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
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
<table>
<thead>
<tr>
<th>Date</th>
<th>Value</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Jul-2013</td>
<td>886</td>
<td>0</td>
</tr>
<tr>
<td>2-Jul-2013</td>
<td>890</td>
<td>4</td>
</tr>
<tr>
<td>3-Jul-2013</td>
<td>880</td>
<td>-10</td>
</tr>
<tr>
<td>5-Jul-2013</td>
<td>890</td>
<td>10</td>
</tr>
<tr>
<td>8-Jul-2013</td>
<td>899</td>
<td>9</td>
</tr>
<tr>
<td>9-Jul-2013</td>
<td>911</td>
<td>12</td>
</tr>
<tr>
<td>10-Jul-2013</td>
<td>903</td>
<td>-8</td>
</tr>
<tr>
<td>11-Jul-2013</td>
<td>913</td>
<td>10</td>
</tr>
<tr>
<td>12-Jul-2013</td>
<td>920</td>
<td>7</td>
</tr>
<tr>
<td>15-Jul-2013</td>
<td>924</td>
<td>4</td>
</tr>
<tr>
<td>16-Jul-2013</td>
<td>927</td>
<td>3</td>
</tr>
<tr>
<td>17-Jul-2013</td>
<td>921</td>
<td>-6</td>
</tr>
<tr>
<td>18-Jul-2013</td>
<td>919</td>
<td>-2</td>
</tr>
<tr>
<td>19-Jul-2013</td>
<td>887</td>
<td>-32</td>
</tr>
<tr>
<td>22-Jul-2013</td>
<td>902</td>
<td>15</td>
</tr>
<tr>
<td>23-Jul-2013</td>
<td>911</td>
<td>9</td>
</tr>
<tr>
<td>24-Jul-2013</td>
<td>907</td>
<td>-3</td>
</tr>
<tr>
<td>25-Jul-2013</td>
<td>894</td>
<td>-13</td>
</tr>
<tr>
<td>26-Jul-2013</td>
<td>887</td>
<td>-7</td>
</tr>
<tr>
<td>29-Jul-2013</td>
<td>885</td>
<td>-2</td>
</tr>
<tr>
<td>30-Jul-2013</td>
<td>885</td>
<td>1</td>
</tr>
<tr>
<td>31-Jul-2013</td>
<td>893</td>
<td>8</td>
</tr>
<tr>
<td>1-Aug-2013</td>
<td>895</td>
<td>2</td>
</tr>
<tr>
<td>2-Aug-2013</td>
<td>903</td>
<td>8</td>
</tr>
<tr>
<td>5-Aug-2013</td>
<td>905</td>
<td>2</td>
</tr>
<tr>
<td>6-Aug-2013</td>
<td>905</td>
<td>-1</td>
</tr>
<tr>
<td>7-Aug-2013</td>
<td>895</td>
<td>-10</td>
</tr>
<tr>
<td>8-Aug-2013</td>
<td>895</td>
<td>0</td>
</tr>
</tbody>
</table>
How Would you find Best Increase?

<table>
<thead>
<tr>
<th>price</th>
<th>i</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>886</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>890</td>
<td>1</td>
<td>-10</td>
</tr>
<tr>
<td>880</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>899</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>911</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>903</td>
<td>5</td>
<td>-8</td>
</tr>
<tr>
<td>913</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>920</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>924</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>927</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>921</td>
<td>10</td>
<td>-6</td>
</tr>
<tr>
<td>919</td>
<td>11</td>
<td>-2</td>
</tr>
<tr>
<td>887</td>
<td>12</td>
<td>-32</td>
</tr>
<tr>
<td>902</td>
<td>13</td>
<td>15</td>
</tr>
</tbody>
</table>

Several names for the Problem

- Maximum contiguous subsequence sum (text)
- Maximum Subarray (wikip)
- 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)

How is payoff computed for start=1 and end=3?

For start=6 and end=10?
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
}
Algorithm 2: Skip the inner loop

```c
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

    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 then 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

```java
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, \(O(N)\)

\(N\): size of the array of deltas

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)
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 discussed
Hot Seats

- Each session, first few rows are hot seats
- First come, first serve (adjust if needed)
- Don’t want/need participation, sit elsewhere
- Just try: answer questions, give feedback, get cards
- Return and count cards at end of each session
- Up to 3% overall bonus
  - Luke and Leia have 20 cards, max in class, 3% bonus each
  - Han and Chewie have 10 cards, 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
Weiss is pretty good

- I’ll assume you’re reading it
- Likely want to get the text source code
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chris Kauffman</td>
<td><a href="mailto:kauffman@cs.gmu.edu">kauffman@cs.gmu.edu</a></td>
<td>Prof</td>
</tr>
<tr>
<td>Fardin Alam</td>
<td><a href="mailto:falam5@masonlive.gmu.edu">falam5@masonlive.gmu.edu</a></td>
<td>GTA</td>
</tr>
</tbody>
</table>
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 probably don’t know how to use IDE X and won’t be learning this semester
- TAs may be able to help you but are not required to do so.
- In class I will use Emacs, command line, DrJava, JGrasp.
- 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

3-4 of them during the semester

- 35% of your grade
- Medium-large implementations using data structures
- Grading in three parts
  - Milestone JUnit test cases
  - Automated JUnit test cases
  - Manual GTA inspection for quality
- Submit to blackboard, 11:59 p.m. Saturdays
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 (Milestone Deadline)
  - Then take a break
  - Then finish strong (Final Deadline)
- Tim Urban: What Happens In The Brain Of An Extreme Procrastinator?

Early

This is a perfect time to get some work done.

Nope!

Later...

AAAAAAAHAAAAH
HHHHHHHHHHHH
HHHHHHHHHHHHH

AAAAAAAHAAAAH
HHHHHHHH

AAAAAAAHAAAAH
HHHHHHHH

waitbutwhy.com
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