

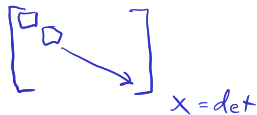
Algorithms and Operation Counting

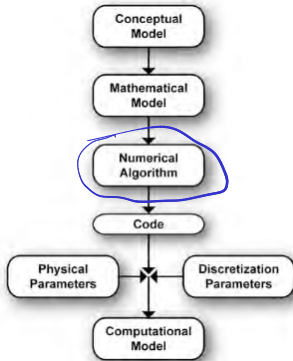
ASEN 3502 — Computational Methods

Please grab a handout in the back!

HW 1 - Due Thursday

Lab min through 1.3
 rec through 1.5





NM-11050-33

Today

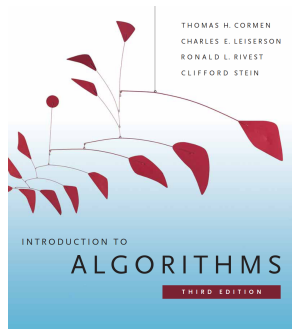
- Algorithms
- Operation Counting
- Asymptotic Notation

$O(n)$

What is an algorithm?

“Informally, an algorithm is any well-defined computational **procedure** that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**.”

— Cormen et al.



Introduction to Algorithms

Cormen, Leiserson,
Rivest, and Stein

mitpress.mit.edu

FLOUR'S FAMOUS BANANA BREAD

I remember grocery shopping with my mom and taking home large bags of overripe bananas when we found them on special for ten cents a pound. My mom would encourage my brother, my dad, and me to "have a banana!" every time we were near the kitchen. My brother and I began to avoid the kitchen for fear of being scolded by Mom and her banana overreliance. In time, I developed this banana bread as a distraction device for us. We loved it so much we sometimes found ourselves buying more bananas just to make it!

INGREDIENTS

- 1 cup (230 grams) unbleached all-purpose flour
- 1 teaspoon baking soda
- 1/2 teaspoon ground cinnamon
- 1/2 teaspoon kosher salt
- 1 cup plus 2 tablespoons (240 grams) sugar
- 2 eggs
- 1 cup (230 grams) ripe or oil
- 20 very ripe, medium banana pieces cut in half (20 cups, peeled/about 340 grams)
- 2 tablespoons olive oil (or any oil)
- 1 teaspoon vanilla extract (or 1/2 cup brown sugar)

BAKE AND 9-INCH LOAF

1. Preheat the oven to 350 degrees F. Butter a 9-by-5-inch loaf pan.

2. In a bowl, sift together the flour, baking soda, cinnamon, and salt. Set aside.

3. Use a stand mixer fitted with the whip attachment (or a hand mixer), beat together the sugar and eggs on medium speed for about 5 minutes, or until light and fluffy. (If you use a handheld mixer, this same step will take about 8 minutes.)

4. On low speed, slowly drizzle in the oil. Don't pour the oil in all at once. Add it slowly so it has time to incorporate into the eggs and doesn't deflate the air you have just beaten into the batter. Adding it should take about 1 minute. Add the bananas, cream, vanilla, and vanilla and continue to mix on low speed just until combined.

5. Using a rubber spatula, fold in the flour mixture and the nuts, if you like. Thoroughly combined. The flour streaks should be visible, and the nuts should be evenly distributed. Pour the batter into the prepared loaf pan and smooth the top.

6. Bake for 1 to 1 1/2 hours, or until golden brown on top and the center springs back when you press it. If your finger sinks when you poke the bread, it needs to bake a little longer. Let cool in the pan on a wire rack for at least 30 minutes, and then pop it out of the pan to finish cooling.

7. The banana bread can be stored tightly wrapped in plastic wrap at room temperature for up to 3 days. Or it can be well wrapped in plastic, wrap and frozen for up to 2 weeks. Thaw overnight at room temperature for serving.

FLOUR'S FAMOUS
BANANA BREAD

Inputs

Procedure

Why algorithms?



- ▶ Algorithms are mathematical objects. Why are they useful?

Why algorithms?

- ▶ Algorithms are mathematical objects. Why are they useful?
 - ▶ We can reason about properties like correctness and resource use

Why algorithms?

- ▶ Algorithms are mathematical objects. Why are they useful?
 - ▶ We can reason about properties like correctness and resource use
 - “An algorithm is said to be **correct** if, for every input instance, it halts with the correct output.” — Cormen et al.

Why algorithms?

- ▶ Algorithms are mathematical objects. Why are they useful?
 - ▶ We can reason about properties like correctness and resource use

“An algorithm is said to be **correct** if, for every input instance, it halts with the correct output.” — Cormen et al.
 - ▶ We can make precise and rigorous statements without having to deal with issues like roundoff error, memory allocation, etc.

