## Properties of Logarithms

1. $\log_a 1 = 0$
2. $\log_a a = 1$
3. $\log_a x^y = y \log_a x$
4. $\log_a xy = \log_a x + \log_a y$
5. $\log_a \frac{x}{y} = \log_a x - \log_a y$
6. $a^{\log_b x} = x^{\log_b a}$
7. $\log_a x = \frac{\log_b x}{\log_b a}$

## Combinatorics

1. Number of permutations of an n-element set: $P(n) = n!$
2. Number of k-combinations of an n-element set: $C(n,k) = \frac{n!}{k!(n-k)!}$
3. Number of subsets of an n-element set: $2^n$

## Summations

1. $\sum_{i=1}^{N} C \times i = C \times \sum_{i=1}^{N} i$
2. $\sum_{i=C}^{N} i = \sum_{i=0}^{N-C} (i + C)$
3. $\sum_{i=C}^{N} i = \sum_{i=0}^{N} i - \sum_{i=0}^{C-1} i$
4. $\sum_{i=1}^{N} (A + B) = \sum_{i=1}^{N} A + \sum_{i=1}^{N} B$
5. $\sum_{i=0}^{N} (N - i) = \sum_{i=0}^{N} i$
6. $\sum_{i=1}^{N} 1 = N$
7. $\sum_{i=1}^{N} C = C \times N$

8. $\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$

9. $\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6}$

10. $\sum_{i=0}^{N} A^i = \frac{A^{N+1}-1}{A-1}$ (where $A \neq 1$) [geometric series]

11. $\sum_{i=1}^{N} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N} \approx \log n$, technically $\ln N + \gamma$ [harmonic series]

## Guidelines for Asymptotic Analysis

1) **Loops:** The running time of the loop is, at most, the running time of the statements inside the loop (including tests) multiplied by the number of iterations.

```
for (int i = 1, m = 0; i <= n; i++) { // Execute n times
    m = m + 2;                        // Constant time, c
}
```

Total time: $c \times n = cn = \theta(n)$

2) **Nested loops:** Analyze from the inside out. The total running time is the product of the sizes of all the loops.

```
for (int i = 1, k = 0; i <= n; i++) { // Execute n times
    for (int j = 1; j <= n; j++) {    // Execute n times
        k = k + 2;                    // Constant time, c
    }
}
```

Total time: $c \times n \times n = cn^2 = \theta(n^2)$

3) **Consecutive statements:** Add the time complexities of each statement.

```
int m = 0;                            // Constant time, c0
for (int i = 1; i <= n; i++) {        // Execute n times
    m = m + 2;                        // Constant time, c1
}
int k = 0;                            // Constant time, c0
for (int i = 1; i <= n; i++) {        // Execute n times
    for (int j = 1; j <= n; j++) {    // Execute n times
        k = k + 1;                    // Constant time, c1
        cout << k;                    // Constant time, c2
    }
}
```

Total time: $2c_0 + c_1 n + (c_1 + c_2)n^2 = \theta(n^2)$

4) **If-then-else statements:** Choose the worst-case running time: the test, plus either the `then` or `else` part, whichever is larger.

```
if (n == 0) {                          // Constant time, c0
    return false;                      // Constant time, c1
} else {
    for (int i = 0; i < n; i++) {      // Execute n times
        if (list[i] != list2[i]) {     // Constant time, c2
            return false;              // Constant time, c1
        }
    }
    return true;                       // Constant time, c1
}
```

Total time: $c_0 + c_1 + c_2 n = O(n)$

Notice $O$ instead of $\theta$, since the loop might terminate prior to examining all $n$ elements.

5) **Logarithmic complexity:** An algorithm is logarithmic if it takes constant time to cut the problem size by a fraction (usually by ½).

```
for (int i = 1, m = 1; i <= n; i = i * 2) {
    m = m + 2;
}
```

Total time: $\theta(\lg n)$

```
for (int i = 1, m = 1; i <= n; i = i * 3) {
    m = m + 2;
}
```

Total time: $\theta(\log_3 n)$