Algorithm: Homework #1

Due Date: April 12, 2018

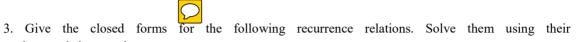
1. Give the pseudo-code of a logarithmic-time ($\theta(\log n)$ -time) algorithm for computing the n-th



Fibonacci number. (cf. Sections 1.4 and 7.2 in the auxiliary textbook.)



2. Prove that $n^2 \in O(2^n)$ using the formal definition of big-O notation.



- characteristic equations. $(1) \ f_{n+2} = f_{n+1} + f_n \ (n \ge 0), \ f_0 = 0, f_1 = 1$
- (2) $f_{n+2} = f_{n+1} + f_n + 1 \ (n \ge 0), \ f_0 = 0, f_1 = 1$
- (3) T(n) = 2T(n/2) + n/2, T(1) = 1 (You may assume that $n = 2^k, k > 0$)
- (4) T(n) = 3T(n/3) + n 2, T(1) = 1 (You may assume that $n = 3^k, k > 0$)
- 4. Consider MergeSort algorithm in the textbook.
- (1) Given array size n, find a recurrence relation for the **best-case** time complexity for MergeSort.
- (2) Solve the recurrence relation for (1), given $n = 2^k$ for some integer k > 0).
- 5. (Theoretically Fast QuickSort) There exists a linear-time (O(n)-time) algorithm that computes a median value among given n values. Assume that we have already known this algorithm. Using this algorithm, (1) design a variant QuickSort algorithm of which the worse-case time complexity is $O(n \log n)$ (just give its pseudo-code) and (2) prove that its time complexity is $O(n \log n)$.

(cf. Sections 6.4, 6.5, and 8.5 in the auxiliary textbook.)



Final location of a pivot:

$$\lfloor \frac{high-low}{2} \rfloor$$
 -th value