Data Structure HW04

112590022

1. Pseudo code:

```
for i in range (1, n-1):
    current ← s.head
    while current.next != null
        if current.value > current.next.value
            swap(current, current.next)
        current ← current.next
```

→ Time complexity = $O(n^2)$

2. in-order traversal ∋ finding the k-th largest key
reverse traversal ∋ finding the k-th smallest key by binary search tree's orderliness.
Time complexity = $O(n)$

3.
Pseudo code:

```
function merge_heaps (T1, T2):
    if not T1 then return T2
    if not T2 then return T1
    if T1.value > T2.value then swap(T1, T2)
    T1.right = merge_heaps(T1.right, T2)
    swap(T1.right, T1.left)
    return T1
T = merge_heaps(T1, T2)
```

Time complexity = $O(h_1 + h_2)$

4. (a) For each node, we can choose whether to swap its left and right subtrees.
Therefore, if T has n nodes, the total number of binary trees isomorphic to T is 2^n. (two choice)

(a) pseudo code:

```
function isomorphic_distance(T1, T2):
    if not T1 and not T2 then return 0
    if not T1 or not T2 then return float('inf')  Trees are not isomorphic
    if T1.value != T2.value then return float('inf')

    dist1 = isomorphic_distance(T1.left, T2.left) + isomorphic_distance(T1.right, T2.right)
    dist2 = isomorphic_distance(T1.left, T2.right) + isomorphic_distance(T1.right, T2.left)
    return min(dist1, dist2) + (1 if dist1 != dist2 else 0)
```

time complexity = O(n)