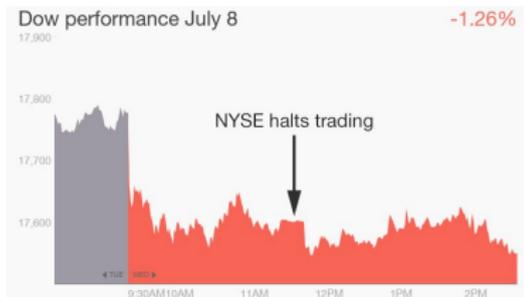


# CS 310: Prelude

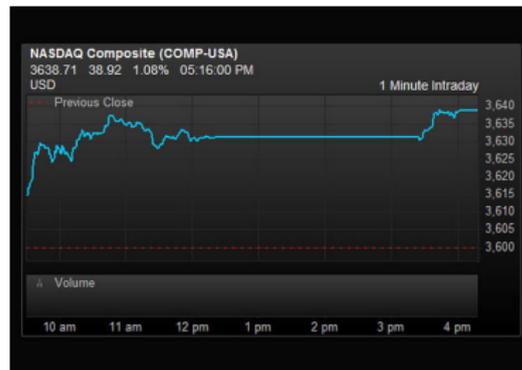
Chris Kauffman

Week 1-1

# Noteworthy



Trading resumes on NYSE after nearly 4-hour outage, CNN 7/8/2015



Pricing Problem Suspends Nasdaq for Three Hours, NYT 8/22/2013

# Make Some Money



## News

Relevance | Date

- L** Apple Inc. (AAPL), Google Inc (GOOG): Two iPhones Are Better Than One  
Insider Monkey - Aug 21, 2013
- M** Google Inc (GOOG), Yahoo! Inc. (YHOO), Unilever plc (ADR) (UL): The First Big...  
Insider Monkey - Aug 19, 2013
- N** Buzz Stocks: Facebook Inc (FB), PG&E Corporation (PCG), Google Inc (GOOG) ...  
Schaeffers Research - Aug 21, 2013
- O** Tech News: Apple Inc. (AAPL)'s Rivals' Effort, Google Inc (GOOG)...  
Insider Monkey - Aug 19, 2013
- P** Is Google Inc (GOOG) Going Too Far with this Form of Advertising?  
Insider Monkey - Aug 20, 2013

You get hired by an investment firm (cha-ching).

First task: analyze historical stock performances to locate good times to buy and sell.

- ▶ Buy low and Sell high
- ▶ Or don't play at all

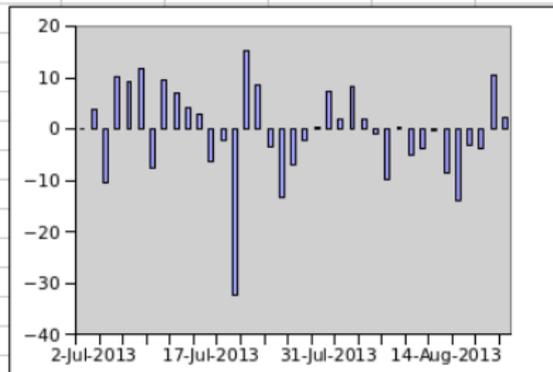
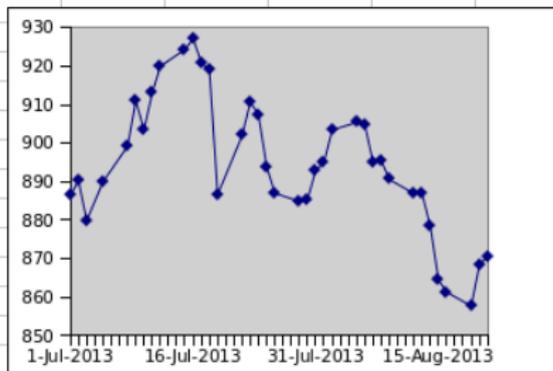
# Many Options



