not null

AddNode(k ,e, left->getLeft())

Else

Node<KEY, ELEMENT>\* n = new Node<KEY, ELEMENT>()

n->SetKey(k)

n->SetElement(e)

n->setParrent(leaf)

leaf->SetLeft(n)

endIf

else

if leaf->getRight not null

AddNode(k ,e, leaf->GetRight())

Else

Node<KEY, ELEMENT>\* n = new Node<KEY, ELEMENT>()

n->SetKey(k)

n->SetElement(e)

n->setParrent(leaf)

leaf->setRight(n)

endIf

endSubalgorithm