CS 310: Prelude

Chris Kauffman

Week 1-1
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

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
Don’t play: 0 gain
The Best Buy

<table>
<thead>
<tr>
<th>Date</th>
<th>Value 1</th>
<th>Value 2</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>4 Jul 2013</td>
<td>890</td>
<td>10</td>
</tr>
<tr>
<td>5 Jul 2013</td>
<td>899</td>
<td>9</td>
</tr>
<tr>
<td>6 Jul 2013</td>
<td>911</td>
<td>12</td>
</tr>
<tr>
<td>7 Jul 2013</td>
<td>903</td>
<td>-8</td>
</tr>
<tr>
<td>8 Jul 2013</td>
<td>913</td>
<td>10</td>
</tr>
<tr>
<td>9 Jul 2013</td>
<td>920</td>
<td>7</td>
</tr>
<tr>
<td>10 Jul 2013</td>
<td>924</td>
<td>4</td>
</tr>
<tr>
<td>11 Jul 2013</td>
<td>927</td>
<td>3</td>
</tr>
<tr>
<td>12 Jul 2013</td>
<td>921</td>
<td>-6</td>
</tr>
<tr>
<td>13 Jul 2013</td>
<td>919</td>
<td>-2</td>
</tr>
<tr>
<td>14 Jul 2013</td>
<td>887</td>
<td>-32</td>
</tr>
<tr>
<td>15 Jul 2013</td>
<td>902</td>
<td>15</td>
</tr>
<tr>
<td>16 Jul 2013</td>
<td>911</td>
<td>9</td>
</tr>
<tr>
<td>17 Jul 2013</td>
<td>907</td>
<td>-3</td>
</tr>
<tr>
<td>18 Jul 2013</td>
<td>894</td>
<td>-13</td>
</tr>
<tr>
<td>19 Jul 2013</td>
<td>887</td>
<td>-7</td>
</tr>
<tr>
<td>20 Jul 2013</td>
<td>885</td>
<td>-2</td>
</tr>
<tr>
<td>21 Jul 2013</td>
<td>885</td>
<td>1</td>
</tr>
<tr>
<td>22 Jul 2013</td>
<td>893</td>
<td>8</td>
</tr>
<tr>
<td>23 Jul 2013</td>
<td>895</td>
<td>2</td>
</tr>
<tr>
<td>24 Jul 2013</td>
<td>903</td>
<td>8</td>
</tr>
<tr>
<td>25 Jul 2013</td>
<td>905</td>
<td>2</td>
</tr>
<tr>
<td>26 Jul 2013</td>
<td>905</td>
<td>-1</td>
</tr>
<tr>
<td>27 Jul 2013</td>
<td>895</td>
<td>-10</td>
</tr>
<tr>
<td>28 Jul 2013</td>
<td>895</td>
<td>0</td>
</tr>
</tbody>
</table>
How Would you find Best Increase?

<table>
<thead>
<tr>
<th>i</th>
<th>price</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>886</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>890</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>880</td>
<td>-10</td>
</tr>
<tr>
<td>4</td>
<td>890</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>899</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>911</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>903</td>
<td>-8</td>
</tr>
<tr>
<td>8</td>
<td>913</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>920</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>924</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>927</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>921</td>
<td>-6</td>
</tr>
<tr>
<td>13</td>
<td>919</td>
<td>-2</td>
</tr>
<tr>
<td>14</td>
<td>887</td>
<td>-32</td>
</tr>
<tr>
<td>15</td>
<td>902</td>
<td>15</td>
</tr>
</tbody>
</table>

Problem Names

Several names

- Maximum contiguous subsequence sum problem (text)
- Maximum Subarray Problem (wikip)
- Find start and end time with best increase out of all possible

Several solutions

- Brute force, simple speed trick,
  Scan solution
- Find me a solution: returns (increase, start, end)
Algorithm 1: Brute Force

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

- Assumptions:
  - Assumes A[] contains deltas
- Method:
  - 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

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

- Try every start and end
- Don’t recalculate current in a loop
- ’Remember’ last current as end changes
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

maxSubsequenceLinear(int A[]){
    best = 0
    current = 0
    bestStart = -1
    bestEnd = -1
    start = 1
    for end=1:A.length {
        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? 10-14 days?

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
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

See Piazza Staff Section
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/
For Next Time

- Weiss Ch 1-4: Review
- Weiss Ch 5: Big-O (It’s Showtime!)
- Get your java environment set up