Homework 2

Question 1

- No, it is incorrect
- It does not have a base case to check nodes without left and right node.
- Correction:

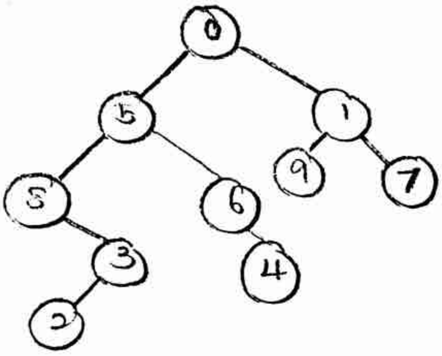
  // Input: A binary tree T
  // Output: The number of leaves in T

  if T = ∅ return 0
  if T_{left} = null && T_{right} = null
  return 1
  else return LeafCounter (T_{left}) + LeafCounter (T_{right})

Question 2

(a) inorder : 8, 2, 3, 5, 6, 4, 0 (root), 9, 1, 7 (L, root, R)

Preorder : 0, 5, 8, 3, 2, 6, 4, 1, 9, 7 (Root, L, R)

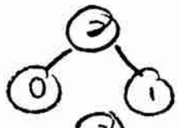


(b) when n = 3

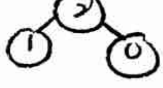
permutations: {0,1,2}, {0,2,1}, {2,0,1}, {2,1,0}, {1,2,0}, {1,0,2}

Eg of 2 permutations:

inorder : {0,2,1} ⇒ L, root, R ⇒



postorder : {1,0,2} ⇒ L, R, root ⇒



(c) ALGORITHM BinaryTree($[a_0, a_1, \_\_ a_n]$, $[b_0, b_1, \ldots b_n]$)

// Constructs a binary tree for which two given lists of n labels 0, 1, ... n-1 are generated by the inorder and postorder traversals of the tree

// Input : Two lists (inorder and postorder)

// Output : Print binary tree

if $b_n \neq a_k$ then
  return -1
else
  print a_k

Binary Tree ($[a_0, a_1, \ldots a_{k-1}]$, $[b_0, b_1, \ldots b_{k-1}]$) // left subtree

Binary Tree ($[a_{k+1}, \ldots a_n]$, $[b_k, b_{k+1}, \ldots b_{n-2}]$) // right subtree

Question 3

s = no of searches

mergesort ⇒ $\theta(n \log n)$
sequential search ⇒ $\theta(\frac{n}{2})$
binary search ⇒ $\theta(\log n)$

} Average case

$n \log n + s \log n \leq s \frac{n}{2}$

$s \geq \dfrac{n \log n}{\frac{n}{2} - \log n}$

when $n = 10$

$s \geq \dfrac{10 \log 10}{\frac{10}{2} - \log 10}$

$= \dfrac{10\left(\frac{\log 10}{\log 2}\right)}{5 - \left(\frac{\log 10}{\log 2}\right)}$

$= \dfrac{33.22}{1.678}$

$= 19.796$

≈ 20

when $n = 10^5$

$s \geq \dfrac{10^5 \log 10^5}{\frac{10^5}{2} - \log 10^5}$

$= \dfrac{10^5 (5) \left(\frac{\log 10}{\log 2}\right)}{\frac{10^5}{2} - 5\left(\frac{\log 10}{\log 2}\right)}$

$= \dfrac{1660964.047}{49983.39036}$

$= 33.23$

≈ 34

Question 4

a) ALGORITHM findDist ([$a_0, a_1, \dots a_n$])

// Find the distance between two closest number in an array of n numbers

// Input: An unsorted array ([$a_0, a_1, \dots a_n$])

// Output: Distance between two closest numbers in the array

```
quicksort ([a0, a1, ... an])

let min = an
for i ← 1 to n-1 do
      if |ai - ai-1| < min
          then min = |ai - ai-1|
      end if
end for
```

b) The worst case for the algorithm above is $O(n \lg n)$ which is way more efficient compared to $O(n^2)$ from brute-force algorithm.

Question 5

ALGORITHM numberPlacement (list of int [$a_0, \dots a_n$], sequence of n boxes with preset inequality [$b_0, \dots b_m$])

// Place the numbers into the boxes to satisfy the sequence of inequality

// Input: List of integers [$a_0, \dots a_n$]
sequence of box [$b_0, \dots b_m$]

// Output: Sequence of integers [$b_0, \dots b_m$] which satisfy the inequalities.

```
quicksort ([a0, a1, ... an])
i ← 0
j ← n
k ← 1
while k < m
   if bk = '<' then
       bk-1 = a[i]
       i++
   else if bk = '>' then
       bk-1 = a[j]
       j--
   k++
bm = a[i]

return sequence of box
```

Question 6

- In inner loop, the algorithm performs $\sum_{i=1}^{n} i$ times of multiplication for each outer loop
- Then in the outer loop, it performs $n+1$ times of multiplication as well.

- Total number of multiplications $= \frac{n(n+1)}{2} + (n+1)$

- In outer loop, it performs $n+1$ times of addition

- Total number of addition $= n+1$