Don't play: 0 gain

# The Best Buy

|    |             |     |     |
|----|-------------|-----|-----|
| 1  | 1-Jul-2013  | 886 | 0   |
| 2  | 2-Jul-2013  | 890 | 4   |
| 3  | 3-Jul-2013  | 880 | -10 |
| 4  | 5-Jul-2013  | 890 | 10  |
| 5  | 8-Jul-2013  | 899 | 9   |
| 6  | 9-Jul-2013  | 911 | 12  |
| 7  | 10-Jul-2013 | 903 | -8  |
| 8  | 11-Jul-2013 | 913 | 10  |
| 9  | 12-Jul-2013 | 920 | 7   |
| 10 | 15-Jul-2013 | 924 | 4   |
| 11 | 16-Jul-2013 | 927 | 3   |
| 12 | 17-Jul-2013 | 921 | -6  |
| 13 | 18-Jul-2013 | 919 | -2  |
| 14 | 19-Jul-2013 | 887 | -32 |
| 15 | 22-Jul-2013 | 902 | 15  |
| 16 | 23-Jul-2013 | 911 | 9   |
| 17 | 24-Jul-2013 | 907 | -3  |
| 18 | 25-Jul-2013 | 894 | -13 |
| 19 | 26-Jul-2013 | 887 | -7  |
| 20 | 29-Jul-2013 | 885 | -2  |
| 21 | 30-Jul-2013 | 885 | 1   |
| 22 | 31-Jul-2013 | 893 | 8   |
| 23 | 1-Aug-2013  | 895 | 2   |
| 24 | 2-Aug-2013  | 903 | 8   |
| 25 | 5-Aug-2013  | 905 | 2   |
| 26 | 6-Aug-2013  | 905 | -1  |
| 27 | 7-Aug-2013  | 895 | -10 |
| 28 | 8-Aug-2013  | 895 | 0   |



# How Would you find Best Increase?

| i  | price | delta |
|----|-------|-------|
| 1  | 886   | 0     |
| 2  | 890   | 4     |
| 3  | 880   | -10   |
| 4  | 890   | 10    |
| 5  | 899   | 9     |
| 6  | 911   | 12    |
| 7  | 903   | -8    |
| 8  | 913   | 10    |
| 9  | 920   | 7     |
| 10 | 924   | 4     |
| 11 | 927   | 3     |
| 12 | 921   | -6    |
| 13 | 919   | -2    |
| 14 | 887   | -32   |
| 15 | 902   | 15    |

How is payoff computed for  
start=5 and end=12?  
For start=7 and end=10?

## Several names for the Problem

- ▶ Maximum contiguous subsequence sum (text)
- ▶ **Maximum Subarray** (wikipedia)
- ▶ Find start and end time with largest payoff out of all possible

## Find a Solution

- ▶ **Input** is the array `delta[]`
- ▶ **Output**: (start, end, payoff) such that payoff is as large as possible
- ▶ Can optionally *not invest* for no payoff; return (-1,-1,0)

## Algorithm 1: Brute Force

```
maxSubsequenceCube(int A[]){
    bestPayoff = 0
    bestStart = -1
    bestEnd = -1
    for start=0 to A.length-1 {
        for end=start to A.length-1 {
            currentPayoff = 0
            for i=start to end {
                currentPayoff += A[i]
            }
            if(currentPayoff > bestPayoff){
                bestPayoff = currentPayoff
                bestStart = start
                bestEnd = end
            }
        }
    }
    return bestPayoff, bestStart, bestEnd
}
```

- ▶ A[] contains deltas
- ▶ Try every possible start and end (outer loops)
- ▶ Calculate increase from start to end
- ▶ Track the best seen
- ▶ Complexity?
- ▶ Anything better

## Algorithm 2: Skip the inner loop

```
maxSubsequenceQuad(int A[]){
    bestPayoff = 0
    bestStart = -1
    bestEnd = -1
    for start=0 to A.length-1 {
        currentPayoff = 0
        for end=start to A.length-1 {
            currentPayoff += A[end]
            if(current > best){
                bestPayoff = currentPayoff
                bestStart = start
                bestEnd = end
            }
        }
    }
    return bestPayoff, bestStart, bestEnd
}
```