How do we represent algorithms?

Matrix-vector multiplication: three ways

English

To get entry r of $\{b\}$, multiply each entry in row r of $[A]$ by the matching entry of $\{x\}$, and add up the results. Do this for every row.

Code

```
b = zeros(n,1);  
for r = 1:n  
    for c = 1:n  
        b(r) = b(r) ...  
            + A(r,c)*x(c);  
    end  
end
```

Pseudocode

```
input      :  $n \times n$  matrix  $[A]$ ,  
              vector  $\{x\}$   
output    :  $\{b\} = [A]\{x\}$   
for  $r \leftarrow 1$  to  $n$  do  
     $b[r] \leftarrow 0$   
    for  $c \leftarrow 1$  to  $n$  do  
         $b[r] \leftarrow b[r] + A[r, c] \cdot x[c]$   
    end  
end
```

“What separates pseudocode from “real” code is that in pseudocode, we employ whatever expressive method is most clear and concise to specify a given algorithm.” — Cormen et al.

Analysis of algorithms

- ▶ *Analyzing* an algorithm means predicting the resources that an algorithm requires
 - ▶ For today, we will count floating point operations (“Flops”): multiplication, division, addition, and subtraction of floating point numbers

Analysis of algorithms

- ▶ *Analyzing* an algorithm means predicting the resources that an algorithm requires
 - ▶ For today, we will count floating point operations (“Flops”): multiplication, division, addition, and subtraction of floating point numbers
- ▶ **Discuss with your neighbor:** Consider the equation $[A]\{x\} = \{b\}$. As n gets larger, which requires more floating point operations? Finding $\{x\}$ given $[A]$ and $\{b\}$, or finding $\{b\}$ given $[A]$ and $\{x\}$?

$$\textcircled{1} \quad b = Ax$$

$$\textcircled{2} \quad Ax = b$$

Harder if using
Gaussian Elimination

Matrix-vector multiplication

input : $n \times n$ matrix $[A]$, vector $\{x\}$

output : $\{b\} = [A]\{x\}$

```
1 for r ← 1 to n do
2   b[r] ← 0
3   for c ← 1 to n do
4     | b[r] ← b[r] + A[r, c] · x[c]
5   end
6 end
```

flops

0
0
2

times

$\sum_{r=1}^n 1 = n$
 $\sum_{r=1}^n \sum_{c=1}^n 1 = n^2$

$$\sum_{r=1}^n \sum_{c=1}^n 1 = \sum_{r=1}^n n = n^2$$

FLOPS: $2n^2$

Gaussian elimination

input : $n \times n$ matrix $[A]$, right-hand side $\{b\}$

output : solution $\{x\}$ of $[A]\{x\} = \{b\}$

// forward elimination

```

1 for c ← 1 to n-1 do
2   for r ← c+1 to n do
3     f ← A[r, c] / A[c, c]
4     for j ← c to n do
5       A[r, j] ← A[r, j] - f · A[c, j]
6     end
7     b[r] ← b[r] - f · b[c]
8   end
9 end

```

// back substitution

```

10 for r ← n, n-1, ..., 1 do
11   s ← b[r]
12   for j ← r+1 to n do
13     s ← s - A[r, j] · x[j]
14   end
15   x[r] ← s / A[r, r]
16 end

```

flops	times
0	$\sum_{i=1}^{n-1} 1 = n-1$
0	$\sum_{i=1}^{n-1} \frac{n^2-n}{2}$
2	$\sum_{i=1}^{n-1} \left\{ \frac{n^3}{3} + O(n^2) \right\}$
2	$\frac{n^2-n}{2}$
0	n
0	n
0	$\sum_{i=1}^{n-1} \frac{n^2-n}{2}$
2	n
1	n

Useful sum identities

$$\sum_{i=1}^m cf(i) = c \sum_{i=1}^m f(i) \quad \left| \quad \sum_{i=1}^m f(i) + g(i) = \sum_{i=1}^m f(i) + \sum_{i=1}^m g(i) \right.$$

$$\sum_{i=1}^m 1 = m$$

$$\rightarrow \sum_{i=1}^m i = \frac{m(m+1)}{2}$$

$$\sum_{i=k}^m 1 = m - k + 1$$

$$\sum_{i=k}^m i^2 = \frac{m(m+1)(2m+1)}{6} - \frac{(k-1)k(2k-1)}{6} = \frac{m^3}{3} + O(m^2)$$

$$\sum_{c=1}^{n-1} \sum_{r=c+1}^n 1 = \sum_{c=1}^{n-1} (n-c) = \sum_{c=1}^{n-1} n - \sum_{c=1}^{n-1} c$$

