Welcome to CSci 4041

Algorithms and Data Structures
Instructor (me)

James Parker
Shepherd Labs 391

Primary contact:
jparker@cs.umn.edu
Teaching Assistant

Pariya Babaie, Jayant Gupta, Song Liu, Anoop Shukla, Nikolaos Stefas, Kshitij Tayal, Nitin Varyani
Textbook

Introduction to Algorithms, Cormen et al., 3rd edition
Discussion sections

These will typically reinforce the topics of the week (or exam review)

The TAs may do exercises, so bring something to write on (these exercises will not be graded)
Class website

www.cs.umn.edu/academics/classes
Or google “umn.edu csci class”

Syllabus, schedule, other goodies

Moodle page will have grades and Possibly homework submission
Syllabus

30% Homework
20% Programming assignments
25% Midterm (Oct. 23)
25% Final (Dec. 18)

(No late homework; must ask for extension 48hr before deadline)
Programming vote

C/C++?

Java?

Python?
Syllabus

Grading scale:
93% A
90% A-
87% B+
83% B
80% B-

77% C+
73% C
70% C-
67% D+
60% D
Below F
Schedule

Ch. 1, 2, 3: Introduction
Ch. 2.1, 2.3, 7, 8: Sequences and Sets
Ch. 6, 9, 13, 32: More Sequences and Sets
Ch. 22, 23, 24, 25, 26: Graph Algorithms
Ch. 33: Geometric Algorithms
Ch. 4.2, 30, 31: Algebraic and Numeric Alg.
Ch. 34: NP-Completeness
Any questions?
Course overview

Major topics:
- Learn lots of algorithms
- Decide which algorithm is most appropriate
- Find asymptotic runtime and prove an algorithm works (mathy)
Algorithms

We assume you can program

This class focuses on improving your ability to make code run faster by picking the correct algorithm

This is a crucial skill for large code
Algorithms

We will do a pretty thorough job of sorting algorithms

After that we will touch interesting or important algorithms

The goal is to expose you to a wide range of ways to solve problems
Algorithms

Quite often there is not a single algorithm that always performs best.

Most of the time there are trade-offs: some algorithms are fast, some use more/less memory, some take use parallel computing...
A major point of this class is to tell how scalable algorithms are.

If you have a 2MB input text file and your program runs in 2 min
... what if you input a 5MB file?

... 20 MB file?
Algorithms

In addition to using math to find the speed of algorithms, we will prove algorithms correctly find the answer.

This is called the “correctness” of an algorithm (and often will be proof-by-induction)
Introduction / Review
Moore's Law

Number of transistors double every two years

This trend has slowed a bit, closer to doubling every 2.5 years
First computer

Memory: 1 MB
CPU: 2.4 Mhz
CPU trends

Stock Clock Speed

- 1996: 200
- 1997: 300
- 1998: 400
- 1999: 500
- 2000: 1000
- 2001: 1800
- 2002: 2530
- 2003: 3200
- 2004: 3600
- 2005: 2200
- 2006: 2930
- 2007: 3000
- 2008: 3200
- 2009: 3330
- 2010: 3330
CPU trends

Intel CPU Trends
(sources: Intel, Wikipedia, K. Olukotun)

- Dual-Core Itanium 2
- Pentium 4
- Pentium
- 386

Graph showing trends from 1970 to 2010 with data points for Transistors (000), Clock Speed (MHz), Power (W), and Perf/Clock (ILP).
CPU trends

Intel Processor Clock Speed (MHz)

- Pentium 4 Prescott
- Core 2 Extreme
- Pentium III
- Celeron
- Pentium
- 80486
- 80386
- 80286
- 8080

- Multicore Crisis is Here!

Year:
- 1968
- 1973
- 1979
- 1984
- 1990
- 1995
- 2000
- 2006
CPU trends

Parallel processing (cooking)

You and your siblings are going to make dinner

How would all three of you make... :
(1) turkey?
(2) a salad?
Parallel processing (cooking)

If you make turkey....

preheat
Parallel processing (cooking)

If you make turkey....

preheat
Parallel processing (cooking)

If you make turkey....

put in turkey
Parallel processing (cooking)

If you make turkey....

