CS 173 Discussion 10: Trees and Asymptotic Analysis

Date: October 31/November 1, 2019.

Problem 1. Recall that a *full* m-ary tree is a rooted tree such that every internal node has exactly m children. Suppose T is a full m-ary tree with i internal nodes.

1. How many (total) nodes does T have?
2. How many leaves does T have?
3. A *full* m-ary tree is a rooted tree where all leaves are at the same level. Consider a full and complete m-ary tree of height h. How many leaves does it have?

Problem 2. Use the definition of $O(\cdot)$ to show that $\frac{n^3+2n}{2n+1} = O(n^2)$.

Problem 3. The algorithm below sorts an array of integers which is "almost" sorted in the sense that every integer starts off at distance at most k from its position in the sorted (in ascending order) array. To be precise, let the position of an integer in the unsorted array be i and let the position of the integer in the array after sorting be j. Then if $|i - j| \leq k$ for all the values in the input array, the output array will be completely sorted.

The function `min` returns the smaller of its two inputs. The function `swap` swaps the values stored in the two positions of the array. Assume that each line in the pseudo-code takes one unit of time.

```
almostSorted (k, a[1 ..  n])
    for i = 1 to n
        m = a[i]
        mp = i
        for j = i+1 to min(i+k,n)
            if (a[j] < m)
                m = a[j]
                mp = j
        swap(a[i],a[mp])
    return a[]
```

Give a tight big-O bound on the running time of the above algorithm.