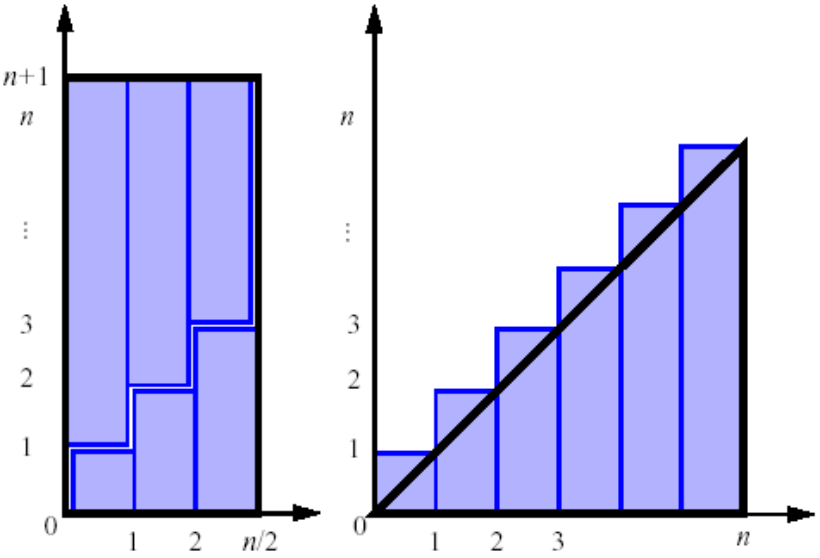
Pick up an in-class quiz from the table near the door

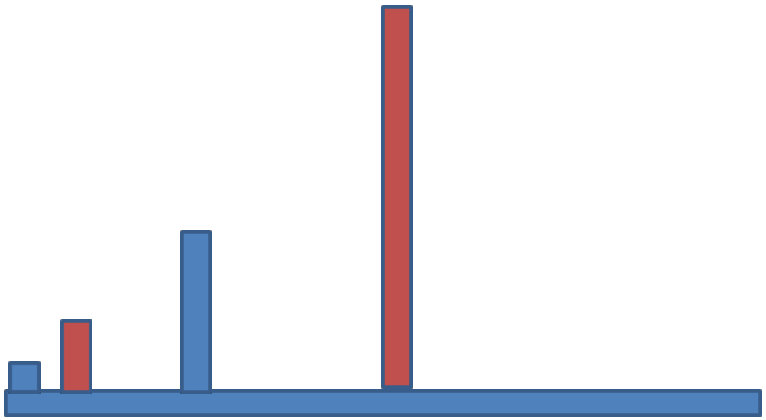
CSSE 230 Data Structures and Algorithm Analysis Day 1

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n = \frac{n^2 + n}{2}$$

- two visual representations



Brief Course Intro
Math Review
Growable Array Analysis



Student Introductions

- Roll call
  - Introduce yourself to the person next to you
  - I'll soon post an assignment to Moodle that asks you to share more with classmates on a Piazza discussion forum, e.g., what's your favorite food, what are your hobbies, types of work you've done, etc.

And intro to daily quizzes, worth 5% of grade: Q1

Introductions

- Joe Hollingsworth, aka Dr. Holly
  - B.S. Indiana University, CS
  - M.S. Purdue University, CS
  - Ph.D. Ohio State University, Software Engineering
  - Special interests in formal methods, software design, how to best teach computing
  - Courses taught at Rose:
    - CSSE220, CSSE230, CSSE373, Senior Project (479)
  - Hobbies: cycling, running, swimming, learning Spanish, travel

Goal: independently design, develop, and debug software that uses correct, clear, and efficient algorithms and data structures

Prove: An AVL Tree has O(log n) height
Proof: By definition,
$| height(T_L) - height(T_R) | \leq 1$
…

Topic	I do	You do	You practice	You perform on
Analysis	Explain, show, do	Listen, follow, read, quiz, challenge	Homework sets	Tests
Programming			Major programs	Tests, project

```
/**
 * A height-balanced binary tree with rank
 * that could be the basis for a text
 * editor.
 * @author Claude Anderson and Matt Boutell.
*/
public class EditTree {
    private Node root;
    private int rotationCount = 0;
    private Node singleLeftRotation(
            Node grandParent, Node parent) {
            // Set parent nodes
…
```

Why *efficient* algorithms?

