Announcements

- Homework 1 up on website
  - Due Feb. 9th before class
- Office Hour Changes:
  - My OH moved to Wednesday (this week only)
  - Nihkil's OH moved to Thurs 4-6 (was 10-12)

Review

- Administrative announcements
- Brief Introduction

Plan

- Mathematical Background
- Theoretical Algorithm Analysis
  - Big-Oh Notation
  - Examples

Math Background: Exponents

$$
\begin{aligned}
X^A X^B &= X^{A+B} \\
\frac{X^A}{X^B} &= X^{A-B} \\
\left(X^A\right)^B &= X^{AB} \\
X^N + X^N &= 2X^N \neq X^{2N} \\
2^N + 2^N &= 2^{N+1}
\end{aligned}
$$

Math Background: Logarithms

$$X^A = B \text{ iff } \log_X B = A$$

$$\log_A B = \frac{\log_C B}{\log_C A};\ A, B, C > 0, A \neq 1$$

$$\log AB = \log A + \log B;\ \ A, B > 0$$

Math Background: Series

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$$

$$\sum_{i=0}^{N} A^i = \frac{A^{N+1} - 1}{A - 1}$$

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

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6} \approx \frac{N^3}{3}$$

Math Background: Proofs

- Proof by Induction:
  - Prove base case,
  - **Inductive hypothesis.** Prove claim for current state assuming truth in previous state
- Proof by Contradiction: assume claim is false.
  - Show that assumption leads to contradiction

Big-Oh Notation

- We adopt special notation to define **upper bounds** and **lower bounds** on functions
- In CS, usually the functions we are bounding are running times, memory requirements.
- We will refer to the running time as *T(N)*

Definitions

- For *N* greater than some constant, we have the following definitions:

$$
\begin{aligned}
T(N) &= O(f(N)) \leftarrow T(N) \leq cf(N) \\
T(N) &= \Omega(g(N)) \leftarrow T(N) \geq cf(N) \\
T(N) &= \Theta(h(N)) \leftarrow \begin{array}{c} T(N) = O(h(N)), \\ T(N) = \Omega(h(N)) \end{array}
\end{aligned}
$$