- ▶ Try every start and end
- ▶ Don't recalculate currentPayoff in a loop
- ▶ 'Remember' last currentPayoff as end changes

## Algorithm 2 Alternative: Convert to global Prices

```
maxSubsequenceQuad(int A[]){
    B = new array size A.length
    B[0] = A[0]
    for i=1 to B.length-1
        B[i] = B[i-1] + A[i]

    best = 0
    bestStart = -1
    bestEnd = -1
    for start=0 to A.length-1 {
        for end=start to A.length-1 {
            current = B[end] - B[start]
            if(current > best){
                best = current
                bestStart = start
                bestEnd = end
            }
        }
    }
    return best, bestStart, bestEnd
}
```

- ▶ Initially convert deltas in A to global prices in B
- ▶ First price doesn't matter as interested in changes
- ▶ Try every start and end
- ▶ Easy to calculate currentPayoff
- ▶ Memory overhead?

## Anything Better?

- ▶ `maxSubsequenceCube()`: most straight-forward enumeration of all possible solutions
- ▶ `maxSubsequenceQuad()`: used a trick to speed up enumeration

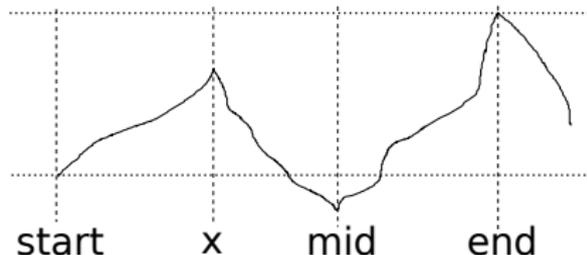
Increasing speed now calls for some deeper insight

## A Helpful Property

**Proposition:** The shortest maximum subsequence beginning at `start` and finishing at `end` contains no point `mid` between them with a lower value than `start`.

**Proof by Contradiction:**

- ▶ Suppose shortest max subsequence exists, looks like picture.
- ▶ `x` must be lower than `end`, o/w could form a shorter maximum subsequence `start` to `x`
- ▶ But if `mid` is lower than `start`, sequence `mid` to `end` has a larger increase than `start` to `end`.



**Consequence:** If `mid` drops below `start`, reset `start` to `mid`. Create a **faster** algorithm based on this property.

*Contradiction* □

## Algorithm 3: Scan

```
maxSubsequenceLinear(int A[]){
    best = 0
    current = 0
    bestStart = -1
    bestEnd = -1
    start = 0
    for end=0 to A.length-1 {
        current += A[end]
        if(current > best){
            best = current
            bestStart = start
            bestEnd = end
        }
        else if(current < 0){
            start = end+1;
            current = 0;
        }
    }
    return best,bestStart,bestEnd;
}
```

- ▶ A[] contains deltas
- ▶ When sum current falls below zero, move start to end and reset
- ▶ Single pass over entire array

# Max Subsequence Algorithms Synopsis

## Comparisons

- ▶ `maxSubsequenceCube()`: triply nested loops over entire array,  $O(N^3)$
- ▶ `maxSubsequenceQuad()`: doubly nested loops over entire array,  $O(N^2)$
- ▶ `maxSubsequenceLinear()`: single loop over entire array,  $O(N)$

Intuition: for large arrays, `maxSubsequenceLinear()` will produce answers faster

## Demonstration

This happens in practice, see `MaxSumTestBetter.java` for implementations with timing.

# Course Synopsis

- ▶ Look at problems
- ▶ Identify solutions
- ▶ Evaluate solution for its "goodness"
  - ▶ What metrics of goodness exist for code?
  - ▶ Which metrics are most important
- ▶ Most solutions will involve an algorithm and a data structure
  - ▶ What's an algorithm
  - ▶ What's a data structure

# Syllabus and Schedule

Both linked on Piazza, tons of info on

- ▶ Grading
- ▶ Assignment submission
- ▶ Policies (late work, etc.)
- ▶ Schedule of events

Highlights to follow. . .

