

Here is a review of some formulas that you will find useful when doing asymptotic analysis.

- $\sum_{i=1}^N i = 1 + 2 + 3 + 4 + \dots + N = \frac{N(N+1)}{2} = \frac{N^2+N}{2}$
- $\sum_{i=0}^{N-1} 2^i = 1 + 2 + 4 + 8 + \dots + 2^{N-1} = 2 \cdot 2^{N-1} - 1 = 2^N - 1$

Intuition

For the following recursive functions, give the worst case and best case running time in the appropriate $O(\cdot)$, $\Omega(\cdot)$, or $\Theta(\cdot)$ notation.

1.1 Give the running time in terms of N .

```

1 public void andslam(int N) {
2     if (N > 0) {
3         for (int i = 0; i < N; i += 1) {
4             System.out.println("datboi.jpg");
5         }
6         andslam(N / 2);
7     }
8 }

```

$\Theta(N)$

$$\underline{2^n = N}$$

$$\underline{N + \frac{N}{2} + \frac{N}{4} + \dots + 1}$$

$$1 + 2 + 4 + \dots + 2^n = 2^{n+1} - 1 = \underline{2^N - 1}$$

$$\downarrow N * \sum_{i=0}^{\log_2 N} \frac{1}{2^i} \leq \underline{N * 2} \in \Theta(N)$$

- 1.2 Give the running time for `andwelcome(arr, 0, N)` where N is the length of the input array `arr`.

```

1 public static void andwelcome(int[] arr, int low, int high) {
2     System.out.print("[ ");
3     for (int i = low; i < high; i += 1) {
4         System.out.print("loyal ");
5     }
6     System.out.println("]");
7     if (high - low > 0) {
8         double coin = Math.random();
9         if (coin > 0.5) {
10             andwelcome(arr, low, low + (high - low) / 2);
11         } else {
12             andwelcome(arr, low, low + (high - low) / 2);
13             andwelcome(arr, low + (high - low) / 2, high);
14         }
15     }
16 }

```

$$\text{best: } N + \frac{N}{2} + \dots + 1 \quad \Theta(N)$$

$$\text{worst: } N + N + \dots + N \quad \Theta(N \lg N)$$

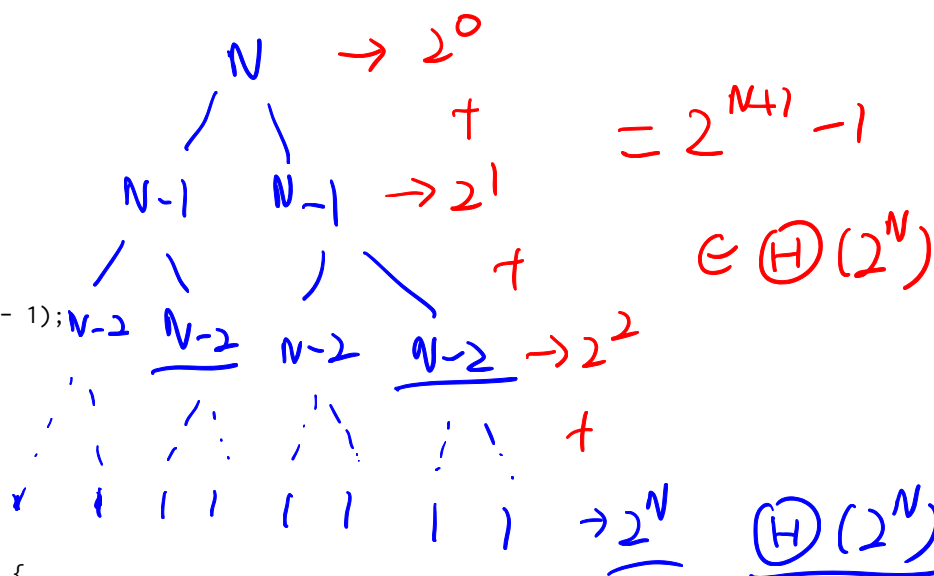
$\lg N$

- 1.3 Give the running time in terms of N .

```

1 public int tothe(int N) {
2     if (N <= 1) {
3         return N;
4     }
5     return tothe(N - 1) + tothe(N - 1);
6 }

