INTRODUCTION
today...
- what is this course about?
- why should you care?
- nuts & bolts
- meet the teaching staff
- good coding practice
what is this course about?
fundamentals of coding

- how to analyze your algorithms
  - improve efficiency
  - make good coding choices

-recursion
  *def.* Recursive loop: See “recursive loop”.

-basic sorting algorithms
  - one of most studied operations in CS

-elementary data structures
  - provide mechanism for what we can do with data
why should you care?
You’re traveling to NYC and plan to get around using public transit. Write an algorithm to compute the fastest travel time between any two locations in the city at any time of the day.
You’re traveling to NYC and plan to get around using public transit. Write an algorithm to compute the fastest travel time between any two locations in the city at any time of the day.

Describe an algorithm to efficiently look up a word in a dictionary.
You’re traveling to NYC and plan to get around using public transit. Write an algorithm to compute the fastest travel time between any two locations in the city at any time of the day.

Describe an algorithm to efficiently look up a word in a dictionary.

Given a destination webpage and starting webpage, determine if the destination is reachable via embedded links.
You’re traveling to NYC and plan to get around using public transit. Write an algorithm to compute the fastest travel time between any two locations in the city at any time of the day.

Describe an algorithm to efficiently look up a word in a dictionary.

Given a destination webpage and starting webpage, determine if the destination is reachable via embedded links.

Given a sentence as a string, reverse the characters in each word.
You’re traveling to NYC and plan to get around using public transit. Write an algorithm to compute the fastest travel time between any two locations in the city at any time of the day.

Describe an algorithm to efficiently look up a word in a dictionary.

Given a destination webpage and starting webpage, determine if the destination is reachable via embedded links.

Given a sentence as a string, reverse the characters in each word.
why do(n’t) algorithms matter?

- many different ways to solve a problem
  - one method may take 1ms longer per item....
  - computers operate on LARGE numbers of items
    - millions
    - billions
    - .... or more
why do(n’t) algorithms matter?

- many different ways to solve a problem
  - one method may take 1ms longer per item....
  - computers operate on LARGE numbers of items
    - millions
    - billions
    - ... or more

\[ 1 \times 10^{12} \times \text{(minuscule amount of time)} = \text{large amount of time} \]
why do(n’t) algorithms matter?

-many different ways to solve a problem
  -one method may take 1ms longer per item....
  -computers operate on LARGE numbers of items
    -millions
    -billions
    -... or more

$1 \times 10^{12} \times (\text{minuscule amount of time}) = \text{large amount of time}$

-this matters, but not as much as algorithmic complexity
we refer to unspecified integer quantities as \( N \)

- \( N \) is the problem size
  - sorting an array of \( N \) numbers
  - searching for an item in a set of \( N \) items
  - inserting an item into a set of \( N \) items

amount of work done for these operations usually depends on \( N \)

- work required is a function of \( N \)
why DO algorithms matter?

-algorithms don’t always require $N$ steps for $N$ items
  -could be linear, quadratic, logarithmic, …
  -called the complexity of an algorithm

-$N^2$ is much MUCH bigger than $N$
  -what if $N == 1$ million?

-we only care about large $N$
sort1 versus sort2
Choosing an Algorithm

How important is it to pick the best algorithm for the job?

Small $N$

running time (milliseconds)

N (size of list)

sort 1  |  sort 2

N

14
Choosing an Algorithm

Medium problem size

Running time (milliseconds)

N (size of list)

Sort 1

Sort 2
Choosing an Algorithm (large problem size)

As $N$ becomes large, complexity matters!

![Graph showing running time versus size of list for two sorting algorithms, sort 1 and sort 2. The graph demonstrates that sort 1 has a significantly higher running time compared to sort 2 as the size of the list increases.]
Choosing an Algorithm

As $N$ becomes large, complexity matters!

