PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
COMPUTER LABORATORY - IV

Assignment No - B6

1 Aim

Implement OBST Tree search using HPC task sub-division. Merge the results to get final result.

2 Objective

- To understand concept of OBST.
- To effectively use multi-core or distributed, concurrent/Parallel environments.
- To develop problem solving abilities using Mathematical Modeling.
- To develop time and space efficient algorithms.

3 Software Requirements

- Linux or Windows Operating System
- Python

4 Mathematical Model

Let S be the system,
S : (s, e, I, O, Fm, Ft, St)

Where,
s: Start State , i.e. taking input e: End State i.e. search result I: Set of inputs
I: size of array, array elements O:
Set of outputs
O: Sorted array, position of element
Fm: Main function or algorithm that gives specific Output i.e. main()
Ft: Failure State. i.e. Getting Exceptions as Output after inserting characters, string or float values as Input.
St : Success State. i.e. Getting the Expected Element and its position in array as output.

5 Theory

5.1 OSBT:

An optimal binary search tree is a binary search tree for which the nodes are arranged on levels such that the tree cost is minimum. For the purpose of a better presentation of optimal binary search trees, we will consider extended binary search trees, which have the keys stored at their internal nodes. Suppose n keys k1, k2, , k n are stored at the internal nodes of a binary search tree. It is assumed that the keys are given in sorted order, so that k1¡ k2 ¡ ¡ kn. An extended binary search tree is obtained from the binary search tree by adding successor nodes to each of its terminal nodes as indicated in the following figure by squares.

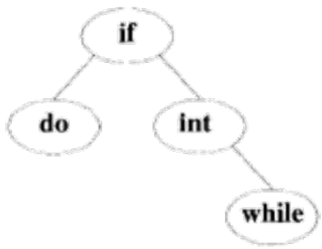
$$c(i,j) = \min_{i<k\leq j}\{c(i,k-1)+c(k,j)+p(k)+w(i,k-1)+w(k,j)\}$$

$$c(i,j) = \min_{i<k\leq j}\{c(i,k-1)+c(k,j)\}+w(i,j)$$

Example 5.18 Let $n = 4$ and $(a_1, a_2, a_3, a_4) =$ (**do, if, int, while**). Let $p(1:4) = (3, 3, 1, 1)$ and $q(0:4) = (2, 3, 1, 1, 1)$. The p's and q's have been multiplied by 16 for convenience. Initially, we have $w(i,i) = q(i), c(i,i) = 0$ and $r(i,i) = 0, 0 \le i \le 4$. Using Equation 5.12 and the observation $w(i,j) = p(j) + q(j) + w(i, j-1)$, we get

$$
\begin{aligned}
w(0,1) &= p(1) + q(1) + w(0,0) = 8 \\
c(0,1) &= w(0,1) + \min\{c(0,0) + c(1,1)\} \quad = \quad 8 \\
r(0,1) &= 1 \\
w(1,2) &= p(2) + q(2) + w(1,1) \quad = \quad 7 \\
c(1,2) &= w(1,2) + \min\ \{c(1,1) + c(2,2)\} \quad = \quad 7 \\
r(0,2) &= 2 \\
w(2,3) &= p(3) + q(3) + w(2,2) \quad = \quad 3 \\
c(2,3) &= w(2,3) + \min\ \{c(2,2) + c(3,3)\} \quad = \quad 3 \\
r(2,3) &= 3 \\
w(3,4) &= p(4) + q(4) + w(3,3) \quad = \quad 3 \\
c(3,4) &= w(3,4) + \min\ \{c(3,3) + c(4,4)\} \quad = \quad 3 \\
r(3,4) &= 4
\end{aligned}
$$