Here's $1,000,000,000 in $100 bills:

- Find serial number KB46279860I
- If unsorted, you could look at all 10 million bills.
- If sorted by serial number, binary search finds it by only looking at _____ bills.

https://commons.wikimedia.org/wiki/File:Oenbilliondollar.jpg

$\log_2(N) = \log_{10}(N)/\log_{10}(2)$

- If you have a sorted array
- Then use binary search to find something

Performance:

- # instructions executed to find any item
- Size of problem = N
- # instructions for binary search is: $\log_2(N)$
- log2(10,000,000)

How to succeed in CSSE230

- Work hard
  - Re-do CSSE220 stuff as needed to make sure your foundations (recursion and linked lists) are strong
- Take initiative in learning
  - Read the text, search Javadocs, post questions to Piazza, come for help
- Focus while in this class
  - https://www.rose-hulman.edu/class/cs/csse230/201820/MiscDocuments/LaptopsAreGreatButNotDuringaLectureoraMeeting.pdf (11/26/2017 NYT)
  - Make it Stick
- Start early and plan for no all-nighters
  - 2 assignments/week: 1 homework set and 1 major program
- Never give or use someone else's answers

How to Start Early?

Tools

- Moodle Site: https://moodle.rose-hulman.edu/course/view.php?id=49906
  - schedule, reading/HW/program assignments, room #s!
  - Read the **Syllabus**: Tomorrow's quiz will start with questions about it.
  - gradebook, homework pdf turn-in, peer evaluations, solutions

- www.piazza.com, not email: homework questions and announcements
  - If you email me, I'll reply, "Great question! Please post it to Piazza"
  - It should auto-email you whenever there is a post.

- moodle.rose-hulman.edu: gradebook, homework pdf turn-in, peer evaluations, solutions

After today's class, you will be able to…

- analyze runtimes of code snippets by counting instructions.
- explain why arrays need to grow as data is added.
- derive the average and worst case time to insert an item into an array [GrowableArray exercise]

Analysis/Math Review

Notation

- Floor

$$\lfloor x \rfloor = \text{the largest integer} \leq x$$

- Ceiling

$$\lceil x \rceil = \text{the smallest integer} \geq x$$

- **`java.lang.Math`, provides the static methods `floor()` and `ceil()`**

Summations

- Summations
  - general definition:

$$\sum_{i=s}^{t} f(i) = f(s) + f(s+1) + f(s+2) + \ldots + f(t)$$

  - where f is a function, s is the start index, and t is the end index

Geometric progressions: each term is a constant multiple of the previous term

- Geometric progression: $f(i) = a^i$
  - given an integer $n \geq 0$ and a real number $0 < a \neq 1$

$$\sum_{i=0}^{n} a^i = 1 + a + a^2 + \ldots + a^n = \frac{1 - a^{n+1}}{1 - a}$$

Memorize this formula!

  - geometric progressions exhibit exponential growth

Exercise: What is $\sum_{i=2}^{6} 3^i$?

The sum can also be written:

$$\frac{a^{n+1} - 1}{a - 1}$$

This will be useful for today's Growable Arrays exercise!

Summations

Open form

- Problem: lots of applications of additions to come up with the answer
- Advantage: brainless to compute

Closed form

- Problem – often takes some insight to determine closed form
- Advantage: brainless to compute and requires few applications of math operations

Arithmetic progressions: constant difference
Most important to us: a difference of 1

- Arithmetic progressions:
  - An example

Memorize this formula!

