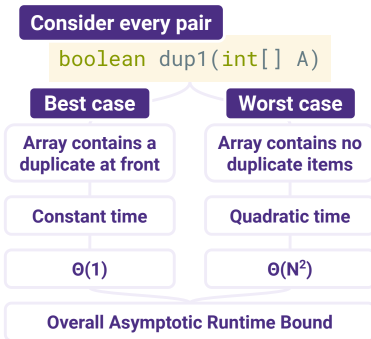
Runtime Analysis Process

Comprehending. Understanding the implementation details of a program.

Modeling. Counting the number of steps in terms of N, the size of the input.

- **Case Analysis**. How certain conditions affect the program execution.
- **Asymptotic Analysis**. Describing what happens for very large N, as N→∞.

Formalizing. Summarizing the final result in precise English or math notation.



The reading described the implementation details for dup1 and dup2 (**Comprehension**) and introduced the idea of counting steps (**Modeling**). In this lecture, we will go in-depth on **modeling** and **formalizing**.

?: Where did case analysis come up in the reading?

Asymptotic Analysis

What happens for very large N, as N→∞.

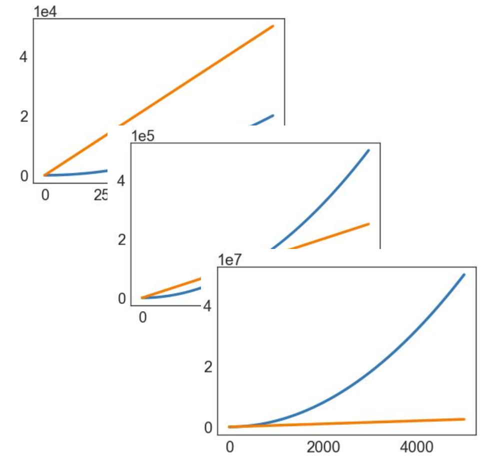
Simulating billions of particles.

Social network with billions of users.

Logging billions of transactions.

Encoding billions of bytes of video data.

Linear-time algorithms **scale better** than quadratic-time algorithms (parabolas).



From this point forward, we'll almost always be working in the mode of asymptotic analysis: considering the behavior of programs as N grows very large.

?: How can we characterize the range of step counts that we saw in dup1 and dup2?

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

Orders of Growth

?: Why might we choose to focus on very large N rather than small N?

?: How do multiplicative constants, e.g. 100N or N^2 / 2, affect the order of growth of the runtime of different algorithms?

Q Asymptotic Analysis and Case Analysis

For a very large array with billions of elements (i.e. asymptotic analysis), is it possible for dup1 to execute only 2 less-than (<) operations?

Operation	dup1: Quadratic/Parabolic	dup2: Linear
i = 0	1	1
less-than (<)	2 to (N^2 + 3N + 2) / 2	0 to N
increment (+= 1)	0 to (N^2 + N) / 2	0 to N - 1
equality (==)	1 to (N^2 - N) / 2	1 to N - 1
array accesses	2 to N^2 - N	2 to 2N - 2


```
public static boolean dup1(int[] A) {
  for (int i = 0; i < A.length; i += 1) {
    for (int j = i + 1; j < A.length; j += 1) {
      if (A[i] == A[j]) {
         return true;
      }
    }
  }
  return false;
}
```

Q1: For a very large array with billions of elements (i.e. asymptotic analysis), is it possible for dup1 to execute only 2 less-than (<) operations?

?: What does the runtime for dup1 vs. dup2 look like if we only consider the best case asymptotic analysis? How does that result compare to the worst case asymptotic analysis?

Q Identifying Orders of Growth

Consider the algorithm step counts below.

What do you expect will be the **order of growth** of the runtime for the algorithm?

A. N [linear]

B. N^2 [quadratic]

C. N^3 [cubic]

D. N^6 [sextic]

Operation	Count
less-than (<)	$100N^2 + 3N$
greater-than (>)	$2N^3 + 1$
and (&&)	5,000

Q1: What do you expect will be the order of growth of the runtime for the algorithm? In other words, if we plotted total runtime vs. N, which curve would we expect?

Simplification 3: Eliminate Lower-Order Terms

Ignore lower-order terms.

```
public static boolean dup1(int[] A) {
  for (int i = 0; i < A.length; i += 1) {
    for (int j = i + 1; j < A.length; j += 1) {
      if (A[i] == A[j]) {
         return true;
      }
    }
  }
  return false;
}
```

Operation	Worst Case: dup1
increment (+= 1)	$(N^2 +$ ~~N~~$) / 2$

?: Why can we ignore lower-order terms?

Q Your Turn: Worst Case Order of Growth for dup2

1. Only consider the **worst case**.
2. Pick a representative operation (cost model).
3. Ignore lower order terms.
4. Ignore multiplicative constants.

Order of growth

Operation	Worst Case Growth

"The worst case order of growth of the runtime for dup2 is …"

Operation	dup2
i = 0	1
less-than (<)	0 to N
increment (+= 1)	0 to N - 1
equality (==)	1 to N - 1
array accesses	2 to 2N - 2

17

Q1: Determine the worst case order of growth for dup2.

Q2: Which operations are appropriate cost models? How do you know?

Simplified Modeling Process

Rather than building the entire table, we can instead:

1. Choose a representative operation to count (cost model).
2. Figure out the order of growth for the count of the representative operation by either:
   - Making an exact count and then discarding the unnecessary pieces.
   - After lots of practice, using inspection to determine order of growth.

Let's redo our analysis of dup1 with this new process.

This time, we'll show all our work.

18

By using our simplifications from the outset, we can avoid building the table at all!

