Algorithm Analysis
CSE 373 Winter 2020

Instructor: Hannah C. Tang

Teaching Assistants:
Aaron Johnston       Ethan Knutson       Nathan Lipiarski
Amanda Park         Farrell Fileas       Sam Long
Anish Velagapudi    Howard Xiao        Yifan Bai
Brian Chan          Jade Watkins       Yuma Tou
Elena Spasova       Lea Quan
Announcements

❖ Schedule for Drop-in Times (finally!!!! 😞) posted
Questions from Reading Quiz

❖ Do we care about the specific operation counts (e.g., calculating 2 to 50,015,001)? What if we are off by one?

❖ What is the conversion between the number of operations and the runtime? I had a little trouble understanding what each "step" was and what counted

❖ Could we go over how to count the operations for dup1?
Lecture Outline

❖ Algorithm Analysis Concepts

❖ Asymptotic Analysis Case Study: dup1 Worst Case

❖ Asymptotic Analysis Simplifications

❖ Case Analysis vs Asymptotic Analysis

❖ Takeaways
Review: What is Runtime Analysis?

- What does it mean for something to be “slow” or “fast”?

- Let’s run it and measure the (wallclock) time! Oh wait …
  - Input can affect time
  - Hardware can affect time
  - Other programs running on the machine can affect time
  - ... and so much more!

- Count how many steps a program takes to execute on an input of size N
Algorithm Analysis: Our Destination

- **Comprehension**: Understanding the implementation
- **Cost Model**: Calculating the cost in terms of N, the size of the input
- **Analysis Types**:
  - **Case Analysis**: How certain conditions affect the program execution
  - **Asymptotic Analysis**: What happens as N→∞
- **Formalizing**: Summarizing the final result in precise English or a mathematical notation

```java
boolean dup1(int[] A)
```

- **Best case**:
  - **Array contains a duplicate at front**: Constant time, \( \Theta(1) \)
  - **Array contains no duplicate items**: Quadratic time, \( \Theta(N^2) \)

- **Worst case**:
  - **Array contains a duplicate at front**: Constant time, \( \Theta(1) \)
  - **Array contains no duplicate items**: Quadratic time, \( \Theta(N^2) \)
Comprehension

❖ What is the data structure or algorithm doing?

```java
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;
}
```
Cost Model: Time

- An algorithm’s runtime is dictated by the number of CPU instructions it executes. This is why we “count steps”: each “step” is an idealized instruction

  - Example CPU instructions: comparing integers (<, ==), incrementing numbers (+=, *), accessing an array

```java
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;
}
```

“Computers are really dumb. They can only do a few things like shuffling around numbers, but they do them really really fast so that they appear smart.”

– Hal Perkins, UW CSE
Cost Model: Space

- We usually talk about time rather than space, but the principles are the same.

- Similar to time, we measure “cost” in higher-level abstractions rather than machine-specific bytes.
  - Examples: queue elements, graph nodes, etc.
Asymptotic Analysis

❖ Asymptotic Analysis is how the algorithm or data structure behaves as \( N \to \infty \)
  - Simulating billions of particles
  - Social network with billions of users
  - Logging billions of transactions
  - Encoding billions of bytes of video data

❖ Why \( \infty \)?
  - We need a way to characterize data structures’ and algorithms’ complexity, which can vary for every finite \( N \)
Asymptotic Analysis Means “Infinity”: Graphically

- Asymptotic analysis is non-intuitive because it deals with infinity!
  - What if you know your algorithm will only run on \( N < 100 \) elements?
  - Why do we “throw away constants”?
  - How come lower-order terms “don’t matter”?

- Linear-time algorithms scale better than quadratic-time algorithms (parabolas).
Asymptotic Analysis Means “Infinity”: Intuitively

- Since we’re dealing with infinity, constants and lower-order terms don’t meaningfully add to the final result
- The highest-order term is what matters and drives growth

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.
Lecture Outline

❖ Asymptotic Analysis Concepts

❖ Asymptotic Analysis Case Study: dup1 Worst Case

❖ Asymptotic Analysis Simplifications

❖ Case Analysis vs Asymptotic Analysis

❖ Takeaways
Counting Steps: Asymptotic Worst Case

- Assume N = A.length = 6
  - I’ve broken up the for-loops to make it easier to count
  - Cheatsheet: $1 + 2 + 3 + 4 + 5 + 6 = 21$

```java
public static boolean dup1(int[] A) {
    int i = 0;
    for ( ; i < A.length; ) {
        int j = i + 1;
        for ( ; j < A.length; ) {
            if (A[i] == A[j]) {
                return true;
            }
            j += 1;
        }
        i += 1;
    }
    return false;
}
```
## Counting Steps: Asymptotic Worst Case

- **Cheatsheet**: $1 + 2 + 3 + 4 + 5 + 6 = 21$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Executions ($N = 6$)</th>
<th>Symbolic Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i = 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Less-than ($&lt;$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Increment (+=)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equals-to (==)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Array accesses</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Counting Steps: Asymptotic Worst Case

- Cheatsheet: $1 + 2 + 3 + \ldots + (N - 1) = ???$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Executions ($N = 6$)</th>
<th>Symbolic Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i = 0$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Less-than ($&lt;$)</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>Increment ($+=)$</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>Equals-to ($==$)</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>Array accesses ($[]$)</td>
<td>42</td>
<td></td>
</tr>
</tbody>
</table>
Counting Steps: Double-checking Our Work