Put in turkey

Wait a lot
Parallel processing (cooking)

If you make turkey....

take out
Parallel processing (cooking)

If you make turkey....

WAIT

take out

WAIT
Parallel processing (cooking)

If you make a salad...

chop
grate
cut
Parallel processing (cooking)

If you make a salad ...

chop
grate
cut
Parallel processing (cooking)

If you make a salad...

dump together
Parallel processing (cooking)

To make use of last 15 years of technology, need to have algorithms like salad

Multiple cooks need to work at the same time to create the end result

Computers these days have 4-8 “cooks” in them, so try not to make turkey
Correctness

An algorithm is correct if it takes an input and always halts with the correct output.

Many hard problems there is no known correct algorithm and instead approximate algorithms are used.
Asymptotic growth

What does $O(n^2)$ mean?

$\Theta(n^2)$?

$\Omega(n^2)$?
Asymptotic growth

If our algorithm runs in $f(n)$ time, then our algorithm is $O(g(n))$ means there is an $n_0$ and $c$ such that $0 \leq f(n) \leq c \times g(n)$ for all $n \geq n_0$

$O(g(n))$ can be used for more than run time
Asymptotic growth

\( f(n) = O(g(n)) \) means that for large inputs \( n \), \( g(n) \) will not grow slower than \( f(n) \)

\[ n = O(n^2)? \]
\[ n = O(n)? \]
\[ n^2 = O(n)? \]
Asymptotic growth

\( f(n) = O(g(n)) \) gives an upper bound for the growth of \( f(n) \).

\( f(n) = \Omega(g(n)) \) gives a lower bound for the growth of \( f(n) \), namely:
there is an \( n_0 \) and \( c \) such that
\( 0 \leq c \cdot g(n) \leq f(n) \) for all \( n \geq n_0 \).
Asymptotic growth

\( f(n) = \Theta(g(n)) \) is defined as:

there is an \( n_0, c_1 \) and \( c_2 \) such that

\[ 0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \]

for all \( n \geq n_0 \)
Asymptotic growth

Suppose \( f(n) = 2n^2 - 5n + 7 \)

Show \( f(n) = O(n^2) \):
we need to find 'c' and 'n_0' so that
\[
c \cdot n^2 > 2n^2 - 5n + 7,
\]
guess \( c=3 \)
\[
3 \cdot n^2 > 2n^2 - 5n + 7
\]
\[
n^2 > -5n + 7
\]
\[
n > 2, \text{ so } c=3 \text{ and } n_0=2 \text{ proves this}
Asymptotic growth

Suppose \( f(n) = 2n^2 - 5n + 7 \)
Show \( f(n) = \Omega(n^2) \):

For any general \( f(n) \) show:
\( f(n) = \Theta(g(n)) \) if and only if
\( f(n) = O(g(n)) \) and \( f(n) = \Omega(g(n)) \)
Asymptotic growth

Suppose $f(n) = 2n^2 - 5n + 7$
Show $f(n) = \Omega(n^2)$:
again we find a 'c' and 'n_0'
$cn^2 < 2n^2 - 5n + 7$, guess $c=1$
$1 n^2 < 2n^2 - 5n + 7$
$0 < n^2 - 5n + 7$, or $n^2 > 5n - 7$
n > 4, so c=1 and $n_0 = 4$ proves this
Asymptotic growth

$f(n)=\Theta(g(n))$ implies $f(n)=O(g(n))$ and $f(n)=\Omega(g(n))$: by definition we have 'c_1', 'c_2', 'n_0' so

\[
0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \text{ after } n_0
\]