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n = \frac{n^2 + n}{2}$$

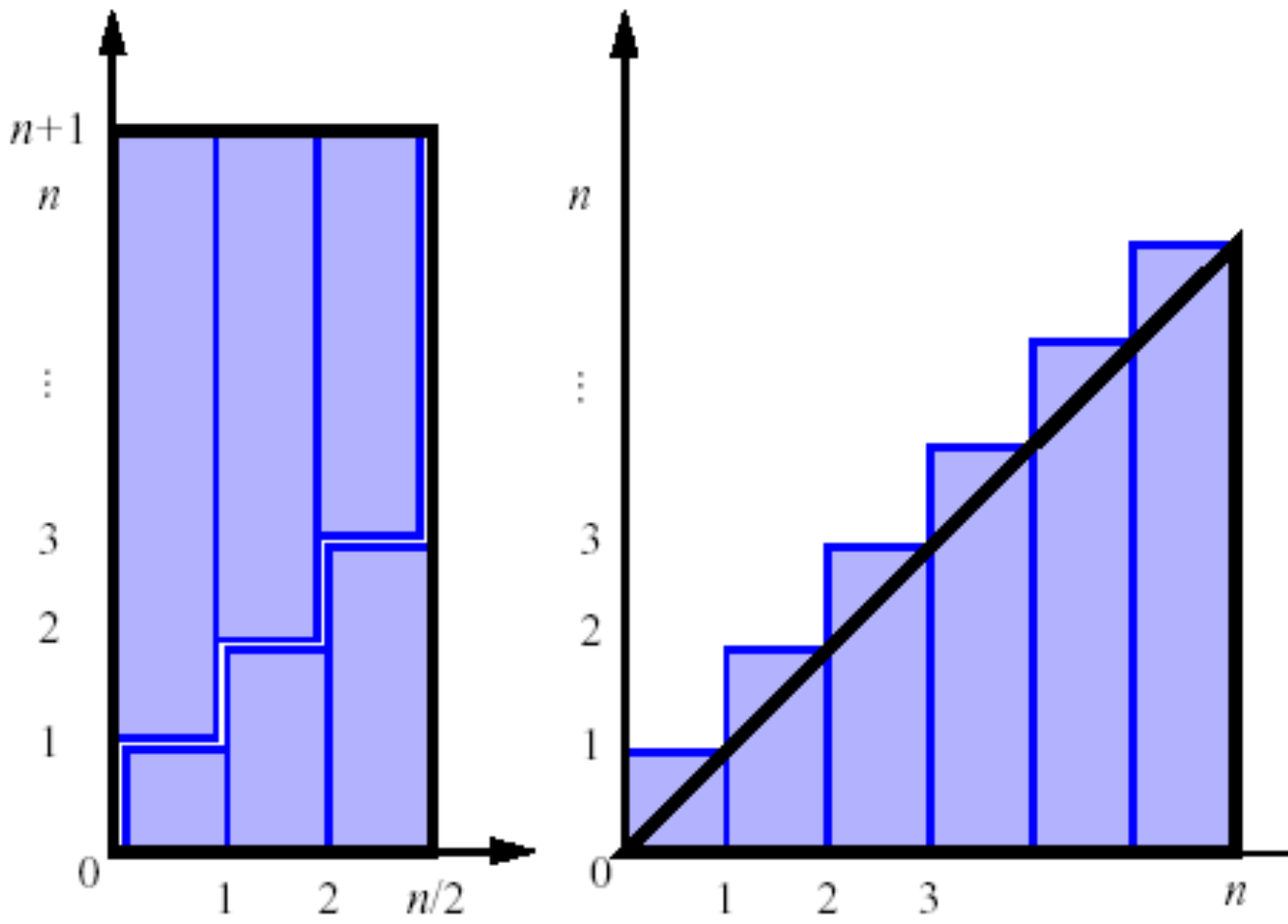
Exercise: $\sum_{i=21}^{40} i$

Also useful for today's Growable Arrays exercise!