	0	1	2	3	4
0	$w_{00} = 2$ $c_{00} = 0$ $r_{00} = 0$	$w_{11} = 3$ $c_{11} = 0$ $r_{11} = 0$	$w_{22} = 1$ $c_{22} = 0$ $r_{22} = 0$	$w_{33} = 1$ $c_{33} = 0$ $r_{33} = 0$	$w_{44} = 1$ $c_{44} = 0$ $r_{44} = 0$
1	$w_{01} = 8$ $c_{01} = 8$ $r_{01} = 1$	$w_{12} = 7$ $c_{12} = 7$ $r_{12} = 2$	$w_{23} = 3$ $c_{23} = 3$ $r_{23} = 3$	$w_{34} = 3$ $c_{34} = 3$ $r_{34} = 4$	
2	$w_{02} = 12$ $c_{02} = 19$ $r_{02} = 1$	$w_{13} = 9$ $c_{13} = 12$ $r_{13} = 2$	$w_{24} = 5$ $c_{24} = 8$ $r_{24} = 3$		
3	$w_{03} = 14$ $c_{03} = 25$ $r_{03} = 2$	$w_{14} = 11$ $c_{14} = 19$ $r_{14} = 2$			
4	$w_{04} = 16$ $c_{04} = 32$ $r_{04} = 2$				



5.2 Advantage:

The major advantage of binary search trees over other data structures is that the related sorting algorithms and search algorithms such as in-order traversal can be very efficient, they are also easy to code.

5.3 Disadvantage:

- The shape of the binary search tree totally depends on the order of insertions and deletions, and can become degenerate.
- When inserting or searching for an element in a binary search tree, the key of each visited node has to be compared with the key of the element to be inserted or found.
- The keys in the binary search tree may be long and the run time may increase.
- After a long intermixed sequence of random insertion and deletion, the expected height of the tree approaches square root of the number of keys, N, which grows much faster than log N.

5.4 Analysis:

The optimal binary search tree has a time complexity of O(n$\hat{3}$). Its space efficiency is only O(n$\hat{2}$). It can possibly be lowered to complexity O(n$\hat{2}$) with smarter recursive functions and smaller ranges of values.

6 Algorithm

Suppose n keys k1, k2, , k n are stored at the internal nodes of a binary search tree. It is assumed that the keys are given in sorted order, so that k1¡ k2 ¡ ¡ kn. An extended binary search tree is obtained from the binary search tree by adding successor nodes to each of its terminal nodes as indicated in the following figure by squares.

$$c(i,j) = \min_{i<k\leq j}\{c(i,k-1)+c(k,j)+p(k)+w(i,k-1)+w(k,j)\}$$

$$c(i,j) = \min_{i<k\leq j}\{c(i,k-1)+c(k,j)\}+w(i,j)$$

7 Testing

7.1 Positive Testing

Sr. No.	Test Condition	Steps to be executed	Expected Result	Actual Result
1.	Enter number of keys.P.Q	Press Enter	Display Binary Tree	Same as Expected

7.2 Negative Testing

Sr. No.	Test Condition	Steps to be executed	Expected Result	Actual Result
1.	Enter number of keys.P.Q not in particular format	Press Enter	Tree is not display	Binary tree

8 Conclusion

From this assignment we have studied concept of OBST and developed time and space efficient algorithm using multicore , concurrent environment.

Roll No.	Name of Student	Date of Performance	Date of Submission
302	Abhinav Bakshi	8/3/16	15/3/16

9 Plagarism Report

Result:

61% Unique

\documentclass[11pt]{article} \usepackage{graphicx} \begin1{document}	- **Unique**
Div: C\\ \end{flushleft}• \begin{center} \begin{Large} \textsc{Assignment	- **Unique**
\end{flushleft}• Using Divide and Conquer Strategies design	- **Unique**
\textbf{Description:} \end{flushleft}• Quick Sort, as the	- **Unique**
not stable search, but it is very fast and requires very	- **Unique**
and Conquer(also called partition-exchange sort). This algorithm	- **Unique**
\item Elements less than the Pivot element \item Pivot element	- **Unique**
In the list of elements, mentioned in below example, we	- **Unique**
will be changed like this. [6 8 17 14 25 63 37 52]\\ Hence	- **Unique**
with all the elements smaller to it on its left and all	- **Plagiarized**
and [63 37 52] are considered as two separate lists, and	- **Unique**
the complete list is sorted.\\ Quicksort is a fast sorting	- **Unique**
but widely applied in practice. On the average, it has O(n	- **Plagiarized**
big data volumes.\\ \begin{flushleft} \textbf{Algorithm}	- **Unique**
strategy is used in quicksort. Below the recursion step	- **Plagiarized**
a pivot value. We take the value of the middle element as	- **Plagiarized**