1

Algorithm findMin(T)

Min<- undefined

If ( T.root ())

Min<-T.root().element()

Current<-T.root()

Prev<-current

While current

Prev=current

Current =T.leftChild(current)

Min = Prev.element()

Return min

Return false

2.

Algorithm inOrder(T)

List🡨new DLinkedList

If (T.root())

While (T.leftChild(T.root()))

Current 🡨 T.root()

Prev🡨 current

While current

Prev=current

Current=T.leftChild(current)

list.insertFirst(prev.element())

if T.rightChild(prev)

list.insertFirst(T.rightChild(prev).element())

T.remove(prev)

List.insertFirst(T.root().element())

While ( T.rightChild(T.root()))

Current🡨T.rightChild(T.root())

Prev🡨current

While (current)

Prev=current

Current=T.leftChild(current)

List.insertFirst(prev.element())

List.insertFirst(T.parent(prev).element() )

List.insertFirst(T.rightChild(T.parent(prev)))

Return list

1

Algorithm insertItem (value)

sizeOfTree🡨 sizeOfTree+1

newPos🡨 new HNode ( value , size())

heap[size]=newPos

upheap(size)

return newPos

2.

Algorithm removeMin()

If (empty) throw Error (invalid)

minELem🡸 minElem()

heap[1] =heap[size]

size()🡨size()-1

downheap(1)

return minElem