- Demo
Symbolic Expression

\[ C = 1 + 2 + 3 + \cdots + (N - 3) + (N - 2) + (N - 1) \]
\[ C = (N - 1) + (N - 2) + (N - 3) + \cdots + 3 + 2 + 1 \]
\[ 2C = N + N + N + \cdots + N + N + N = N(N - 1) \]
\[ \therefore \quad C = \frac{N(N - 1)}{2} \]

```java
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;
}
```
Symbolic Expression, Intuitively

- We want the area of the right triangle
- If its side length is N, its order of growth is $N^2$.

```java
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;
}
```
Counting Steps: Asymptotic Worst Case

- Cheatsheet: $1 + 2 + 3 \ldots N = (N^2 - N)/2$

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Executions ($N = 6$)</th>
<th>Symbolic Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Less-than (&lt;)</td>
<td>28</td>
<td>$(N^2 + 3N + 2)/2$</td>
</tr>
<tr>
<td>Increment (+=)</td>
<td>21</td>
<td>$(N^2 + N)/2$</td>
</tr>
<tr>
<td>Equals-to (==)</td>
<td>15</td>
<td>$(N^2 - N)/2$</td>
</tr>
<tr>
<td>Array accesses</td>
<td>30</td>
<td>$N^2 + N$</td>
</tr>
</tbody>
</table>
Lecture Outline

❖ Asymptotic Analysis Concepts

❖ Asymptotic Analysis Case Study: dup1 Worst Case

❖ Asymptotic Analysis Simplifications

❖ Case Analysis vs Asymptotic Analysis

❖ Takeaways
Simplification 1: Eliminate Lower-Order Terms

❖ Ignore lower-order terms

<table>
<thead>
<tr>
<th>Operation</th>
<th>Number of Executions (N = 6)</th>
<th>Symbolic Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>i = 0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Less-than (&lt;)</td>
<td>28</td>
<td>(N^2 + 3N + 2)/2</td>
</tr>
<tr>
<td>Increment (+=)</td>
<td>21</td>
<td>(N^2 + N)/2</td>
</tr>
<tr>
<td>Equals-to (==)</td>
<td>15</td>
<td>(N^2 - N)/2</td>
</tr>
<tr>
<td>Array accesses</td>
<td>30</td>
<td>N^2 + N</td>
</tr>
</tbody>
</table>

❖ 6 + (N^2 + 3N + 2)/2 + (N^2 + N)/2 + (N^2 - N)/2 + N^2 + N

❖ 6 + \frac{1}{2}N^2 + \frac{3}{2}N + 1 + \frac{1}{2}N^2 + \frac{1}{2}N + \frac{1}{2}N^2 - \frac{1}{2}N + N^2 + N

❖ 6 + \frac{1}{2}N^2 + \frac{1}{2}N - 1 + \frac{1}{2}N^2 + \frac{1}{2}N + \frac{1}{2}N^2 - \frac{1}{2}N + N^2 - N

❖ \frac{5}{2}N^2
Simplification 2: Eliminate Multiplicative Constants

- Ignore multiplicative constants.
  - We already threw away the meaningful constant when we used a simplified cost model

\[ \frac{5}{2} N^2 \]

- \( N^2 \)
Algorithm Analysis: Our Destination

- "The worst-case order-of-growth for dup1’s runtime is quadratic"
Order of Growth

What is the order of growth for each function?
- (Informally, what is the shape of each function for very large N?)

<table>
<thead>
<tr>
<th>Function</th>
<th>Order of Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N^3 + 3N^4$</td>
<td>$N^4$</td>
</tr>
<tr>
<td>$(1 / N) + N^3$</td>
<td>$N^3$</td>
</tr>
<tr>
<td>$Ne^N + N$</td>
<td>$Ne^N$</td>
</tr>
<tr>
<td>$40 \sin(N) + 4N^2$</td>
<td>$N^2$</td>
</tr>
</tbody>
</table>
Lecture Outline

- Asymptotic Analysis Concepts
- Asymptotic Analysis Case Study: dup1 Worst Case
- Asymptotic Analysis Simplifications
- Case Analysis vs Asymptotic Analysis
- Takeaways
For a very large array with billions of elements (i.e. asymptotic analysis), is it possible for dup1 to execute exactly 1 equals (==) operation?

**YES! In its best case:**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbolic Expression (Worst Case)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals-to (==)</td>
<td>( \frac{N^2 - N}{2} )</td>
</tr>
</tbody>
</table>

```
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;
}
```
Simplification 3: Consider the Worst Case

- Oftentimes, we only care about worst case
  - If a runtime statement doesn’t mention, assume it’s a worst-case analysis

- But dup1’s best case is significantly better than its worst-case!
  - Demo
Algorithm Analysis: Our Destination

❖ “The worst-case order-of-growth for dup1’s runtime is quadratic (parabolic)"

❖ “The best-case order-of-growth for dup1’s runtime is constant”

boolean dup1(int[] A)

Best case

Worst case

Array contains a duplicate at front
Constant time
Θ(1)

Array contains no duplicate items
Quadratic time
Θ(N^2)
Lecture Outline

❖ Asymptotic Analysis Concepts

❖ Asymptotic Analysis Case Study: dup1 Worst Case

❖ Asymptotic Analysis Simplifications

❖ Case Analysis vs Asymptotic Analysis

❖ Takeaways
Takeaways

- Runtime analysis uses a simplified cost model to represent computation
  - Space analysis also uses a simplified cost model

- Asymptotic analysis can take liberties with mathematical expressions because it deals with infinity
  - Eg, dropping lower-order terms and constants

- Asymptotic analysis gives us a common “frame of reference” with which to compare algorithms
  - “Parabolas grow faster than lines”

- Case analysis is a different axis on which to evaluate runtime and space