Department of Computer Technology and Information Systems

CTIS264 – Computer Algorithms

Spring 2019 - 2020

Lab Guide 7 - Week 10

Instructor : Füsun YÜRÜTEN

Assistant : Leyla SEZER

OBJECTIVE:

- **Empirical Analysis of the Quick-Sort Algorithm**
- **Empirical Analysis of the Merge-Sort Algorithm**
- **Binary-Tree Creation and Post-Order, Pre-Order, In-Order Traversals**

Q1. Write a Python program that makes the empirical analysis of the quick-sort algorithm by using following algorithm, which uses Hoare's Partitioning Algorithm.

ALGORITHM Quicksort(A[l..r])
//Sorts a subarray by quicksort
//Input: Subarray of array A[0..n − 1], defined by its left and right
// indices l and r
//Output: Subarray A[l..r] sorted in nondecreasing order if l < r
s ←Partition(A[l..r]) //s is a split position
Quicksort(A[l..s − 1])
Quicksort(A[s + 1..r])

Algorithm *Partition*$(A[l..r])$
//Partitions a subarray by using its first element as a pivot
//Input: A subarray $A[l..r]$ of $A[0..n-1]$, defined by its left and right
// indices l and r $(l < r)$
//Output: A partition of $A[l..r]$, with the split position returned as
// this function's value
$p \leftarrow A[l]$
$i \leftarrow l$; $j \leftarrow r+1$
repeat
  repeat $i \leftarrow i+1$ **until** $A[i] \geq p$
  repeat $j \leftarrow j-1$ **until** $A[j] \cdot p$
  swap$(A[i], A[j])$
until $i \geq j$
swap$(A[i], A[j])$ //undo last swap when $i \geq j$
swap$(A[l], A[j])$
return j

- Program generates a random array. Size of the array is 10000 and the numbers between 0-100.

Output:

```
None
 0.0156233 seconds
```

Q2. Write a Python program that makes the empirical analysis of the merge-sort algorithm by using following algorithm;

ALGORITHM *Mergesort*$(A[0..n-1])$

```
//Sorts array A[0..n − 1] by recursive mergesort
//Input: An array A[0..n − 1] of orderable elements
//Output: Array A[0..n − 1] sorted in nondecreasing order
if n > 1
    copy A[0..⌊n/2⌋ − 1] to B[0..⌊n/2⌋ − 1]
    copy A[⌊n/2⌋..n − 1] to C[0..⌈n/2⌉ − 1]
    Mergesort(B[0..⌊n/2⌋ − 1])
    Mergesort(C[0..⌈n/2⌉ − 1])
    Merge(B, C, A)
```

ALGORITHM *Merge*$(B[0..p-1], C[0..q-1], A[0..p+q-1])$

```
//Merges two sorted arrays into one sorted array
//Input: Arrays B[0..p − 1] and C[0..q − 1] both sorted
//Output: Sorted array A[0..p + q − 1] of the elements of B and C
i ← 0; j ← 0; k ← 0
while i < p and j < q do
    if B[i] ≤ C[j]
        A[k] ← B[i]; i ← i + 1
    else A[k] ← C[j]; j ← j + 1
    k ← k + 1
if i = p
    copy C[j..q − 1] to A[k..p + q − 1]
else copy B[i..p − 1] to A[k..p + q − 1]
```

- Program generates a random array using numpy module. Size of the array is 10000 and and numbers will be between 1-1000.

Output:

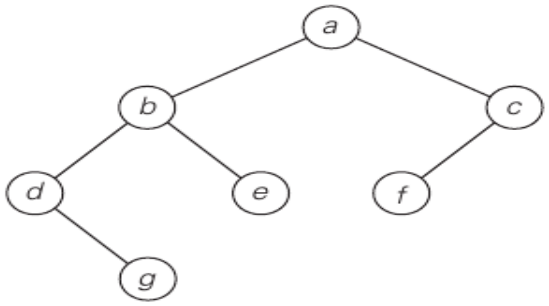
```
0.12713909149169922 seconds
[  1   1   1 ... 396 396 981]
```

Q3. Write a OOP-Python program that gets 10 numbers from the user, to construct a sorted binary-tree. Then program displays the **post-order**, **pre-order** and **in-order** traversals of the given tree.

Please write down :

Insert_node(new_node_value) recursive function,
get_right_child()
get-left_child()
set_root_val()
get_root_val()
preorderTrav()
inorderTrav()
postorderTrav() functions

Check the given examples.



preorder: *a, b, d, g, e, c, f*
inorder: *d, g, b, e, a, f, c*
postorder: *g, d, e, b, f, c, a*

Output:

```
The binary tree will be created with your numbers.
Please enter 10 numbers to create sorted binary tree
Enter a number:35
root: None
Enter a number:60
root: 35
right child <__main__.BinaryTree object at 0x03E09410>
Enter a number:90
root: 35
right child <__main__.BinaryTree object at 0x03E09A70>
right tree <__main__.BinaryTree object at 0x03E09410>
Enter a number:5
root: 35
left child <__main__.BinaryTree object at 0x03E09E10>
Enter a number:4
root: 35
left child <__main__.BinaryTree object at 0x03E09E30>
left tree <__main__.BinaryTree object at 0x03E09E10>
Enter a number:40
root: 35
left child <__main__.BinaryTree object at 0x03E09E50>
right tree <__main__.BinaryTree object at 0x03E09410>
Enter a number:3
root: 35
left child <__main__.BinaryTree object at 0x03E09E70>
left tree <__main__.BinaryTree object at 0x03E09E30>
left tree <__main__.BinaryTree object at 0x03E09E10>
Enter a number:15
root: 35
right child <__main__.BinaryTree object at 0x03E09E90>
left tree <__main__.BinaryTree object at 0x03E09E10>
Enter a number:37
root: 35
left child <__main__.BinaryTree object at 0x03E09EB0>
left tree <__main__.BinaryTree object at 0x03E09E50>
right tree <__main__.BinaryTree object at 0x03E09410>
...***...
Pre-Order
35
5
4
3
15
60
40
37
90
```

```
...***...
Post-Order
3
4
15
5
37
40
90
60
35
...***...
In-Order
3
4
5
15
35
37
40
60
90
>>>
```