Asymptotic Notation

- $f(n) = o(g(n)) : \; 0 \leq f(n) < cg(n))$
- $f(n) = O(g(n)) : \; 0 \leq f(n) \leq cg(n)$ (upper bound)
- $f(n) = \Theta(g(n)) \leftrightarrow \Omega(g(n))$ & $O(g(n))$
- $f(n) = \Omega(g(n) : \; 0 \leq cg(n) \leq f(n)$ (lower bound)
- $f(n) = \omega(g(n) : \; 0 \leq cg(n) < f(n)$

Useful Equations

$$\begin{aligned}
\sum_{k=1}^{n} k &= \frac{1}{2}n(n+1) = \Theta(n^2) \\
\sum_{k=1}^{n} p^i &= \frac{1-p^{k+1}}{1-p} \\
\text{Stirling:} \quad n! &\approx \sqrt{2\pi n}(\frac{n}{e})^n
\end{aligned}$$

Amortized Analysis

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\
\sum_{i=1}^{n} \hat{c}_i &= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)
\end{aligned}$$

Master Method For recurrences of the form $T(n) = aT(n/b) + f(n)$, where a is the number of branches at each level, b is the reduction in size at each level, and $f(n)$ is the initial cost:

Case 1: $f(n) = O(n^{\log_b a - \epsilon}) \rightarrow T(n) = \Theta(n \log_b a), \epsilon > 0$: $f(n)$ is polynomially slower than $n^{\log_b a}$, weight increases toward leaves because $b < a$

Case 2: $f(n) = \Theta(n^{\log_b a}) \rightarrow T(n) = \Theta(n^{\log_b a} \log^{k+1} n), k \geq 0$: $f(n)$ & $n^{\log_b a}$ grow similarly, weight remains constant because $b = a$

Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon}) \rightarrow T(n) = \Theta(f(n))$: weight decreases because $b > a$

Amortized Analysis The amortized cost (upper bound) $\hat{c}_i$ for the potential function Φ and data D_i is defined as:

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\
\sum_{i=1}^{n} \hat{c}_i &= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)
\end{aligned}$$

Hashing A hash function is *perfect* if it causes no collisions, but there is no perfect hash function if $|S| > m$.

Chaining: put a linked list at each of the slots in the table, cost is proportional to length of the lists. Good hash functions:

$$h(k) = kmod\ m$$
$$h(k) = (ak + b)mod\ m$$
$$h(k) = ((ax + b)mod\ p)mod\ m$$

A family H of hash functions from U to M is 2-universal if for all $x \neq y$, $Pr(h(x) = h(y)) \leq \frac{1}{m}$. BUT there are u^m functions from U to M, requiring $m \log u$ bits to choose/represent/store. We just pick one at random. For r operations, the expected total work is $r(1 + \frac{s}{m})$.

Good hashing: Let m be a prime number, $(x_0, ..., x_r)$ represent a key x, $\bar{a} = (a_0, ..., a_r)$.

$$h_{\bar{a}}(x) = (\sum_{i=0}^{r} a_i x_i)mod\ m$$
$$H = \{h_{\bar{a}}(x) | a_i \in \{0, ..., m-1\}\}$$

Open-addressing: Load factor for number of keys n is $\alpha = \frac{n}{m}$. Search takes $\Theta(1 + \alpha)$ expected time. Because elements are stored in the table itself, the load factor cannot exceed 1. The expected number of probes for inserting in a table with load factor α is $\frac{1}{1-\alpha}$. The expected number of probes for a successful search is $\frac{1}{\alpha} ln \frac{1}{1-\alpha} + \frac{1}{\alpha}$.