Instructor: Jeff Ames (jca105@cs.rutgers.edu)

TAs:

- Chengguizi Han (chengguizi.han@rutgers.edu)
- Jiancho Ji (jianchao.ji@rutgers.edu)
- Yuwei Jin (yuwei.jin@rutgers.edu)
- Zelong Li (zl359@rutgers.edu)
Recitations

04  Tue 8:25 PM - 9:20 PM, SEC-205 (Yuwei)
05  Thu 8:55 AM - 9:50 AM, BE-250 (Zelong)
06  Thu 8:25 PM - 9:20 PM, SEC-209 (Jianchao)
07  Tue 8:25 PM - 9:20 PM, SEC-207 (Chengguizi)
Homeworks

- 5–6 written homeworks
- 1–2 weeks for each
- small groups are allowed (≤ 3 people)
Textbooks

**DPV**

Sanjoy Dasgupta
Christos Papadimitriou
Umesh Vazirani

**CLRS**

Thomas H. Cormen
Charles E. Leiserson
Ronald L. Rivest
Clifford Stein

*Introduction to Algorithms, Third Edition*
Exams

- Midterm 1: Oct. 17
- Midterm 2: Nov. 14
- Final: Dec. 17 (8 pm – 11 pm)
### Grading (tentative)

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework</td>
<td>20%</td>
</tr>
<tr>
<td>Midterm 1</td>
<td>25%</td>
</tr>
<tr>
<td>Midterm 2</td>
<td>25%</td>
</tr>
<tr>
<td>Final exam</td>
<td>30%</td>
</tr>
</tbody>
</table>
• Big-O notation, asymptotics
• Divide and conquer
• Dynamic programming
• Greedy algorithms
• Sorting
• Graph algorithms

• Linear programming
• Number theoretic algorithms
• Computational geometry
• String matching
• NP-completeness
• Approximation algorithms
Why algorithms?
Fibonacci numbers: \{0, 1, 1, 2, 3, 5, 8, 13, 21, ...\}

\[ F_n = F_{n-1} + F_{n-2} \]

This grows quickly:

\[ F_n \approx 2^{0.649n} \]
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
Three questions:

- Is it correct?
- How much time does it take?
- Can we do better?
For $n \leq 1$, 1 operation.

For $n > 1$,

$$T(n) = T(n - 1) + T(n - 2) + 3$$
\[ T(n) = T(n - 1) + T(n - 2) + 3 \]

\[ F_n = F_{n-1} + F_{n-2} \]

\[ F_n \approx 2^{0.649n} \]
def fib2(n):
    fibs = [0, 1]
    for i in range(2, n+1):
        fibs.append(fibs[i-1] + fibs[i-2])
    return fibs[n]
Sum numbers 1 through $n$:

```
sum = 0
for i in range(1, n+1):
    sum += i
```
• Is it correct?
• How long does it take?
• Can we do better?
Sum numbers 1 through $n$:

$$sum = n \times (n + 1) / 2$$
- Is it correct?
- How long does it take?
- Can we do better?
Base case ($n = 0$):

$$\sum_{i=1}^{0} i = 0 = \frac{0(0 + 1)}{2}$$
Inductive step:
Assume that
\[
\sum_{i=1}^{n-1} i = \frac{(n - 1)((n - 1) + 1)}{2} = \frac{(n - 1)n}{2}
\]
and prove that
\[
\sum_{i=1}^{n} i = \frac{n(n + 1)}{2}
\]
\[
\sum_{i=1}^{n} i = \sum_{i=1}^{n-1} i + n
\]
\[
= \frac{(n - 1)n}{2} + n
\]
\[
= \frac{(n - 1)n + 2n}{2}
\]
\[
= \frac{n^2 - n + 2n}{2}
\]
\[
= \frac{n^2 + n}{2}
\]
\[
= \frac{n(n + 1)}{2}
\]
How long does it take?

- One addition
- One multiplication
- One division
We generally want to abstract out certain details:

- Per-operation cost
- Specialized CPU instructions
- Compiler optimizations
- Memory layout
- Cache effects
This goes from exponential to polynomial, but is it that important? We have Moore’s law...

- How many new Fibonacci numbers could we compute?
- Exponential algorithm $\Rightarrow$ polynomial benefit
- Polynomial algorithm $\Rightarrow$ exponential benefit
We care mainly about the growth of the running time, in terms of the input size, $n$.

