Contents

Chapter 1

Growth of Functions

Definition:

Θ-notation asymptotically tight bound

$$\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2, n_0 \text{ s.t. } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \ \ \forall n \ge n_0\}$$

O-notation asymptotic upper bound

$$O(g(n)) = \{f(n) \mid \exists c, n_0 \text{ s.t. } 0 \le f(n) \le c g(n) \ \ \forall n \ge n_0\}$$

Ω-notation asymptotic lower bound

$$\Omega(g(n)) = \{f(n) \mid \exists c, n_0 \text{ s.t. } 0 \le c g(n) \le f(n) \ \ \forall n \ge n_0\}$$

o-notation asymptotically smaller

$$o(g(n)) = \{f(n) \mid \forall c > 0,\, \exists n_0 \text{ s.t. } 0 \le f(n) < c g(n) \ \ \forall n \ge n_0\}$$

ω-notation asymptotically larger

$$\omega(g(n)) = \{f(n) \mid \forall c > 0,\, \exists n_0 \text{ s.t. } 0 \le c g(n) < f(n) \ \ \forall n \ge n_0\}$$

Proposition 1.1.

1. *For any two function $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.*

2. *$f(n) = \omega(g(n))$ if and only if $g(n) = o(f(n))$ and $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$*

A function $f(n)$ is **polylogarithmically bounded** if $f(n) = O\big(\lg^k n\big)$. Any exponential function with a base strictly greater than 1 grows faster than any polynomial function and any polynomial function grows faster than any polylogarithmic function.

Remark 1 (Stirling's approximation).

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right) \tag{1.1}$$

Theorem 1.2 (Master's theorem). *Let $a \geq 1$ and $b > 1$ be constants. The recurrence*

$$T(n) = aT\left(\frac{n}{b}\right) + f(b)$$

has the bounds

1. *If $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for some $\epsilon > 0$ then $T(n) = \Theta\left(n^{\log_b a}\right)$.*
2. *If $f(n) = \Theta\left(n^{\log_b a}\right)$ then $T(n) = \Theta\left(n^{\log_b a} \lg n\right)$.*
3. *If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ for some $\epsilon > 0$ and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n then $T(n) = \Theta(f(n))$.*

Chapter 2

Data Structures

2.1 Elementary structures

2.1.1 Stack

Stack is FIFO, *First in first out.* We can do m pushes (with doubling if needed) and n pops in the order of $O(m+n)$.

Example 2.1. Some examples of stacks include

1. Bracket matching

2.1.2 Queue

Queue is LIFO, *Last in first out.* Pushing (amortized) and poping is done in constant time, $O(1)$.

Example 2.2. Some examples of stacks include

1. simulating queues -_-

2.1.3 Linked lists

In **singly linked list** every element points to the next element. In **doubly linked list** every element points to the next and previous element. In **circulare linked list** the last element's next is the head and the head's previous is the last. Insertion and deletion is done in constant time.