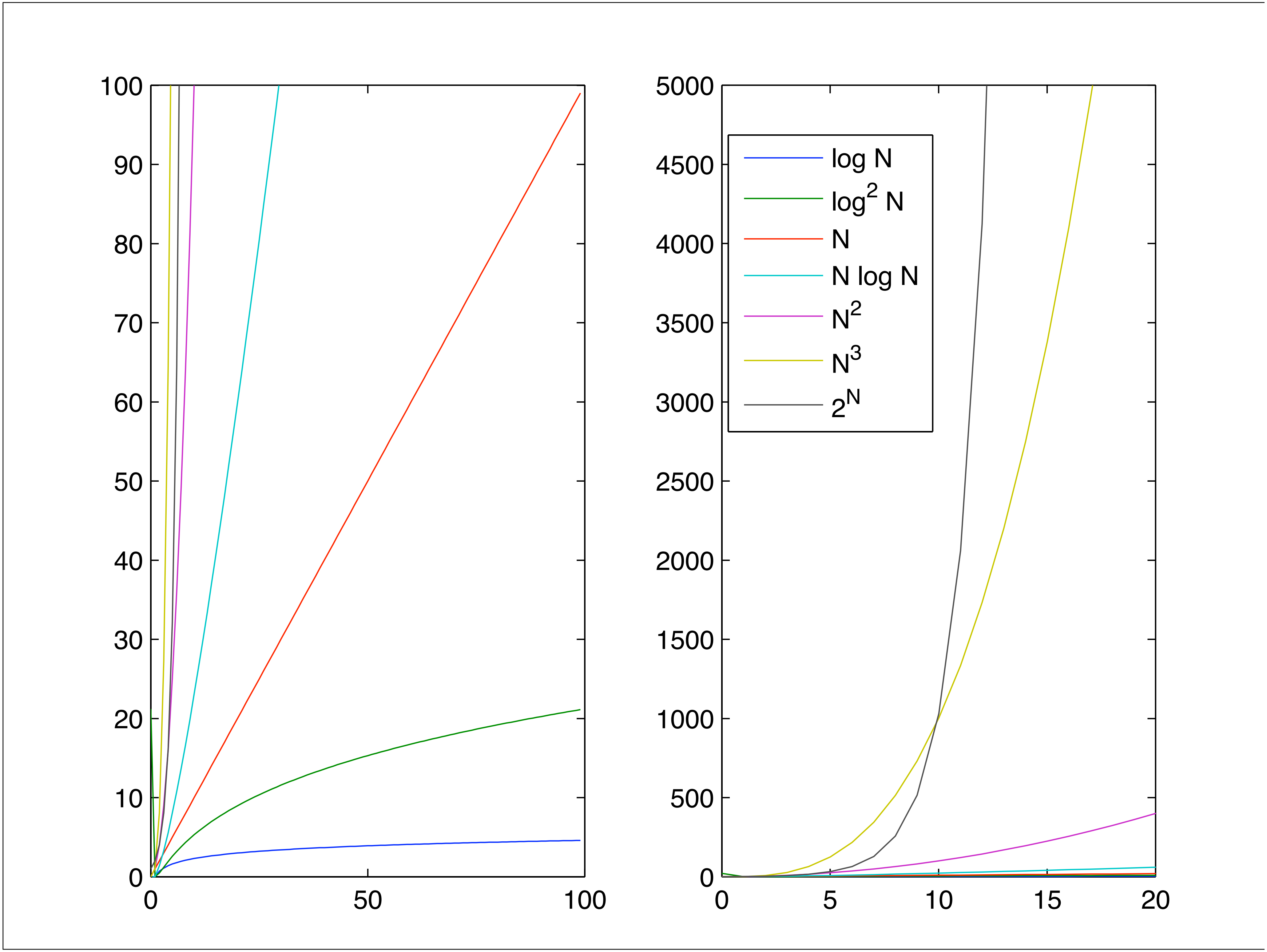
- There exists some constant c such that cf(N) bounds T(N)

Definitions

- Alternately, O(f(N)) can be thought of as meaning

$$T(N) = O(f(N)) \leftarrow \lim_{N \to \infty} f(N) \geq \lim_{N \to \infty} T(N)$$

- Big-Oh notation is also referred to as **asymptotic** analysis, for this reason.



100
90
80
70
60
50
40
30
20
10
0
0
50
100
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
5
10
15
20
log N
log² N
N
N log N
N²
N³
2ᴺ

Comparing Growth Rates

$$T_1(N) = O(f(N)) \text{ and } T_2(N) = O(g(N))$$

then

$$(a) \qquad T_1(N) + T_2(N) = O(f(N) + g(N))$$

$$(b) \qquad T_1(N)T_2(N) = O(f(N)g(N))$$

- If you have to, use l'Hôpital's rule

$$\lim_{N \to \infty} f(N)/g(N) = \lim_{N \to \infty} f'(N)/g'(N)$$

Example: Maximum Subsequence

- Given a sequence of integers (possibly negative), find the subsequence whose sum is the maximum

-2	11	-4	13	-5	-2

Cubic Time Algorithm

```
for i=1 to N {
  for j=i to N {
    sum = 0
    for k=i to j
      sum = sum+A[k]
    if (sum > maxSum)
      maxSum = sum
  }
}
```

$$T(N) = \sum_{i=1}^{N} \sum_{j=i}^{N} \sum_{k=i}^{j} 1$$

Cubic Time Algorithm

$$T(N) = \sum_{i=1}^{N} \sum_{j=i}^{N} \sum_{k=i}^{j} 1$$

$$T(N) = \sum_{i=1}^{N} \sum_{j=i}^{N} j - i + 1$$

$$T(N) = \sum_{i=1}^{N} \frac{(N - i + 2)(N - i + 1)}{2}$$

$$T(N) = \frac{N^3 + 3N^2 + 2N}{6}$$