- $42n^2 + 3n + 4$
- $7n + 2$
- $2^{0.649n} + n^4 + 2n^2$
And we care mainly about the dominant factor in the running time.

- $42n^2 + 3n + 4 \rightarrow 42n^2$
- $7n + 2 \rightarrow 7n$
- $2^{0.649n} + n^4 + 2n^2 \rightarrow 2^{0.649n}$
Finally, we want to eliminate constant coefficients.

- $42n^2 + 3n + 4 \rightarrow n^2$
- $7n + 2 \rightarrow n$
- $2^{0.649n} + n^4 + 2n^2 \rightarrow 2^n$
Big-O notation:

- $42n^2 + 3n + 4 = O(n^2)$
- $7n + 2 = O(n)$
- $2^{0.649n} + n^4 + 2n^2 = O(2^n)$
Aside: “equality” in big-O notation

- $O(n) = O(n^2)$
- $O(n^2) \neq O(n)$
Aside: “equality” in big-O notation

- $O(n) \in O(n^2)$
- $O(n^2) \notin O(n)$
What does $f(n) = O(g(n))$ mean?

- $g(n)$ is a kind of upper bound on $f(n)$
- for sufficiently large $n$
What does $f(n) = O(g(n))$ mean?

Attempt 1:
For all $n$, $f(n) \leq g(n)$.
What does $f(n) = O(g(n))$ mean?

Attempt 2:
There exists some constant $c$, such that
for all $n$, $f(n) \leq c \cdot g(n)$. 
What does $f(n) = O(g(n))$ mean?

There exists some constants $c$ and $N$ such that, for all $n > N$, $f(n) \leq c \cdot g(n)$.

$\exists c, N \forall n > N, f(n) \leq c \cdot g(n)$
\[ 42n^2 + 3n + 4 = O(n^2) \]

means \( 42n^2 + 3n + 4 \leq c \cdot n^2 \) for \( n > N \).
\[42n^2 + 3n + 4 \leq c \cdot n^2 \text{ for } n > N.\]

Let \( c = 43 \). When is \( 42n^2 + 3n + 4 \leq 43n^2 \)?

\[
42n^2 + 3n + 4 \leq 43n^2 \\
3n + 4 \leq n^2 \\
0 \leq n^2 - 3n - 4
\]
\[ n^2 - 3n - 4 \]
\[ 42n^2 + 3n + 4 = O(n^2) \]

since for all \( n > 4 \),

\[ 42n^2 + 3n + 4 \leq 43n^2 \]
Rules of thumb:

- Constants can be omitted: $7n \rightarrow n$
- Higher exponents dominate: $n^3 + n^2 \rightarrow n^3$
- Exponentials dominate polynomials: $2^n + n^2 \rightarrow 2^n$
- Polynomials dominate logarithms: $n + \log n \rightarrow n$
We also need to consider different inputs, since they may cause the algorithm to perform more or less work.

- Worst case
- Best case
- Average case
Insertion sort: work from left to right, sorting as we go, and “insert” a value where it belongs in the already-sorted region.

- Say we’re looking at element $i$
- Copy $arr[i]$ to a temporary variable
- Find where that value belongs and insert it
Insertion sort:

for each position i:
- walk backwards from i,
  shifting values as you go,
  until you find a value less than a[i]
- put a[i] into this new position
sorted region
Insertion sort:

- Worst case: $O(n^2)$
- Best case: $O(n)$
Big-O notation is similar to a “less-than” relation.
What about “greater-than” or “equal”? 
- $O(\cdot)$: “less-than”
- $\Omega(\cdot)$: “greater-than”
- $\Theta(\cdot)$: “equal”
\( f(n) = \Omega(g(n)) \) means \( g(n) = O(f(n)) \)

\( f(n) = \Theta(g(n)) \) means \( f(n) = O(g(n)) \)
and \( g(n) = O(f(n)) \)
$f(n) = \Theta(g(n))$

$f(n) = O(g(n))$

$f(n) = \Omega(g(n))$
\begin{itemize}
  \item $7n^2 + 3 = O(n^2)$
  \item $7n^2 + 3 = O(n^3)$
  \item $7n^2 + 3 = \Omega(n)$
  \item $7n^2 + 3 = \Theta(n^2)$
\end{itemize}