**Take away:**

as $N$ becomes large, complexity matters!
void sort1(int[] arr) {
    for(int i = 0; i < arr.length-1; i++) {
        int j, minIndex;
        for(j = i+1, minIndex = i; j < arr.length; j++)
            if(arr[j] < arr[minIndex])
                minIndex = j;
        swap(arr, i, minIndex);
    }
}

best case: $O(N^2)$

average case: $O(N^2)$

worst case: $O(N^2)$
void sort1(int[] arr) {
    for(int i = 0; i < arr.length-1; i++) {
        int j, minIndex;
        for(j = i+1, minIndex = i; j < arr.length; j++)
            if(arr[j] < arr[minIndex])
                minIndex = j;
        swap(arr, i, minIndex);
    }
}

best case: O(N^2)
average case: O(N^2)
worst case: O(N^2)

void sort2(int[] arr, int beg, int end) {
    if (end > beg + 1) {
        int piv = arr[beg], l = beg + 1, r = end;
        while (l < r) {
            if (arr[l] <= piv)
                l++;
            else
                swap(arr, l, --r);
        }
        swap(arr, --l, beg);
        sort2(arr, beg, l);
        sort2(arr, r, end);
    }
}

best case: O(N log N)
average case: O(N log N)
worst case: O(N^2)
complexity matters...

the difference between 1ms and 30ms doesn’t matter if you run the algorithm once...
complexity matters...

the difference between 1ms and 30ms doesn’t matter if you run the algorithm once...

... but this is rarely the case in computing

~30ms/frame for all algorithms in a game

~1 billion Google searches per day, every day
data structures & algorithms matter

-for large $N$, the difference between $O(N \log N)$ and $O(N^2)$ is HUGE!
data structures & algorithms matter

-for large $N$, the difference between $O(N \log N)$ and $O(N^2)$ is HUGE!

-is running time the only measure of efficiency?
data structures & algorithms matter

- for large $N$, the difference between $O(N \log N)$ and $O(N^2)$ is HUGE!

-is running time the only measure of efficiency?

-transitioning from cs1410 to cs2420

**cs1410:** correct algorithms (or just code?) to solve problems

**cs2420:** correct algorithms analyzed for efficiency; advanced structures for intuitive organization of data
nuts & bolts
This course provides an introduction to tools found throughout computer science -- basic algorithms and data structures that lend themselves naturally to computational problem solving, and engineering computational efficiency into programs. Students will gain an understanding of classical algorithms (including sorting, searching, tree and graph traversal) and data structures (including linked-lists, trees, graphs, hash tables, and heaps). Students will complete extensive programming assignments that require the implementation and testing of these concepts.

**SCHEDULE**

<table>
<thead>
<tr>
<th>WEEK</th>
<th>DATE</th>
<th>TOPIC</th>
<th>EXAMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/12 &amp; 1/14</td>
<td>Introduction &amp; Java review</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1/19 &amp; 1/21</td>
<td>OO &amp; generic programming</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1/26 &amp; 1/28</td>
<td>Algorithm analysis &amp; data structures</td>
<td>Midterm 1 on Tues</td>
</tr>
<tr>
<td>4</td>
<td>2/2 &amp; 2/4</td>
<td>Basic sorting</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2/9 &amp; 2/11</td>
<td>Recursive sorting</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2/16 &amp; 2/18</td>
<td>Linked lists</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2/23 &amp; 2/25</td>
<td>Stacks &amp; queues</td>
<td></td>
</tr>
</tbody>
</table>
- programming homework
  - one assignment per week
    - *out Thursdays, due next Thursday at midnight*
  - must be done with a partner, except this week
  - 1-day late policy, 10% off
  - all programming to be done in Java 8

-exams
  - two midterms, held during class time
  - one final

-labs
  - will focus on technical issues
  - **required**: they count towards your final grade!
- assignments
- grades
- student-to-student discussion forum
- announcements
tough love slide...
tough love slide...

- this is the gate-keeper course for the CS major
  - getting [A || A-] means likely getting to major status
  - getting [B+ || B] is a gray area
  - getting [<= B-] is a no-go
tough love slide...

- this is the gate-keeper course for the CS major
  - getting [A || A-] means likely getting to major status
  - getting [B+ || B] is a gray area
  - getting [<= B-] is a no-go

-historically...
  - 25% of students get A’s
  - 25% of students get B’s
  - 33% of students drop or fail
meet the teaching staff
this week...
- **reading**
  - chapters 1 & 2

- **homework**
  - proficiency exam *(do not hand in)*
  - assignment 1 out on Thursday

- **no lab**
  - in-class mini-lab on Thursday