## Preconditions

This is a 3rd programming class.

- ▶ CS 211 Prereq
- ▶ Know Java
- ▶ You have easy access to a computer with java

Not sure if you're ready?

- ▶ Review first chapters of Weiss for Java refresher, should mostly be stuff you already know
- ▶ Inspect past CS 211 projects: could you solve them in given times?

<https://cs.gmu.edu/~kauffman/cs211/p3.html> (7 days)

<https://cs.gmu.edu/~kauffman/cs211/p6.html> (10-14 days)

<https://cs.gmu.edu/~kauffman/cs211/p7.html> (5 days)

# Cheating

## Don't cheat

- ▶ Easy to catch
- ▶ Pain for you
- ▶ Pain for me (makes me ornery)
- ▶ If you don't get caught, you'll still suck at programming

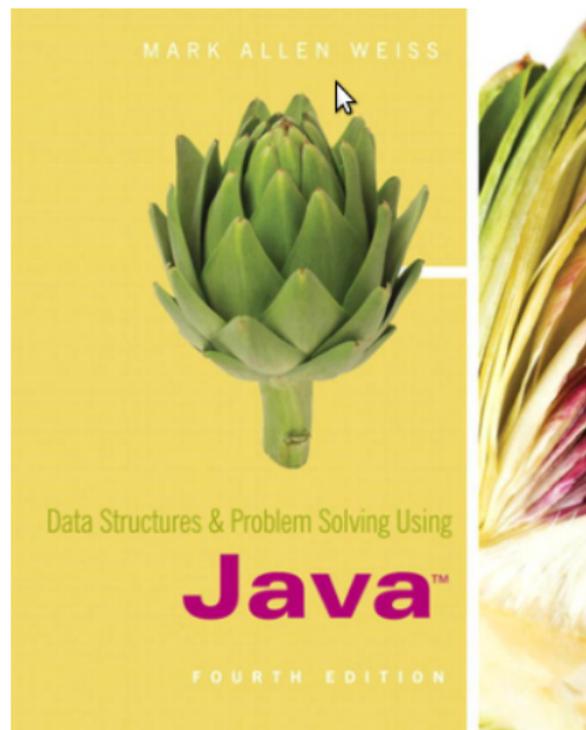
Cooperation is not automatically cheating.

- ▶ Examples

# Hot Seats

- ▶ Each session, first few rows are hot seats
- ▶ First come, first serve (adjust if needed)
- ▶ Just try: answer questions, give feedback
- ▶ Don't want/need participation, sit elsewhere
- ▶ Up to 3% overall bonus
  - ▶ Luke and Leia have 20 part pts, max in class, 3% bonus each
  - ▶ Han and Chewie have 10 part pts, 1.5% bonus each
  - ▶ Greedo has 0 part pts, 0.0% bonus
- ▶ Scoring described in **Syllabus**
- ▶ Participation is **only** opportunity for extra credit
- ▶ May be a few other opportunities for participation

# Textbook



Weiss is pretty good

- ▶ I'll assume you're reading it
- ▶ Likely want to get the text [source code](#)
- ▶ On 2-hour reserve in Johnson Center Library, tell them the course number

## We're on Piazza

Should all have received an invitation to join the Piazza class  
(piazza.com)

- ▶ Discussion
- ▶ Announcements
- ▶ Schedule

Blackboard only for

- ▶ Assignment submission
- ▶ Grades

95% of the time you should post, not email

Mail me for

- ▶ Personal appointments
- ▶ Unresolvable grading disputes

# Your Teaching Team and Office Hours

See Piazza Staff Section

- ▶ Kauffman Plans Office Hours Tue 3-5pm (OK?)
- ▶ Remaining course staff will have office hours posted on Piazza by week's end

| Name             | Email                     |      |
|------------------|---------------------------|------|
| Chris Kauffman   | kauffman@cs.gmu.edu       | Prof |
| Dave Nordstrom   | dnordstr@cs.gmu.edu       | Prof |
| Richard Carver   | rcarver@cs.gmu.edu        | Prof |
| Yujing Chen      | ychen37@gmu.edu           | GTA  |
| Guilin Liu       | gliu2@gmu.edu             | GTA  |
| Hui Zheng        | hzheng5@gmu.edu           | GTA  |
| Veda Bhamidipati | vbhamid2@gmu.edu          | GTA  |
| Claire Cecil     | ccecil2@masonlive.gmu.edu | UTA  |
| Daniel Lacher    | dlacher@masonlive.gmu.edu | UTA  |

