RA 11

7.1 An introduction to algorithms

- An algorithm is a step by step method for solving a problem.

- Algorithms are often described in pseudocode, which is a language in between written English and a computer language.

- An important type of step in an algorithm is assignment, in which a variable is given a value.
  - This is denoted $x := y$

- The output of an algorithm is specified by a return statement.

7.2 Asymptotic growth of function

- Let f be a function that maps positive integers to nonnegative real numbers. The asymptotic growth of the function f is a measure of how fast the output $f(n)$ grows as the input n grows.

- O-notation:
  - Let f and g be functions from $\mathbb{Z}^+$ to $\boxed{\mathbb{R}^+ \cup \{0\}}$
  - Then f is $O(g)$ if there are positive real numbers c and n_0 such that for any $n \in \mathbb{Z}^+$ where $n \geq n_0$, $f(n) \leq c \cdot g(n)$

- In showing that a polynomial function $f(n)$ of degree k is $O(n^k)$, the following combination work as a witness:
  - $n_0 = 1$
  - c = the sum of the positive coefficients in f (including the constant term)

Example $f(n) = 4n^5 + 9n^4 - 8n^3 + 2$, a proof that f is $O(n^5)$ could use: $c = 4+9+2 = 15$
$n_0 = 1$

- Ω notation
  - Let f and g be functions from $\mathbb{Z}^+$ to $[\mathbb{R}^+ \cup \{0\}]$. Then f is $\Omega(g)$ if there are positive real numbers c and n_0 such that for any $n \in \mathbb{Z}^+$ where $n \geq n_0$, $f(n) \geq c \cdot g(n)$

1. Theorem:
   - Let f and g be functions from $\mathbb{Z}^+$ to $[\mathbb{R}^+ \cup \{0\}]$. Then f is $\Omega(g)$ i.f.f. g is $O(f)$

- In proving that f is $\Omega(g)$, many different choices of c and n_0 will suffice.
  - Rule of thumb:
    - Let a^k be the coefficient on the highest degree term.
      - If f has no negative coefficients, then $c = a_k$ and $n_0 = 1$ will suffice.
      - If f has negative coefficients (but $a_k > 0$), then let A be the sum of the absolute values of the negative coefficients in $f(n)$. The choices $c = \frac{a_k}{2}$ and $n_0 = \max\{1, \frac{2A}{a_k}\}$ are sufficient.

- Θ notation
  - indicates two functions have the same rate of growth
  - Let f and g be functions from $\mathbb{Z}^+$ to $[\mathbb{R}^+ \cup \{0\}]$. Then f is $\Theta(g)$ if f is $O(g)$ and f is $\Omega(g)$
  - If f is $\Theta(g)$ then f is order of g.

- Asymptotic growth of logarithm functions with different bases
  - Let a and b be two real numbers greater than 1, then: $\log_a n$ is $\Theta(\log_b n)$

Rules for asymptotic growth of functions

- Let f, g, and h be functions from $\mathbb{Z}^+$ to $\boxed{\mathbb{R}^+ \cup \{0\}}$
  - If f is $O(h)$ and g is $O(h)$, then $f+g$ is $O(h)$
  - If f is $\Omega(h)$ and g is $\Omega(h)$, then $f+g$ is $\Omega(h)$
  - If f is $O(g)$ and c is a positive real number, then $c \cdot f$ is $O(g)$.
  - If f is $\Omega(g)$ and c is a positive real number, then $c \cdot f$ is $\Omega(g)$.
  - If f is $O(g)$ and g is $O(h)$, then f is $O(h)$.
  - If f is $\Omega(g)$ and g is $\Omega(h)$, then f is $\Omega(h)$.

7.3 Analysis of algorithms

- Computational Complexity: the amount of resources an algorithm uses
  - time complexity: amount of time the algorithm requires to run.
  - space complexity: amount of memory used.

- The time complexity of an algorithm is defined by a function $f: \mathbb{Z}^+ \to \mathbb{Z}^+$ such that $f(n)$ is the maximum number of atomic operations performed by the algorithm on any input size n.

- The asymptotic time complexity of an algorithm is the rate of asymptotic growth of the algorithms time complexity function $f(n)$.

- The **worst-case analysis** of an algorithm evaluates the time complexity of the algorithm on the input of a particular size that takes the longest time.

- The **worst-case** time complexity function $f(n)$ is defined to be the maximum number of atomic operations the algorithm requires, where the maximum is taken over all inputs of size n.

  - When proving an **upper bound** on the worst-case complexity of an algorithm (using Oh-notation), the upper bound must apply for every input size of n.
  - When proving the **lower bound** for the worst-case complexity of an algorithm (using Ω notation), the lower bound need only apply for at least one possible input size of n.

- **Average-case analysis**: evaluates the time complexity of an algorithm by determining the average running time of the algorithm on a random input.