

**CMSC 451: Some Useful Background Information**

**Asymptotic Forms:** The following gives both the formal “ $c$  and  $n_0$ ” definitions and an equivalent limit definition for the standard asymptotic forms. Assume that  $f$  and  $g$  are nonnegative functions.

Asymptotic Form	Relationship	Limit Form	Formal Definition
$f(n) \in \Theta(g(n))$	$f(n) \equiv g(n)$	$0 < \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$	$\exists c_1, c_2, n_0, \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n).$
$f(n) \in O(g(n))$	$f(n) \preceq g(n)$	$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$	$\exists c, n_0, \forall n \geq n_0, 0 \leq f(n) \leq cg(n).$
$f(n) \in \Omega(g(n))$	$f(n) \succeq g(n)$	$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0$	$\exists c, n_0, \forall n \geq n_0, 0 \leq cg(n) \leq f(n).$
$f(n) \in o(g(n))$	$f(n) \prec g(n)$	$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$	$\forall c, \exists n_0, \forall n \geq n_0, 0 \leq f(n) \leq cg(n).$
$f(n) \in \omega(g(n))$	$f(n) \succ g(n)$	$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$	$\forall c, \exists n_0, \forall n \geq n_0, 0 \leq cg(n) \leq f(n).$

**L’Hôpital’s rule:** If  $f(n)$  and  $g(n)$  both approach 0 or both approach  $\infty$  in the limit, then

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)},$$

where  $f'(n)$  and  $g'(n)$  denote the derivatives of  $f$  and  $g$  relative to  $n$ .

**Polylog-Polynomial-Exponential:** For any  $a, b$ , and  $c$ , where  $b > 0$  and  $c > 1$ .

$$\log^a n \prec n^b \prec c^n.$$

**Common Summations:** Let  $c \neq 1$  and  $n \geq 0$ .

Name of Series	Formula	Closed-Form Solution
Constant Series	$\sum_{i=a}^b 1$	$= \max(b - a + 1, 0)$
Arithmetic Series	$\sum_{i=0}^n i = 0 + 1 + 2 + \dots + n$	$= \frac{n(n+1)}{2}$
Geometric Series	$\sum_{i=0}^n c^i = 1 + c + c^2 + \dots + c^n$	$= \frac{c^{n+1} - 1}{c - 1}$
Quadratic Series	$\sum_{i=0}^n i^2 = 1^2 + 2^2 + \dots + n^2$	$= \frac{2n^3 + 3n^2 + n}{6}$
Linear-geometric Series	$\sum_{i=0}^{n-1} ic^i = c + 2c^2 + 3c^3 \dots + nc^n$	$= \frac{(n-1)c^{(n+1)} - nc^n + c}{(c-1)^2}$
Harmonic Series	$\sum_{i=1}^n \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$	$\approx \ln n$

**Summations with general bounds:** For  $1 \leq a \leq b$

$$\sum_{i=a}^b f(i) = \sum_{i=0}^b f(i) - \sum_{i=0}^{a-1} f(i).$$

**Approximate using integrals:** Let  $f(x)$  be any *monotonically increasing function* (the function increases as  $x$  increases).

$$\int_0^n f(x)dx \leq \sum_{i=1}^n f(i) \leq \int_1^{n+1} f(x)dx.$$

**(Simplified) Master Theorem for Recurrences:** Let  $a \geq 1$ ,  $b > 1$  be constants and let  $T(n)$  be the recurrence  $T(n) = aT(n/b) + cn^k$ , defined for  $n \geq 0$ .

**Case 1:**  $a > b^k$  then  $T(n)$  is  $\Theta(n^{\log_b a})$ .

**Case 2:**  $a = b^k$  then  $T(n)$  is  $\Theta(n^k \log n)$ .

**Case 3:**  $a < b^k$  then  $T(n)$  is  $\Theta(n^k)$ .

**Comparison-Based Sorting Algorithms:** A *stable* sorting algorithm preserves the relative order of equal elements. An *in-place* sorting algorithm uses no additional array storage (although  $O(\log n)$  additional space is allowed for the recursion stack).

Algorithm	Time	Stable	In-place
BubbleSort	$\Theta(n^2)$	Yes	Yes
InsertionSort	$\Theta(n^2)$	Yes	Yes
MergeSort	$\Theta(n \log n)$	Yes	No
HeapSort	$\Theta(n \log n)$	No	Yes
QuickSort*	$\Theta(n \log n)$	Yes/No	No/Yes

\*There are two versions of QuickSort, one which is stable but not in-place, and one which is in-place but not stable.

**Non-Comparison-Based Sorting Algorithms:** All of these algorithms are stable, but not in-place.

Algorithm	Assumptions	Time	Space
CountingSort	Integers over $[0..k]$	$\Theta(n + k)$	$\Theta(n + k)$
RadixSort	Integers over $[0..n^d]$	$\Theta(d(n + k))$	$\Theta(n + k)$
BucketSort	Integers uniformly distributed	$\Theta(n)$ (Expected)	$\Theta(n)$

**Selection:** For any  $k$ ,  $1 \leq k \leq n$ , the  $k$ th smallest element of a list of size  $n$  can be computed in  $\Theta(n)$  time.

**Useful Data Structures:** All these data structures use  $\Theta(n)$  space to store  $n$  objects.

**Unordered Dictionary:** (by Randomized Hashing)

Insert, delete, and find in  $\Theta(1)$  expected time each.

**Ordered Dictionary:** (by Balanced Trees)

Insert, delete, find, predecessor, successor, merge, split in  $\Theta(\log n)$  time each.

**Priority Queues:** (by Binary Heaps)

Insert, delete, extract-min, union, decrease-key, increase-key in  $\Theta(\log n)$  time. Find-min in  $O(1)$  time each. Make-heap from  $n$  keys in  $\Theta(n)$  time.

**Priority Queues:** (by Fibonacci Heaps)

Any sequence of  $n$  insert, extract-min, union, decrease-key can be done in  $\Theta(1)$  amortized time each. (That is, the sequence takes  $\Theta(n)$  total time.) Extract-min and delete take  $\Theta(\log n)$  amortized time. Make-heap from  $n$  keys in  $\Theta(n)$  time.

**Disjoint Set Union-Find:** (by Inverted Trees with path compression)

Union of two disjoint sets and find the set containing an element in  $\Theta(\log n)$  time each. A sequence of  $m$  operations can be done in  $\Theta(\alpha(m, n))$  amortized time. That is, the entire sequence can be done in  $\Theta(m\alpha(m, n))$  time. ( $\alpha$  is the *very* slow growing inverse Ackerman function.)