1. Provide solutions (using big-Oh or big-Theta) for the following recurrence relations.

(a) \( t(n) = 7 \ t(\frac{n}{5}) + n^2 \)
(b) \( t(n) = 7 \ t(\frac{n}{5}) + 313 \ast n \)
(c) \( t(n) = 36 \ t(\frac{n}{6}) + n^2 \)
(d) \( t(n) = 27 \ t(\frac{n}{3}) + n(\lg n)^3 \)

(Sol’n)

(a) \( t(n) = \Theta(n^2) \) by case 3 (\( \log_5 7 < 2 \)).
(b) \( t(n) = \Theta(n^{\log_5 7}) \) by case 1 (\( \log_5 7 > 1 \)).
(c) \( t(n) = \Theta(n^2 \log n) \) by case 2.
(d) \( t(n) = \Theta(n^3) \) by case 1 (\( \log_3 27 = 3 \), so \( n(\lg n)^3 = O(n^{3-\epsilon}) \) for say \( \epsilon = 1 \)).

□

2. Into an initially empty AVL tree, insert the following values:

\[ 1, 3, 25, 20, 35, 15, 12, 18, 5, 10, 29.\]

(Sol’n)

See figure 1 for the final tree.

□

3. Insert the values above into an initially empty 2-3-4 tree.

(Sol’n)

See figure 2 for the final tree. This uses the standard insertion followed in class - bottom-up, node split just before overflow. Other versions acceptable.

□

4. What are the run-times of the following pieces of code?

(a) \( \text{for } i = 1 \text{ to } n*(\lg n) \)

\[ \text{for } j = 1 \text{ to } i \]

\[ \text{sum++} \]
5. Suppose we augment a BST by adding to each node a field called size. So given a node \( p \), the field \( p.size \) contains the number of nodes in the subtree of \( p \).

We want to use this field to efficiently implement a select method. The idea is that \( \text{select}(k,p) \) will find the \( k \)th smallest value in the subtree of \( p \) for each integer \( k \) with \( 1 \leq k \leq p.size \). Thus, \( \text{select}(1,T.root) \) returns the smallest value in \( T \), \( \text{select}(2,T.root) \) returns the 2nd smallest value stored in \( T \), and \( \text{select}(T.root.size,T.root) \) returns the largest value in \( T \).

Imagine a tree \( T \) with the following characteristics at the top (let \( r = T.root \) be the root pointer):

- \( r.size = 22 \) (the tree contains 22 values)
- \( r.left.size = 15 \) (the left subtree contains 15 values)
Figure 2: The 234 tree for problem 3.

- \( r.right.size = 6 \) (the right subtree contains 6 values)

(a) If we execute \( \text{select}(5, r) \), will the result be found in the left or right subtree of \( r \)? How about \( \text{select}(18, r) \)? Or \( \text{select}(16, r) \)?

(b) A call to \( \text{select}(22, r) \) will find the result in the right subtree of \( r \). For what value of \( k \) will \( \text{select}(k, r.right) \) return the same thing?

(c) Write a short procedure (maybe recursive) to find \( \text{select}(k, p) \) whose run-time is bounded by the height of the tree.

(Sol’n)

For part (a), obviously the smallest 15 elements are in the left subtree of \( r \). So \( \text{select}(5, r) \), which is looking for the 5th smallest element, will return something in the left subtree of \( r \). Similarly, \( \text{select}(18, r) \) would return something in the right subtree of \( r \). \( \text{select}(16, r) \) would return the value stored in \( r \), since 15 values are smaller than that.

For part (b), a search to the right of \( r \) looks at items bigger than that in \( r \) and \( r \)'s left subtree. There are \( 1 + r.left.size = 1 + 15 = 16 \) such items. So we set

\[
k \leftarrow k - (r.left.size + 1) = 22 - 16 = 6.
\]

The code below answers part (c). For simplicity, assume that if \( p.left \) is null, then \( p.left.size \) returns 0.

```plaintext
procedure SELECT(int k, node p)
    if p=null return error
    if (k<p.size) or (k>p.size) return error

    if k <= p.left.size
        return SELECT(k, p.left)

    if k = 1+p.left.size
        return p
```

The code above answers part (c). For simplicity, assume that if \( p.left \) is null, then \( p.left.size \) returns 0.
return p.value

return SELECT( k-(1+p.left.size), p.right)