Design & Analysis of Algorithms

Monsoon Semester III 2020-21

**Lab – 3**

Date: **21 September 2020**

**Topics: Data Structure Binary Search Tree**

# AIM

To explore Binary search tree.

# EXPERIMENT

1. A binary search tree was created by reading the inputs from file ‘numbers.txt’. The inorder walk of the tree was displayed.

Properties of BST −

* The value of the key of the left sub-tree is less than the value of its parent (root) node's key.
* The value of the key of the right sub-tree is greater than or equal to the value of its parent (root) node's key.

1. The max and min depth of binary tree was calculated.

The **minimum depth** is the number of nodes along the shortest path from the root node down to the nearest leaf node.

**Maximum Depth** = Height of the tree

1. The lowest common ancestor of binary search tree was calculated. The input is taken from user after displaying the binary tree (in -order walk)

***Following is definition of LCA from***[***Wikipedia***](http://en.wikipedia.org/wiki/Lowest_common_ancestor)***:***  
Let T be a rooted tree. The lowest common ancestor between two nodes n1 and n2 is defined as the lowest node in T that has both n1 and n2 as descendants (where we allow a node to be a descendant of itself).

The LCA of n1 and n2 in T is the shared ancestor of n1 and n2 that is located farthest from the root. Computation of lowest common ancestors may be useful, for instance, as part of a procedure for determining the distance between pairs of nodes in a tree: the distance from n1 to n2 can be computed as the distance from the root to n1, plus the distance from the root to n2, minus twice the distance from the root to their lowest common ancestor.

1. A heap sort was implement by varying the number of input elements (10, 100, 500, 1000, 10000). The time and memory taken for each of input case was computed

**Approach**:

  Heap sort happens in two phases. In the first phase, the array

  is transformed into a heap. A heap is a binary tree where

  1) each node is greater than each of its children

  2) the tree is perfectly balanced

  3) all leaves are in the leftmost position available.

  In phase two the heap is continuously reduced to a sorted array:

  1) while the heap is not empty

  - remove the top of the head into an array

  - fix the heap.

**MoveDown**:

  The movedown method checks and verifies that the structure is a heap.

**Technical Details**:

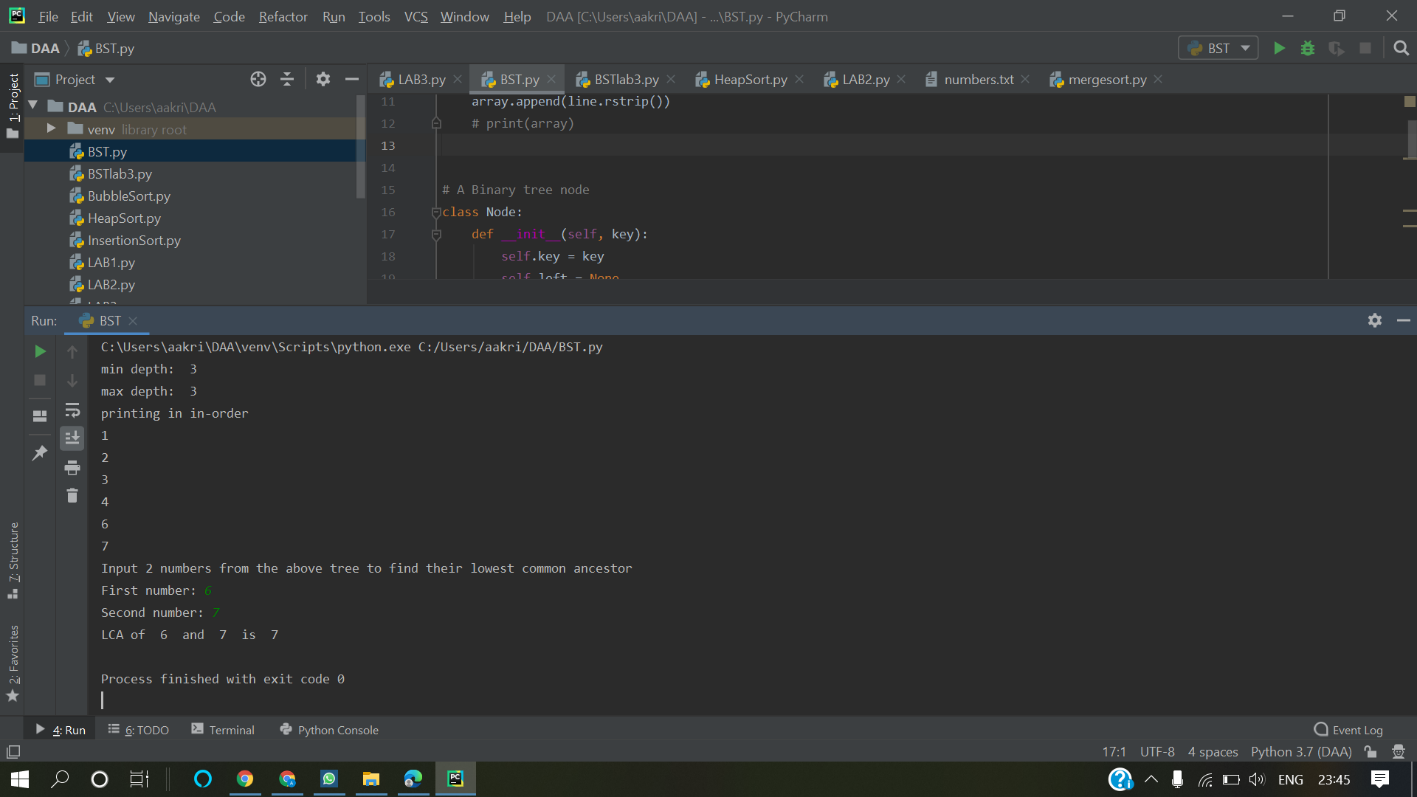
  A heap is based on an array just as a hashmap is based on an array. For a heap, the children of an element n are at index 2n+1 for the left child and 2n+2 for the right child.

