Please solve the following 2 problems, both of which have multiple parts.

1. Suppose you are given an instance sets of the MyDisjSets class representing a partition of the set \( \{0, 1, \ldots, n-1\} \) into disjoint subsets. Instances of MyDisjSets are stored as a single array representing a collection of rooted trees as seen in class. However, the details of how \( \text{union} \) and \( \text{find} \) are implemented are unknown. In particular, you have no idea how the trees look except that each tree contains exactly the members of one set in the partition.

Now, consider the following method:

```java
public void labelEverything(MyDisjSets sets, int n) {
    int[] label = new int[n];
    for (int i = 0; i < n; i++) {
        label[i] = sets.find(i);
    }
    return label;
}
```

(a) What is the worst-case running time of \( \text{labelEverything} \) if we implement \( \text{find} \) without path compression?
(b) Prove that if we implement \( \text{find} \) using path compression, \( \text{labelEverything} \) runs in \( O(n) \) time in the worst case.

2. Using \( \Theta \)-notation, provide asymptotically tight bounds in terms of \( n \) for the solution to each of the following recurrences. Assume each recurrence \( T(n) \) has a non-trivial base case of \( T(n) = \Theta(1) \) for all \( n < n_0 \) where \( n_0 \) is a suitably large constant. For example, if asked to solve \( T(n) = 2T(n/2) + n \), then your answer should be \( \Theta(n \log n) \). You do not need to justify your solutions.

(a) \( A(n) = 3A(n/3) + n \)
(b) \( B(n) = 2B(n/3) + n \)
(c) \( C(n) = 4C(n/3) + n \)
(d) \( D(n) = 9D(n/3) + n^2 \)
(e) \( E(n) = 5E(n/2) + n^2 \)