$0 \leq c_1 \cdot g(n) \leq f(n) \text{ after } n_0$ is $\Omega(g(n))$

$0 \leq f(n) \leq c_2 \cdot g(n) \text{ after } n_0$ is $O(g(n))$
Asymptotic growth

\[ f(n) = O(g(n)) \text{ and } f(n) = \Omega(g(n)) \]
implies \[ f(n) = \Theta(g(n)) \] by definition. We have \( c_1, c_2, n_0, n_1 \)

\[ \Omega(g(n)) \text{ is } 0 \leq c_1 g(n) \leq f(n) \text{ after } n_0 \]

\[ O(g(n)) \text{ is } 0 \leq f(n) \leq c_2 g(n) \text{ after } n_1 \]

\[ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ after } \max(n_0, n_1) \]
Asymptotic growth

There are also \( o(g(n)) \) and \( w(g(n)) \) but are rarely used

\( f(n) = o(g(n)) \) means for any \( c \) there is an \( n_0 \) such that
\( 0 \leq f(n) < c \cdot g(n) \) after \( n_0 \)

\[ \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \]

\( w(g(n)) \) is the opposite of \( o(g(n)) \)
Asymptotic growth

Big-O notation is used very frequently to describe run time of algorithms.

It is fairly common to use big-O to bound the worst case and provide empirical evaluation of runtime with data.
Asymptotic growth

What is the running time of the following algorithms for n people:
1. Does anyone share my birthday?
2. Does any two people share a birthday?
3. Does any two people share a birthday (but I can only remember and ask one date at a time)?
Asymptotic growth

1. O(n) or just n
2. O(n) or just n for small n
   (https://en.wikipedia.org/wiki/Birth
   day_problem)
Worst case: 365 (technically 366)
Average run time: 24.61659
3. O(n^2) or n^2
Math review

Monotonically increasing means: for all $m \leq n$ implies $f(m) \leq f(n)$
Math review

Monotonically decreasing means:
for all $m \leq n$ implies $f(m) \geq f(n)$

Strictly increasing means:
for all $m < n$ implies $f(m) < f(n)$

In proving it might be useful to use
monotonicity of $f(n)$ or $d/dn f(n)$
Math review

floor/ceiling?
modulus?
exponential rules and definition?
logs?
factorials?
Floors and ceilings

floor is “round down”
floor(8/3) = 2

ceiling is “round up”
ceiling(8/3) = 3
(both are monotonically increasing)

Prove: floor(n/2) + ceiling(n/2) = n
Floors and ceilings

Prove: \( \text{floor}(n/2) + \text{ceiling}(n/2) = n \)

Case: \( n \) is even, \( n = 2k \)
\[
\text{floor}(2k/2) + \text{ceiling}(2k/2) = 2k \\
k + k = 2k
\]

Case: \( n \) is odd, \( n = 2k+1 \)
\[
\text{floor}((2k+1)/2) + \text{ceiling}((2k+1)/2) \\
\text{floor}(k+1/2) + \text{ceiling}(k+1/2) \\
k + k+1 = 2k + 1
\]
Modulus

Modulus is the remainder of the quotient $a/n$: 
$a \mod n = a - n \lfloor a/n \rfloor$
$7 \mod 2 = 1$
Factorial

\[ n! = 1 \times 2 \times 3 \times \ldots \times n \]

\[ 4! = 4 \times 3 \times 2 \times 1 = 24 \]

Guess the order (low to high):
1,000  1,000,000  1,000,000,000
\[ 2^5 \quad 2^{10} \quad 2^{15} \quad 2^{20} \quad 2^{30} \]
5!  10!  15!  20!
Factorial

The order is (low to high):
{2^5, 5!, (1,000), 2^{10}, 2^{15}, (1,000,000), 2^{20}, 10!, (1,000,000,000), 2^{30}, 15!, 20!}

10! = 3,628,800
15! ≈ 1,307,674,400,000
20! ≈ 2,432,902,000,000,000,000
(2^{10} = 1024 ≈ 1,000 = 10^3)
Find $g(n)$ such that ($g(n) \neq n!$):

