1. Can write down the formal definition of O (Oh), Ω (Omega) and Θ (Theta), o (oh), ω (omega) in terms of constants c and N.
2. Can use the formal definition of O, Ω, θ, o, $and\ \omega$ to prove function g(n) $\in$ O(f(n)) and other asymptotic notations [See theory HW1, Question 2].
3. Can use limits to prove θ, o, $and\ \omega$, either by L'Hopital's Rule or simple calculation [See theory HW1, Question 3].
4. Can order the functions by their asymptotic growth rates [See ch2-complexity-growth, slide 5 and slide 44, and theory HW1, Question 1 and 4].
5. Can prove correctness of an algorithm by (1) figuring out a loop invariant for a given program code (e.g. insertion sort), (2) showing that the initialization holds, (3) showing that the maintenance step holds, (4) showing that the algorithm is correct at the termination step. [See ch3-correctness, all slides; theory HW1, Question 6; and theory HW2.2, Question 3].
6. Can analyze the time complexity of an iterative algorithm [See theory HW1, Question 7 and 8].
7. Can write a recursive algorithm to solve a simple realistic problem using the divide and conquer approach and derive the recurrence equation for its time complexity [See theory HW2.2, Question 1].
8. Can derive a recurrence equation for a recursive algorithm (e.g. you are given the Merge Sort algorithm in slide 43 in ch4-recurrence) [See theory HW2.1, Question 3].
9. Can solve recurrence equations by the Master theorem, iteration method, and recursion tree methods. [See theory HW2.1, Question 1 and 2 and 3].
10. Be familiar with Max-Heapify algorithm and its time complexity analysis, and can illustrate the algorithm using examples [See ch6-sorting-heap-linear, slides 15-17].
11. Be familiar with Heap Sort algorithm and its time complexity analysis, and can illustrate the algorithm using examples [See ch6-sorting-heap-linear, slides 22-25, and theory HW2.2, Question 2].