```

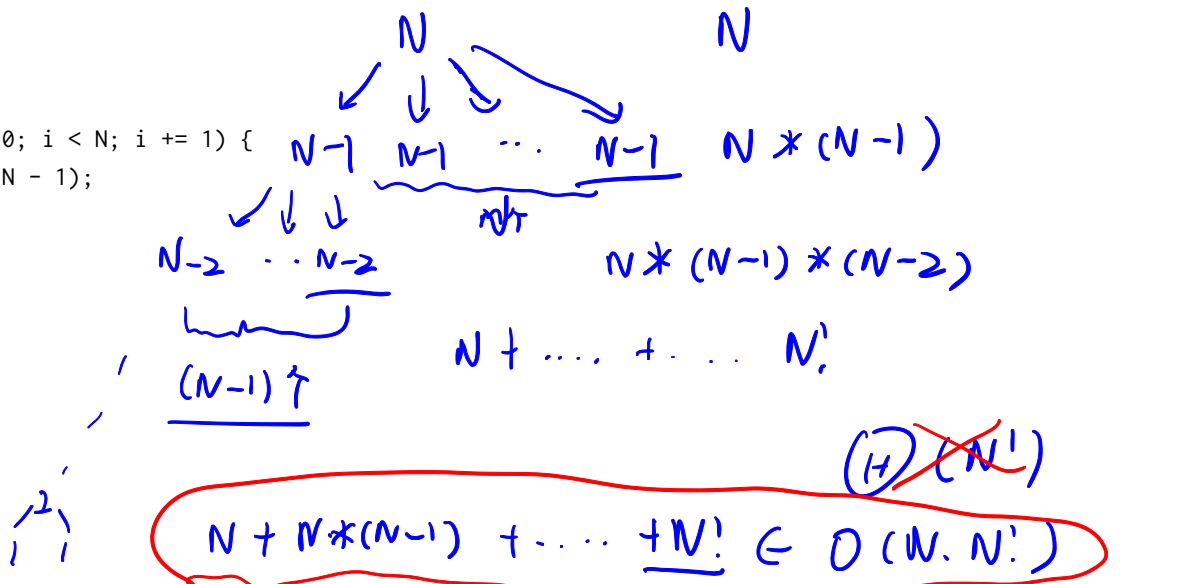


- 1.4 Give the running time in terms of N .

```

1 public static void spacejam(int N) {
2     if (N <= 1) {
3         return;
4     }
5     for (int i = 0; i < N; i += 1) {
6         spacejam(N - 1);
7     }
8 }

```



Hey you watchu gon do

- 2.1 For each example below, there are two algorithms solving the same problem. Given the asymptotic runtimes for each, is one of the algorithms **guaranteed** to be faster? If so, which? And if neither is always faster, explain why.

(a) Algorithm 1: $\Theta(N)$, Algorithm 2: $\Theta(N^2)$

(b) Algorithm 1: $\Omega(N)$, Algorithm 2: $\Omega(N^2)$

NO

(c) Algorithm 1: $O(N)$, Algorithm 2: $O(N^2)$

NO

(d) Algorithm 1: $\Theta(N^2)$, Algorithm 2: $O(\log N)$

(e) Algorithm 1: $O(N \log N)$, Algorithm 2: $\Omega(N \log N)$

maybe

as faster as

Would your answers above change if we did not assume that N was very large (for example, if there was a maximum value for N , or if N was constant)?

Asymptotic Notation

- 3.1 Draw the running time graph of an algorithm that is $O(\sqrt{N})$ in the best case and $\Omega(N)$ in the worst case. Assume that the algorithm is also trivially $\Omega(1)$ in the best case and $O(\infty)$ in the worst case.

Extra: Following is a question from last week, now that you have properly learned about $O(\cdot)$, $\Omega(\cdot)$, or $\Theta(\cdot)$.

- 3.2 Are the statements in the right column true or false? If false, correct the asymptotic notation ($\Omega(\cdot)$, $\Theta(\cdot)$, $O(\cdot)$). Be sure to give the tightest bound. $\Omega(\cdot)$ is the opposite of $O(\cdot)$, i.e. $f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$.

$$f(n) = 20501$$

$$f(n) = n^2 + n$$

$$f(n) = 2^{2n} + 1000$$

$$f(n) = \log(n^{100})$$

$$f(n) = n \log n + 3^n + n$$

$$f(n) = n \log n + n^2$$

$$f(n) = n \log n$$

$$g(n) = 1$$

$$g(n) = 0.000001n^3$$

$$g(n) = 4^n + n^{100}$$

$$g(n) = n \log n$$

$$g(n) = n^2 + n + \log n$$

$$g(n) = \log n + n^2$$

$$g(n) = (\log n)^2$$

$$f(n) \in O(g(n)) \quad \checkmark \quad \textcircled{H}$$

$$f(n) \in \Omega(g(n)) \quad \times \quad \textcircled{O}$$

$$f(n) \in O(g(n)) \quad \checkmark \quad \textcircled{H}$$

$$f(n) \in \Theta(g(n)) \quad \times \quad \textcircled{O}$$

$$f(n) \in \Omega(g(n)) \quad \checkmark$$

$$f(n) \in \Theta(g(n)) \quad \checkmark$$

$$f(n) \in O(g(n)) \quad \times \quad \textcircled{O}$$

Fall 2015 *Extra*

- 4.1 If you have time, try to answer this challenge question. For each answer true or false. If true, explain why and if false provide a counterexample.

- (a) If $f(n) \in O(n^2)$ and $g(n) \in O(n)$ are positive-valued functions (that is for all n , $f(n), g(n) > 0$), then $\frac{f(n)}{g(n)} \in O(n)$.

false

$$g(n) = 1$$

$$f(n) = n^2$$

$$\frac{f(n)}{g(n)} = n^2 \notin O(n)$$

- (b) If $f(n) \in \Theta(n^2)$ and $g(n) \in \Theta(n)$ are positive-valued functions, then $\frac{f(n)}{g(n)} \in \Theta(n)$.