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

```
int N = A.length; // N == 6
for (int i = 0; i < N; i += 1)
  for (int j = i + 1; j < N; j += 1)
    if (A[i] == A[j])
      return true;
return false;
```

"The worst case order of growth of the runtime for dup1 is N^2."

$$C = 1 + 2 + 3 + \cdots + (N-3) + (N-2) + (N-1)$$
$$C = (N-1) + (N-2) + (N-3) + \cdots + 3 + 2 + 1$$
$$2C = N + N + N + \cdots + N + N + N = N(N-1)$$
$$\therefore \quad C = N(N-1)/2$$

Worst Case Order of Growth: **Exact Count of == Operations**

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

```
int N = A.length; // N == 6
for (int i = 0; i < N; i += 1)
  for (int j = i + 1; j < N; j += 1)
    if (A[i] == A[j])
      return true;
return false;
```

"The worst case order of growth of the runtime for dup1 is N^2."

Area of right triangle of side length N - 1.

Order of growth of area is N^2.

Worst Case Order of Growth: **Geometric Argument**

Order of Growth Exercise

Informally, what is the shape of each function for very large N?

In other words, what is the order of growth of each function?

Function	Order of Growth
$N^3 + 3N^4$	
$(1 / N) + N^3$	
$(1 / N) + 5$	
$Ne^N + N$	
$40 \sin(N) + 4N^2$	

23

Q1: Informally, what is the shape of each function for very large N? In other words, what is the order of growth of each function?

Big-Theta Definition

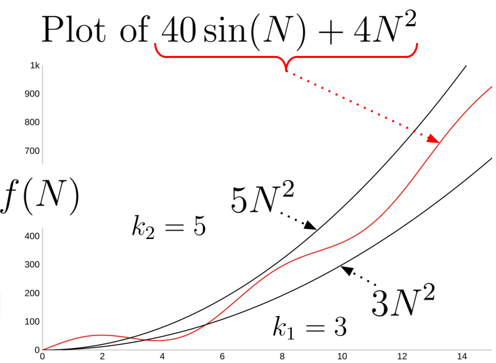
$$R(N) \in \Theta(f(N))$$

means there exist positive constants k_1 and k_2 such that

$$k_1 \cdot f(N) \leq R(N) \leq k_2 \cdot f(N)$$

for all values of N greater than some N_0.

"Very large N"



25

?: What is a value that we can choose for N_0 according to the plot on the right?

Big-Theta Challenge

Demo

$$R(N) = \frac{4N^2 + 3N \ln N}{2}$$

Find a simple f(N) and corresponding k_1 and k_2.

$$R(N) \in \Theta(f(N))$$

means there exist positive constants k_1 and k_2 such that

$$k_1 \cdot f(N) \leq R(N) \leq k_2 \cdot f(N)$$

for all values of N greater than some N_0.

26

Big-O Definition

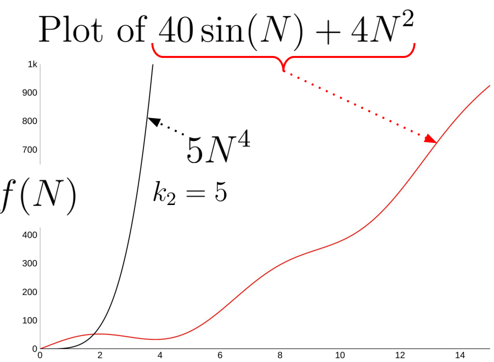
$$R(N) \in O(f(N))$$

means there exists a positive constant k_2 such that

$$R(N) \leq k_2 \cdot f(N)$$

for all values of N greater than some N_0.

"Very large N"



28

Q1: Find a simple f(N) and corresponding k_1 and k_2.

?: Why can we say that 40 sin(N) + $4N^2$ is in O(N^4)? Explain in terms of the formal definition of Big-O.

?: Why is it incorrect to say that 40 sin(N) + $4N^2$ is in Θ(N^4)? Explain in terms of the formal definition of Big-Theta.

Big-Omega Definition

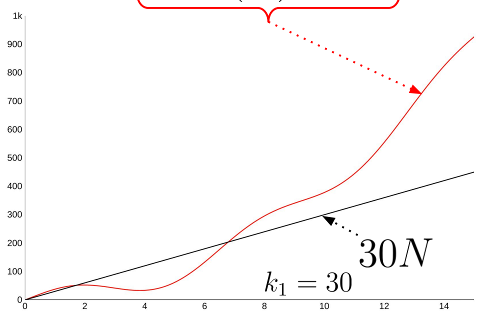
$$R(N) \in \Omega(f(N))$$

means there exists a positive constant k_1 such that

$$k_1 \cdot f(N) \le R(N)$$

for all values of N greater than some N_0.

"Very large N"



Likewise, we have a Big-Omega definition for the other half of the inequality.

?: Describe 40 sin(N) + 4N² ∈ Ω(N) in your own words using the plot on the right.

?: Does Θ(f(N)) imply O(f(N)) and Ω(f(N))? Does O(f(N)) and Ω(f(N)) imply Θ(f(N))?

Q Overall Asymptotic Runtime Bound for dup1

Demo

$$R_{\text{best}}(N) = 2$$

$$R_{\text{worst}}(N) = \frac{N^2 + 3N + 2}{2}$$

Give an overall asymptotic runtime bound for R as a combination of **Θ**, **O**, and/or **Ω** notation. Take into account both the best and the worst case runtimes (R_{best} and R_{worst}).

Q1: Give an overall asymptotic runtime bound for R as a combination of **Θ**, **O**, and/or **Ω** notation. Take into account both the best and the worst case runtimes (R_{best} and R_{worst}).