1. Complexity Analysis

(1) Landau Symbols

Definitions with $\lim_{n\to\infty}$:

$$
\begin{aligned}
f(n) = \Theta(g(n)) &\impliedby \lim_{n\to\infty} \frac{f(n)}{g(n)} = c \\
f(n) = O(g(n)) &\impliedby 0 \le \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty \\
f(n) = \Omega(g(n)) &\impliedby 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} \le \infty \\
f(n) = o(g(n)) &\iff \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \\
f(n) = \omega(g(n)) &\impliedby \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty
\end{aligned}
$$

Definitions without $\lim_{n\to\infty}$:

$$
\begin{aligned}
f(n) = \Theta(g(n)) &: \exists c_1, c_2 \in \mathbb{R}^+, 0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \text{ as } n \to \infty. \\
f(n) = O(g(n)) &: \exists c \in \mathbb{R}^+, 0 \le f(n) \le c \cdot g(n) \text{ as } n \to \infty. \\
f(n) = \Omega(g(n)) &: \exists c \in \mathbb{R}^+, 0 \le g(n) \le c \cdot f(n) \text{ as } n \to \infty. \\
f(n) = o(g(n)) &: \forall c \in \mathbb{R}^+, 0 \le f(n) < c \cdot g(n) \text{ as } n \to \infty. \\
f(n) = \omega(g(n)) &: \forall c \in \mathbb{R}^+, 0 \le g(n) < c \cdot f(n) \text{ as } n \to \infty.
\end{aligned}
$$

L'Hopital's rule

$$
\lim_{n\to\infty} \frac{\ln(n)}{n^p} = \lim_{n\to\infty} \frac{\frac{1}{n}}{pn^{p-1}} = \lim_{n\to\infty} \frac{1}{pn^p} = \frac{1}{p} \lim_{n\to\infty} n^{-p} = 0
$$

Order

$$
1 < \log n < n < n \log n < n^2 < n^2 \log n < n^3 < 2^n < 3^n < n! < n^n
$$

(2) Code Analysis

`for (i = 0 ; i < n ; i++){//do something}` $\Theta(n)$;

`if (condition){//true body}` $\Theta(1)$;

`if C then S1 else S2` $T(C) + \max\{T(S1), T(S2)\}$;

Recursive functions time analysis will be taught more in the Lecture of Divide and Conquer.

(3) Best-, Worst-, and Average-case

- Average-case complexity: the expectation (need to choose a probability distribution over inputs)
- Amortized complexity: $\frac{\text{total complexity}}{\text{number of operations}}$

2. Hash Table

Application: DNS; `dict` in Python; `std::unordered_map` in C++.

(1) Hash Function Properties

- Deterministic;
- Equal objects hash to equal values;
- Fast is better;
- Map to an index $0, \cdots, M-1$ (uniformly better).

(2) Dealing with Collisions

Load factor $\lambda = \frac{n}{M}$: the average number of objects per bin.

Probability of at least one collision: increases very fast as λ increases (birthday paradox)

$$1 - \frac{M(M-1)\dots(M-n+1)}{M^n}$$

Ways to deal with collisions:

- chained hash table: searching and inserting is $\Theta(\lambda)$ on average.
- open addressing
  - linear probing: insertion and deletion; lazy erasing; primary clustering.
  - quadratic probing: no primary clustering; probe sequence may $< M$.
  - double hashing: $h_1(k) + i \cdot h_2(k)$