COMP3506/7505: Special Exercise Set 3

Prepared by Yufei Tao

Problem 1. Let $f(n)$ be a function of positive integer n. We know:

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 2 \\ f(n) &= 3 + f(n-2). \end{aligned}$$

Prove $f(n) = O(n)$.

Problem 2. Let $f(n)$ be a function of positive integer n. We know:

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 2 \\ f(n) &= n/10 + f(n-2). \end{aligned}$$

Prove $f(n) = O(n^2)$.

Problem 3. Let $f(n)$ be a function of positive integer n. We know:

$$\begin{aligned} f(1) &= 1 \\ f(n) &= 5n + f(\lceil n/1.01 \rceil). \end{aligned}$$

Prove $f(n) = O(n)$. Recall that $\lceil x \rceil$ is the ceiling operator that returns the smallest integer at least x. (Hint: the master theorem)

Problem 4. Let $f(n)$ be a function of positive integer n. We know:

$$\begin{aligned} f(1) &= 1 \\ f(n) &= 10 + 2 \cdot f(\lceil n/8 \rceil). \end{aligned}$$

Prove $f(n) = O(n^{1/3})$.

Problem 5. Let us revisit the dictionary search problem again. Recall that n integers have been stored in an array in ascending order. The goal is to determine whether an integer v is in the array. Consider the following recursive algorithm:

1. If $n = 0$, return "no". Otherwise, proceed to the next step.
2. Compare v to the $\lceil n/3 \rceil$-th element e of the array.
3. If $v = e$, return "yes".
4. If v is smaller than e, recur in the part of the array before e. Otherwise, recursively recur in the part of the array after e.

Note that the algorithm differs from binary search in that, it does not compare v to the middle element of the array, but the $\lceil n/3 \rceil$-th element instead. Prove that the algorithm has running time $O(\log n)$, that is, asymptotically the same as binary search.

Problem 6. Consider a set S of n integers that are stored in an array (*not* necessarily sorted). Let e and e' be two integers in S such that e is positioned before e'. We call the pair (e, e') an *inversion* in S if $e > e'$. Write an algorithm to count the number of inversions in S. Your algorithm must terminate in $O(n^2)$ time.

For example, if the array stores the sequence $(10, 15, 7, 12)$, then your algorithm should return 3 because there are 3 inversions: $(10, 7)$, $(15, 7)$, and $(15, 12)$.