COMP2119A Introduction to Data Structures and Algorithms - Quiz 2

University Number: ______________________ **Time Limit: 1 hour 50 minutes**

Rules: You can get at most 100 points if attempting all problems. Please make your answers precise and concise.

Course Outcomes

- **[O1]. Mathematics foundation**
- **[O2]. Basic data structures**
- **[O3]. Problem solving**
- **[O4]. Implementation**

1. **[O1, O2]** (30 pts) Fill in the blanks.

   I (18 pts) You need to state your proof.

   (a) In a complete k-ary tree, every node has exactly k children or no child. The internal node has exact k children. The number of leaves in such a tree with n internal nodes is ______________________. (Express it in terms of n and k.)
   Solution: $n(k-1)+1$.
   Assume the number of leaves is x. On the one hand, there are $x+n$ nodes in total, so there are $x+n-1$ edges. On the other hand, the number of edges is nk. $nk = x+n-1, x = n(k-1)+1$.

   (b) There exists an input array of length n that forces randomized quicksort to run in $\Theta(n^2)$ time in expectation. (True or false)

   Solution: False. For any input, the expected running time of quicksort is $O(n \log n)$, where the expectation is taken over the random choices made by quicksort, independent of the choice of the input.

   (c) There exists a comparison-based sorting algorithm for 5 numbers that uses at most 6 comparisons in the worst case. (True or false)
   Solution: False. The number of leaves of a decision tree which sorts 5 numbers is 5! and the height of the tree is at least $\lg(5!)$. Since $6 < \log(5!) < 7$. Thus at least 7 comparisons are required.

   II (12 pts) No proof is needed.

   (a) A binary search tree T produces sequence of keys "0, 16, 13, 22, 19, 36, 75, 88, 54, 23", after a post-order traversal of T. Please write down the sequence if we do an pre-order traversal of T:

Solution: 23, 19, 13, 0, 16, 22, 54, 36, 88, 75

(b) A binary search tree contains the values 1, 2, 3, 4, 5, 6, 7, 8. The tree is traversed in pre-order and the values are printed out. Which of the following sequences might be a valid output?
A. 53124786
B. 53126487
C. 53241678
D. 53124768
Solution: D

(c) The complexity of which of the following sorting algorithms remains to be the same in its best, average and worst case, counting sort, quick sort, insertion sort or selection sort? (You may choose more than one sorting algorithm.)
____________________.

Solution: Counting sort and selection sort.

(d) If the array is already sorted, which of comparison-based sorting algorithms will exhibit the best performance, bubble sort, insertion sort, selection sort, merge sort, or quick sort? ____________________.
Solution: Insertion sort.

University Number: ____________________

2. **[O2, O3]** (12 pts) Height of a tree is the length of the path from root of that tree to its farthest node. In a binary tree, for every node the difference between the number of nodes in the left and right subtrees is at most 2. Assume the minimum number of nodes in such a tree with height h is a_h.

   (a) (4 pts) Please calculate a_1, a_2.
   Solution: $a_1 = 2, a_2 = 3$.

   (b) (8 pts) What is a_h? Express it in terms of h, and prove it's tight. (By showing such a tree exists.)
   Solution: $a_h = 2^{h-1} + 1$.
   Consider root node, assume the height of left subtree is $h-1$, then, the number of nodes in right subtree is at least $a_{h-1} - 2, h > 2$.
   $a_h \geq a_{h-1} + a_{h-1} - 2 + 1$.
   $a_h - 1 \geq 2(a_{h-1} - 1), a_h - 1 \geq 2^{h-2}(a_2 - 1)$. We have $a_1 = 2, a_2 = 3$.
   Therefore, $a_h \geq 2^{h-1} + 1, h > 0$.

   Proof:
   We contruct a tree with height h and the number of nodes a_h as follows:
   First, generate a tree with height 1: it contains two nodes and one edge.

   i. Generate a new root node r.
   ii. Make current tree r's left subtree.
   iii. Generate a new tree rt by removing two nodes carefully from r's left subtree without breaking the difference property. (If the number of left subtree is more than 2, assume the height of r's left subtree is d, remove the two nodes with depth d and $d-1$; Otherwise, rt=NULL.)
   iv. Make rt r's right subtree. A new tree is generated.

   Repeat the above process $h-1$ times until the height of current tree reaches h.
   Therefore, $a_h = 2^{h-1} + 1, h > 0$.

University Number: ____________________

3. **[O3, O4]** (13 pts) You are given a binary search tree with n nodes of height h, together with a range $[a, b]$. The keys in the tree are distinct integers. Please write a program that outputs all keys in the tree that are **at least** a and **at most** b. Suppose there are t such keys, you program needs to run in $O(h + t)$ time.

   function TRAVERSE($root$,a,b)
   /* Write your algorithm here */

   Solution:

   function TRAVERSE($root$,a,b)
       if $root$ is NULL **then**
           RETURN
       if $a \leq root.key \leq b$ **then**
           PRINT $root.key$
       if $root.key \geq a$ **then**
           TRAVERSE($root.left$,a,b)
       if $root.key \leq b$ **then**
           TRAVERSE($root.right$,a,b)

University Number: ____________________

4. **[O1]** (15 pts) There are $n!$ different arrays of size n containing $1, 2, \ldots, n$. If we do bubble sort on all these arrays, what is the average number of swap? (For example, when $n = 2$, there are two arrays: [1,2] and [2,1]. The total number of swap is 1, so the average is 0.5.)

```
void bubble_sort(int *array, int n)
{
    int i, j;
    for (i = 0; i < n - 1; ++i)
    {
        for (j = 0; j < n - 1; ++j)
        {
            if (array[j] > array[j + 1])
            {
                swap(array[j],array[j+1]);
            }
        }
    }
}
```

Solution:
We consider inverse pair: if $i, j \in \{1, 2, \ldots, n\}, i < j$, and $array[i] > array[j]$, we call i and j an inverse pair.
For any array A, assume there are x inverse pair(s). Reverse A to B, such that $A[i] = B[n-1-i]$. The number of inverse pair(s) in B is $\frac{n(n-1)}{2} - x$.
Therefore, the average is $\frac{n(n-1)}{4}$.