# Tools

The official java tools of the course are

- ▶ [jdk 1.8](#), official build and run tools from Oracle
- ▶ [DrJava](#), a simple, superior java IDE (if you're into IDEs)

Minor support given for (though not official)

- ▶ [jGrasp](#), a decent IDE with drawing capabilities, used for some in-class examples

Special Note:

- ▶ *I do not know how to use eclipse*
- ▶ *I will not be learning how this semester.*
- ▶ *If I can help it I will never learn eclipse.*
- ▶ *TAs may be able to help you but are **not** required to do so.*
- ▶ *In class I will use jGrasp, Emacs, and command line.*
- ▶ *If you have questions on those I'm happy to help.*

## Special Note on DrJava

We've made some improvements at GMU

- ▶ Better test result printing
- ▶ Fixed debugger activation bug
- ▶ Unofficial, trying to get into main distrib
- ▶ **Strongly** encourage DrJava users to grab this version
- ▶ Download here: <https://cs.gmu.edu/~kauffman/drjava/>

# Slides

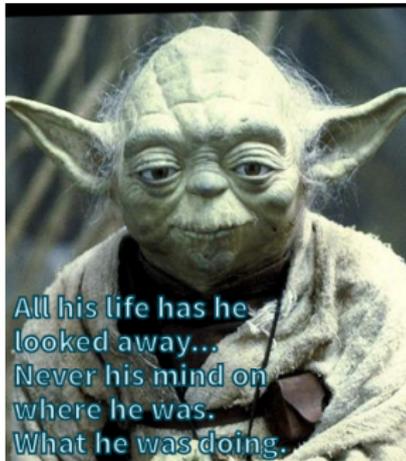
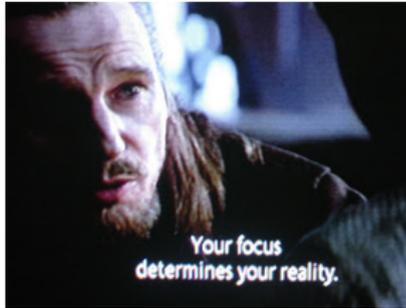
- ▶ Will try to make slides available before class
- ▶ Slides always available sometime after class
- ▶ Slides are not much good without accompanying conversation
- ▶ Code examples posted after class
- ▶ Link to slide page: [Pizza/Resources](#)

# Programming Assignments

4 of them during the semester

- ▶ 35% of your grade
- ▶ Medium-large implementations using data structures
- ▶ Grading in two parts
  - ▶ Automated junit test cases
  - ▶ Manual GTA inspection for quality
- ▶ Submit to blackboard, 11:59 p.m. ????
- ▶ *What day should programming assignments be due?*

# Focus



## A Study

The students in the first experiment who were asked to multitask [during lecture] averaged 11 per cent lower on their quiz.

The students in the second experiment who were surrounded by laptops scored 17 per cent lower.

*Laptop use lowers student grades, experiment shows, The Canadian Press, 8-14-2013*

# Effective Procrastination

- ▶ Adam Grant: Can Slowing Down Help You Be More Creative?
  - ▶ Start something early (Milestones)
  - ▶ Then take a break
  - ▶ Then finish strong (Deadline)
- ▶ Tim Urban: What Happens In The Brain Of An Extreme Procrastinator?

Early



Later....



# Logistics

## At Home

- ▶ Read Weiss Ch 1-4: Java Review
- ▶ Read Weiss Ch 5: Big-O
- ▶ Get your java environment set up

## Goals Today

- ▶ More Course Mechanics
- ▶ Basic understanding of Big O and friends
- ▶ This Chapter 5 material