Visual proofs of the summation formula

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n = \frac{n^2 + n}{2}$$

- two visual representations

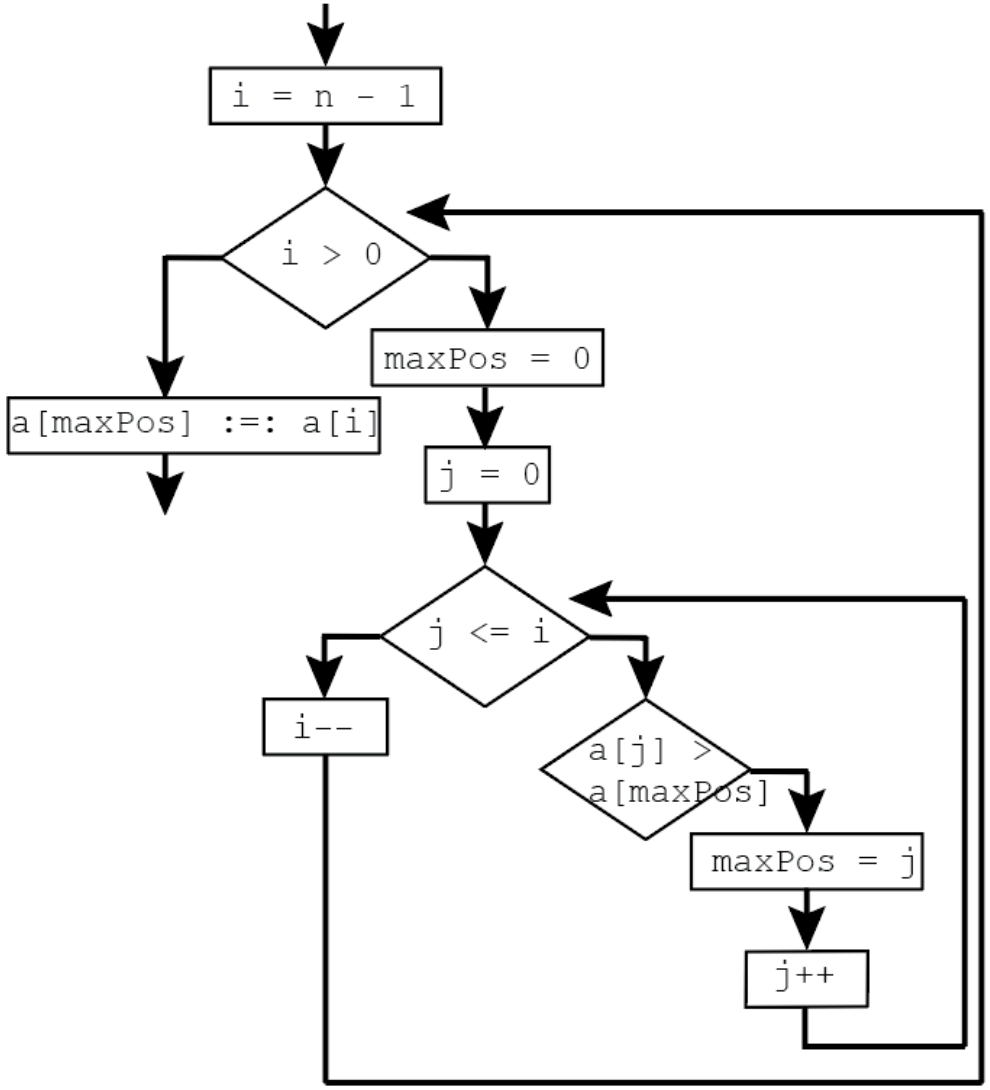


Application: Runtime of Selection Sort

```
1 for (int i = n-1; i > 0; i--) {
2    int maxPos = 0;
3    for (int j = 0; j <= i; j++) {
4        if (a[j] > a[maxPos]) {
5            maxPos = j;
6        }
7    }
8    swap a[maxPos] with a[i] ;
9 }
```

Diagram of Selection Sort

```
1 for (int i = n-1; i > 0; i--) {
2   int maxPos = 0;
3   for (int j = 0; j <= i; j++) {
4       if (a[j] > a[maxPos]) {
5           maxPos = j;
6       } // end if
7   } // end for
8   swap a[maxPos] with a[i] ;
9 } // end for
```



Selection Sort

- Basic idea:
  - Think of the array as having a sorted part (at the end) and an unsorted part (the beginning)

0	1	2	3	4	5	6	7	8	9
7	388	310	438	79	10	121	537	974	1391

  - Find the *largest* value in the unsorted part
  - Move it to the *beginning* of the sorted part (making the sorted part bigger and the unsorted part smaller)