University Number: ____________________

5. **Close Numbers [O1, O3]** (30 pts, 3 parts)
Consider a set S of $n \geq 2$ distinct numbers. For simplicity, assume that $n = 2^k + 1$ for some $k > 0$. Call a pair of distinct numbers $x, y \in S$ close in S if

$$|x - y| \leq \frac{1}{n-1}(\max_{z \in S} z - \min_{z \in S} z),$$

i.e. if the distance between x and y is at most the average distance between consecutive numbers in the sorted order.

(a)(10 pts) Explain why every set S of $n \geq 2$ distinct numbers contains a close pair of numbers.

Solution: Without loss of generality, assume $S = \{z_1, z_2, \ldots z_n\}$, with $z_i \leq z_{i+1}$. The average distance between two consecutive numbers z_i and z_{i+1} is

$$\frac{1}{n-1} \sum_{i=1}^{n-1} (z_{i+1} - z_i) = \frac{1}{n-1}(z_n - z_1)$$

There exists at least one pair of consecutive numbers x and y whose distance between them is less than or equal to the average. The result then follows from the definition of the *close* pair.

(b)(10 pts) Suppose that we partition S around a pivot element $p \in S$, organizing the result into two subsets of $S : S_1 = \{x \in S | x \leq p\}$ and $S_2 = \{x \in S | x \geq p\}$. Prove that either
1. every pair $x, y \in S_1$ of numbers that is close in S_1 is also close in S, or
2. every pair $x, y \in S_2$ of numbers that is close in S_2 is also close in S.
Show how to determine, in $O(n)$ time, a value $k \in \{1, 2\}$ such that every pair $x, y \in S_k$ of numbers that is close in S_k is also close in S.
Notice: elements in S are distinct.

Solution: Without loss of generality, assume that the elements in S_k are in sorted order. For $k = 1, 2$, let a_k be the average distance between two consecutive numbers in S_k , and let n_k the number of elements in S_k . Using the result from Part (a), we have

$$a_1 = \frac{1}{n_1 - 1}(\max_{z \in S_1} z - \min_{z \in S_1} z) = \frac{1}{n_1 - 1}(p - \min_{z \in S_1} z)$$

and

$$a_2 = \frac{1}{n_1 - 1}(\max_{z \in S_2} z - \min_{z \in S_2} z) = \frac{1}{n_2 - 1}(\max_{z \in S_2} z - p)$$

The average distance a between two consecutive numbers in S in sorted order is then given by

$$a = \frac{1}{n-1}(\max_{z \in S} z - \min_{z \in S} z) = \frac{1}{n-1}(p - \min_{z \in S} z) + \frac{1}{n-1}(\max_{z \in S} z - p) = \frac{n_1 - 1}{n-1} a_1 + \frac{n_2 - 1}{n-1} a_2$$

Note that $n_1 + n_2 = n + 1$, because p is included in both S_1 and S_2 . So, a is a weighted average of a_1 and a_2,

$$a = (1 - \alpha)a_1 + \alpha a_2$$

, where $\alpha = \frac{n_2 - 1}{n-1}$. Suppose that $a_1 \leq a_2$. If x and y are a close pair in S_1 , then

$$|x - y| \leq a_1 = (1 - \alpha)a_1 + \alpha a_1 \leq (1 - \alpha)a_1 + \alpha a_2 = a.$$

This implies that every close pair in S_1 is also a close pair in S. Similarly, if $a_2 \leq a_1$, then every close pair in S_2 is a close pair in S. The average distance a_k can be computed in $O(n)$ time, by searching for the minimum or the maximum number in S_k . Therefore, the subset S_k with the specified property can be computed in $O(n)$ time.

University Number: ____________________

(c) (10 pts) Describe an expected $O(n)$-time algorithm to find a close pair of numbers in S. Explain briefly why your algorithm is correct, and analyse its running time. (Hint: 1. Use divide and conquer; 2. Might be a randomized algorithm, like quick sort.)

Solution: The idea is to partition S recursively until we find a close pair.
1. Determine the median of S and use it to partition S into S_1 and S_2 .
2. Use the result from Part (b) to determine the set S_k that contains a close pair of S.
3. Recursion on S_k until S_k contains 2 elements.
Since each recursive step reduces the cardinality of the set by roughly a half, the recursion is guaranteed to terminate. After each recursive step, the remaining set contains a close pair of S. Step 1 takes $O(n)$ time in the worst case, if we use the deterministic median-finding algorithm. Step 2 takes $O(n)$ time based on the result from Part (b). Therefore, the running time of the algorithm is given by the following recurrence: $T(n) = T(n/2) + O(n)$, with the solution $T(n) = O(n)$.