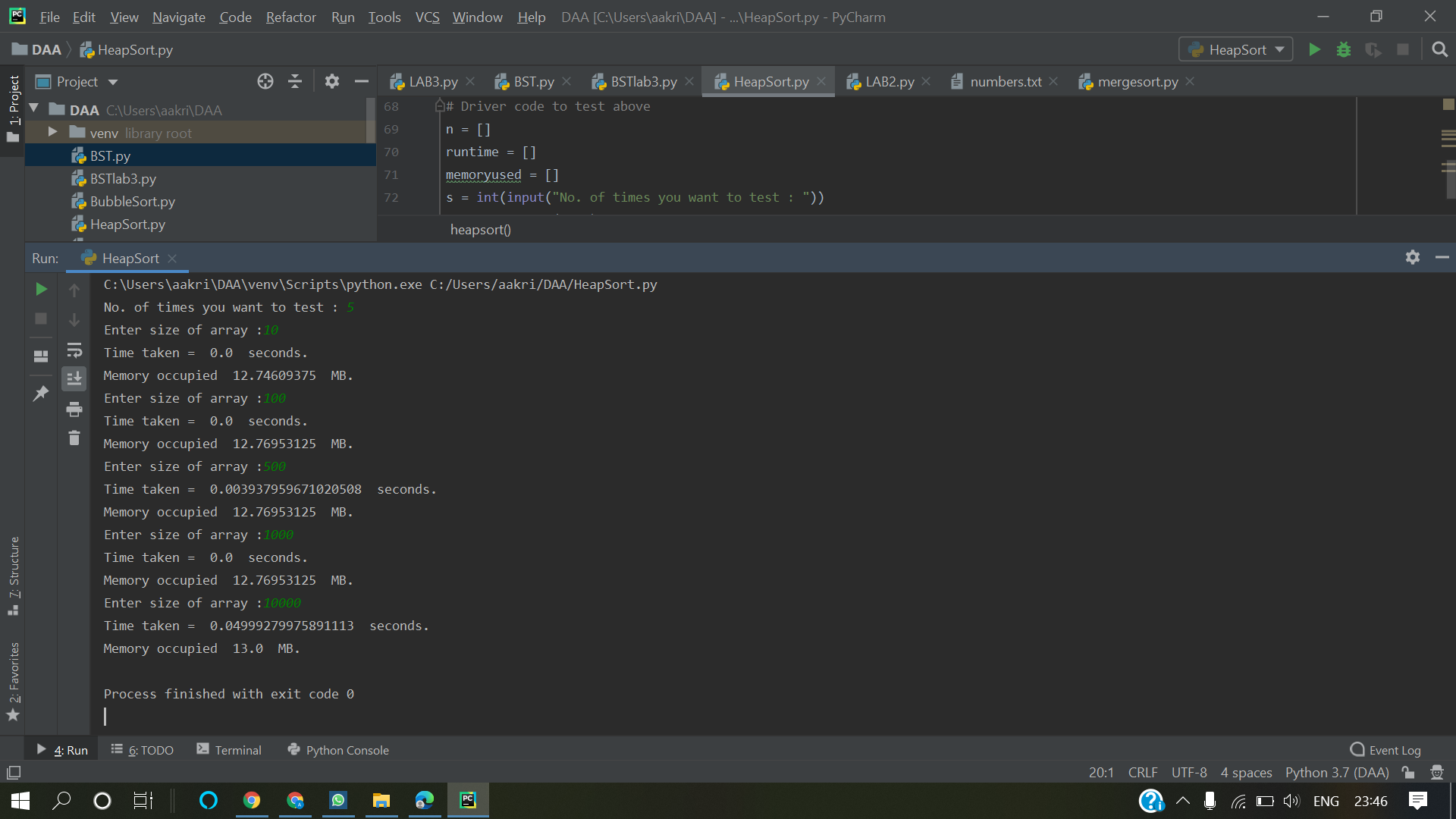
  The movedown function checks that an element is greater than its children. If not the values of element and child are swapped. The function continues to check and swap until the element is at a position where it is greater than its children.

**Time Complexity** of Solution:

Best O(nlog(n)); Average O(nlog(n)); Worst O(nlog(n)).

# OBSERVATIONS





## CONCLUSION

Binary Search Tree and Heap Sort was explored.