1. $n! = \Omega(g(n))$
2. $n! = O(g(n))$
Factorial

1. $n! = \Omega(g(n))$
   - $n! = \Omega(1)$ is a poor answer
   - $n! = \Omega(2^n)$ is decent

2. $n! = O(g(n))$
   - $n! = O(n^n)$
Exponentials

\[(a^n)^m = a^{nm}: \ (2^3)^4 = 8^4 = 4096 = 2^{12}\]

\[a^n a^m = a^{n+m}: \ 2^3 2^4 = 8 \times 16 = 128 = 2^7\]

\[a^0 = 1\]

\[a^1 = a\]

\[a^{-1} = 1/a\]
for all constants: $a > 1$ and $b$: 
\[
\lim_{n \to \infty} \frac{n^b}{a^n} = 0
\]

What does this mean in big-O notation?
Exponentials

What does this mean in big-O notation?

\[ n^b = O(a^n) \] for any \( a > 1 \) and \( b \)
i.e. the exponential of anything eventually grows faster than any polynomials
Exponentials

Sometimes useful facts:

\[ e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} \]

\[ e^x = \lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n \]
Recurrence relationships

Write the first 5 numbers, can you find a pattern:

1. $F_i = F_{i-1} + 2$ with $f_0 = 0$
2. $F_i = 2F_{i-1}$ with $f_0 = 3$
3. $F_i = F_{i-1} + F_{i-2}$, with $f_0 = 0$ and $f_1 = 1$
Recurrence relationships

1. \( F_i = F_{i-1} + 2 \) with \( f_0 = 0 \)
   - \( F_0 = 0, \ F_1 = 2, \ F_2 = 4, \ F_3 = 6, \ F_4 = 8 \)
   - \( F_i = 2i \)

2. \( F_i = 2F_{i-1} \) with \( f_0 = 3 \)
   - \( F_0 = 3, \ F_1 = 6, \ F_2 = 12, \ F_3 = 24, \ F_4 = 48 \)
   - \( F_i = 3 \times 2^i \)
Recurrence relationships

3. \( F_i = F_{i-1} + F_{i-2} \), with \( f_0 = 0 \) and \( f_1 = 1 \)

- \( F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3 \)
- \( F_0 = 5, F_1 = 8, F_2 = 13, F_3 = 21, F_4 = 34 \)

- \( F_i = \frac{\left[(1+\sqrt{5})^i - (1-\sqrt{5})^i\right]}{2^i \sqrt{5}} \)
Recurrence relationships

3. $F_i = F_{i-1} + F_{i-2}$ is homogeneous

We as $F_i = cF_{i-1}$ is exponential, we guess a solution of the form:

$F_i = F_{i-1} + F_{i-2}$, divide by $F_{i-2}$

$F^2 = F + 1$, solve for $F$

$F = \left(1 \pm \sqrt{5}\right)/2$, so have the form

$a\left[(1 + \sqrt{5})/2\right]^i + b\left[(1 - \sqrt{5})/2\right]^i$
Recurrence relationships

\[ a\left(\frac{1 + \sqrt{5}}{2}\right)^i + b\left(\frac{1 - \sqrt{5}}{2}\right)^i \]

with \( F_0 = 0 \) and \( F_1 = 1 \)

2x2 System of equations \( \rightarrow \) solve

\( i=0: \ a[1] + b[1] = 0 \ \rightarrow \ a = -b \)

\( i=1: \ a[1 + \sqrt{5}/2] - a[1 - \sqrt{5}/2] = 1 \)

\[ a[\sqrt{5}] = 1 \]

\( a = 1/\sqrt{5} = -b \)
Recurrence relationships

\[ F_i = 2F_{i-1} - F_{i-2}, \text{ change to exponent} \]
\[ F^i = 2F^{i-1} - F^{i-2}, \text{ divide by } F^{i-2} \]
\[ F^2 = 2F - 1 \rightarrow (F-1)(F-1) = 0 \]
This will have solution of the form:
\[ 1^i + i \times 1^i \]
Next week sorting

- Insert sort
- Merge sort
- Bucket sort
- And more!