$$= n(n-1) - \frac{(n-1)n}{2} = \frac{n^2-n}{2}$$

$$\sum_{c=1}^{n-1} \sum_{r=c+1}^n \sum_{j=c}^n 1 = \frac{n^3}{3} - \frac{n}{3}$$

see flop-count.pdf section 2.2

<https://asen3502.com/lectures/020-algorithms/flop-count.pdf>

$$\text{FLOPS: } \frac{n^2-n}{2} + 2\left(\frac{n^3}{3} + O(n^2)\right) + 2\frac{n^2-n}{2}$$

$$\text{FLOPS: } 2\frac{n^3}{3} + O(n^2) + 2\frac{n^2-n}{2} + n$$

function $[C] = \text{matmult}([A], [B])$	flops	times
for $i = 1$ to n	O	n
for $j = 1$ to n	O	n^2
$c_{ij} = 0$	O	$\vdots$
for $k = 1$ to n	O	n^3
$c_{ij} = c_{ij} + \underline{a_{ik}} \underline{b_{kj}}$	2	n^3
end		
end	,	
end		
return $[C]$		
end		

Asymptotic notation

- ▶ Asymptotic notation is a tool for keeping track of only the terms that are most important as a function approaches an asymptote

Asymptotic notation

- ▶ Asymptotic notation is a tool for keeping track of only the terms that are most important as a function approaches an asymptote
- ▶ Chapra: $O(m^n)$ means “terms of order m^n and lower”

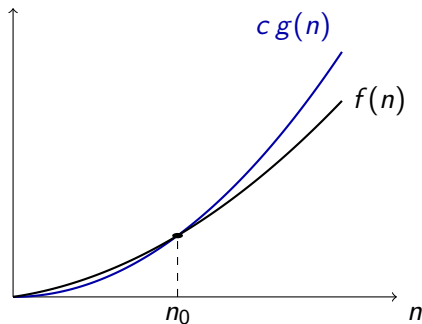
Asymptotic notation

- ▶ Asymptotic notation is a tool for keeping track of only the terms that are most important as a function approaches an asymptote
- ▶ Chapra: $O(m^n)$ means “terms of order m^n and lower”

Definition

$f(n) = O(g(n))$ as $n \rightarrow \infty$ if there exist positive constants c and n_0 such that

$$0 \leq f(n) \leq c g(n) \quad \text{for all } n \geq n_0$$



$$f(n) \leq c g(n) \quad \text{for all } n \geq n_0$$

← $f(n) = 3n^2 + 5n + 2$

$$f(n) = 3n^2 + 5n + 2$$

$$c = 4.0$$

$$n_0 = 6$$

then $0 \leq f(n) \leq c g(n)$
for all $n \geq n_0$

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

$c \geq \text{sum of coefficients}$

$$n_0 = 1$$

$$c = 10$$

$$n_0 = 1$$

Useful Rules

1) $c f(n) = O(f(n)) \quad \forall c > 0$

2) if $f_1(n) = O(n^a)$ and $f_2(n) = O(n^b)$, $f_1(n) + f_2(n) = O(n^{\max(a,b)})$

3) if $f_1(n) = O(n^a)$ and $f_2(n) = O(n^b)$, $f_1(n) \cdot f_2(n) = O(n^{a+b})$

4) if $|f(n)|$ is bounded, $f(n) = O(1)$

Example: $f(n) = \underbrace{n^2}_{O(n^2)} + \underbrace{12n}_{O(n)} \underbrace{\sin(n)}_{O(1)} + \underbrace{50}_{O(1)} = O(n^2)$



Gaussian elimination

input : $n \times n$ matrix $[A]$, right-hand side $\{b\}$

output : solution $\{x\}$ of $[A]\{x\} = \{b\}$

// forward elimination

```
for c ← 1 to n-1 do ←  $\leq n$ 
  for r ← c+1 to n do ←  $\leq n$ 
    f ←  $A[r, c] / A[c, c]$ 
    for j ← c to n do ←  $\leq n$ 
      |  $A[r, j] \leftarrow A[r, j] - f \cdot A[c, j]$ 
    end
     $b[r] \leftarrow b[r] - f \cdot b[c]$ 
  end
end
```

end

// back substitution

```
for r ← n, n-1, ..., 1 do
  s ←  $b[r]$ 
  for j ← r+1 to n do
    |  $s \leftarrow s - A[r, j] \cdot x[j]$ 
  end
   $x[r] \leftarrow s / A[r, r]$ 
end
```

end

flops	Order of times called	
x	}	$O(n)$
x		$O(n^2)$
✓	}	$O(n^3)$
x		$O(n^3)$
✓	}	$O(n^2)$
x		$O(n^2)$
✓	}	$O(n)$
x		$O(n)$
x	}	$O(n)$
x		$O(n)$
✓	}	$O(n^2)$
✓		$O(n)$

$O(n^3)$ Flops