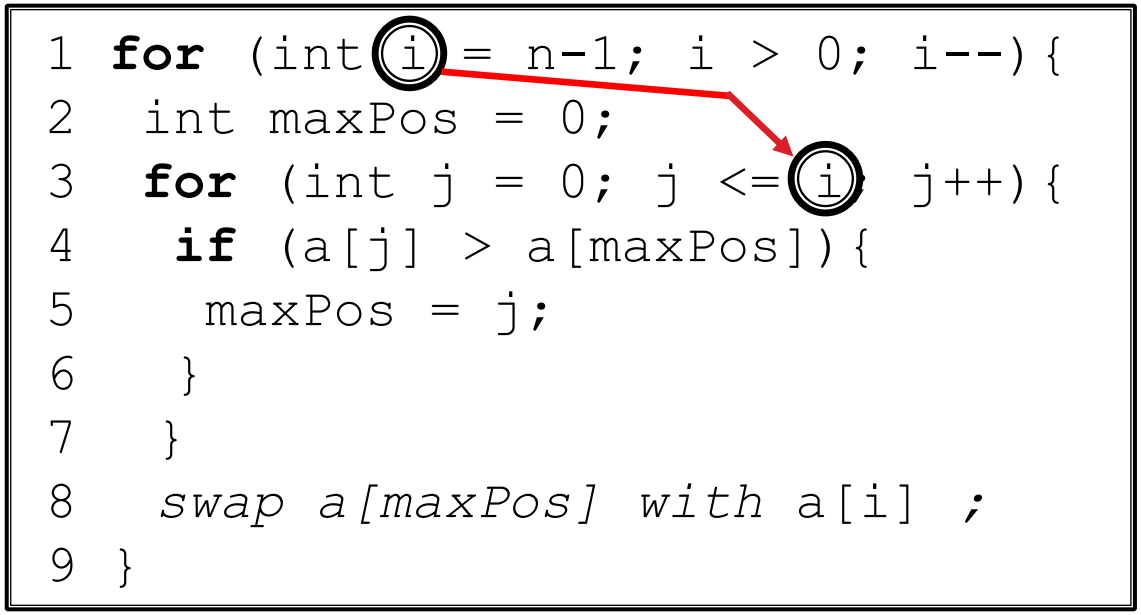
Repeat until unsorted part is empty

Application: Find exact and big-Oh runtime of Selection Sort

```
1 for (int i = n-1; i > 0; i--) {
2     int maxPos = 0;
3     for (int j = 0; j <= i; j++) {
4         if (a[j] > a[maxPos]) {
5             maxPos = j;
6         }
7     }
8     swap a[maxPos] with a[i] ;
9 }
```

- On what line is comparison performed?
- How many comparisons of array elements are executed? Exact? Big-Oh?
- How many times are array elements copied?

When to use the "table method" for analyzing performance



```
1 for (int i = n-1; i > 0; i--){
2  int maxPos = 0;
3  for (int j = 0; j <= i; j++){
4   if (a[j] > a[maxPos]){
5    maxPos = j;
6   }
7  }
8  swap a[maxPos] with a[i] ;
9 }
```

Rule: Use the "table method" when the inner loop's upper bound depends on the outer loop's loop counter

Example:

- In the code above *for* loop at line #3 has an upper bound that depends on variable *i*. Variable *i* is the loop counter from line #1's *for* loop
- So in this case, the "table method" should be used for the analysis

When to use the "table method" for analyzing performance

```
1 for (int i = n-1; i > 0; i--){
2  int maxPos = 0;
3  for (int j = 0; j <= i; j++){
4   if (a[j] > a[maxPos]){
5    maxPos = j;
6   }
7   }
8   swap a[maxPos] with a[i] ;
9 }
```

i	j	count
n – 1	0,1,2, …, n -1	n
n – 2	…	…
…		
1		

1. Draw the table
2. Label the columns
   - the first two columns use the *for* loop counter variables
   - the last column is labeled with *count*
3. Fill in the first column
4. Fill in the second column
5. Determine the count for each row

Growable Array Analysis

An exercise in doubling, done by pairs of students

Arrays are ubiquitous

- Basis for ArrayLists, sorting, and hash tables
- Why? O(1) access to any position, regardless of the size of the array.

- Limitation of ArrayLists:
  - Fixed capacity!
  - If it fills, you need to re-allocate memory and copy items
    - How efficient is this?
    - Consider two schemes: “add 1” and “double”

Work on Growable Array Exercise

- Work with a partner
- Hand in the document before you leave today if possible. Otherwise due start of day 2's class.
- Get help as needed from me and the assistants.

Handy for Growable Arrays HW

Properties of logarithms

$$log_b(xy) = log_b(x) + log_b(y)$$

$$log_b\left(x/y\right) = log_b(x) - log_b(y)$$

$$log_b(x^\alpha) = \alpha log_b(x)$$

$$log_b(x) = \frac{log_a(x)}{log_a(b)}$$

$$a^{log_b(n)} = n^{log_b(a)}$$

Properties of exponents

$$a^{(b+c)} = a^b a^c$$

$$a^{bc} = \left(a^b\right)^c$$

$$a^b/_{a^c} = a^{(b-c)}$$

$$b = a^{log_a(b)}$$

$$b^c = a^{c*log_a(b)}$$