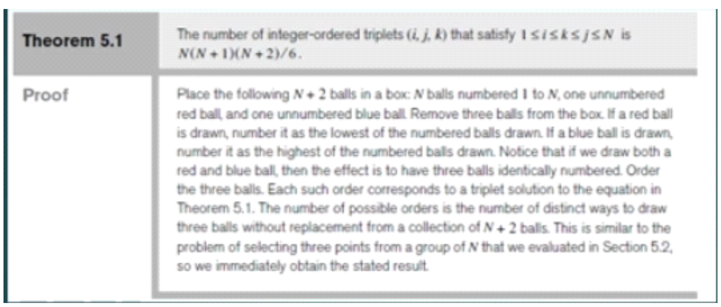
Chapter 5

Friday, July 7, 2023 4:35 PM

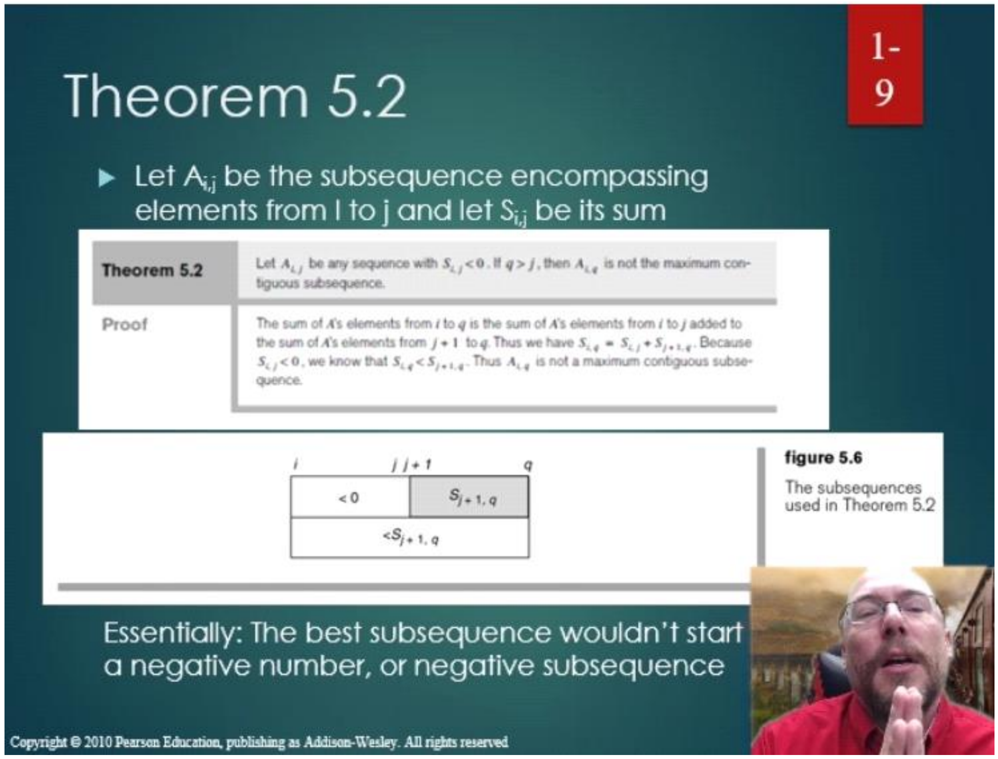
- Timing your java code
  - Algorithm
    - Cleary specified set of instructions that a computer follows that solves a problem correctly.
    - We want to analyze this with time
  - Runtime
    - This is what we measure in. There are a lot of things that factor into it. But it really is just time measurement
  - Mean
    - What the code currently does. Add up all the times and divide by the total
    - Outliers really mess this up
  - Mode
    - Value that occurs most often
    - All times are varied so that does not work
  - Median
    - Middle number. We can put times into an array, sort the array, and report the middle number.
  - Timing in code
    - Long startTime = System.nanoTime();
    - Long endTime = System.nanoTime();
    - Long totalTime = endTime - startTime;
    - System.out.println(totalTime);
  - Excel
    - Paste dropdown and use text import wizard, Delimited, Comma, and finish
    - This works with commas in between each value
    - Make sure to delete unwanted columns
- Big Oh Notation
  - Input
    - Timing mainly has to do with input and how much time increase is caused from input growth
    - Look for common graphs (Cubic, Quadratic, N log N, Linear)
  - Linear function
    - Like downloading a file over the internet. Twice as large the file takes twice the amount of time.
  - Cubic
    - Cubic function is a function whose dominant term is some

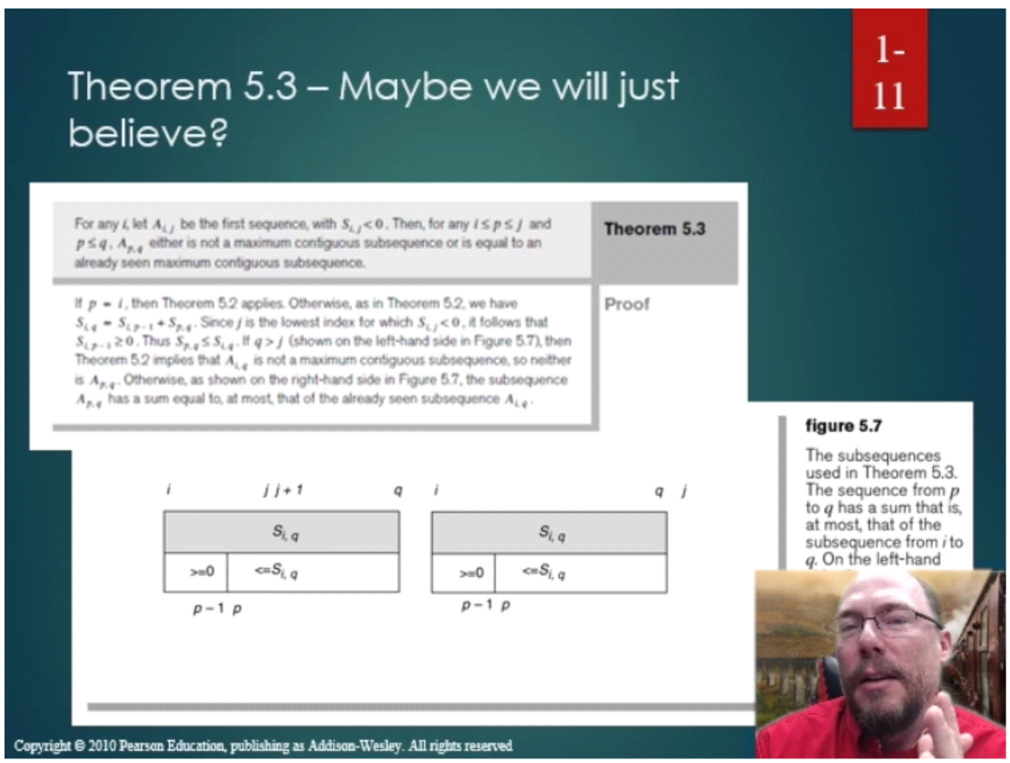
constant times N^3

- Quadratic N^2
- Logarithm
  - The expression O(N log N) Represents a function whose dominant term is N times the logarithm of N
  - Slowly growing function. For instance, the logarithm of 1,000,000 is only 20. The logarithm grows slower than a square or cube root.
- Big-Oh
  - Just means that we talk about the largest value in the function
- If we touch each index 1 time its linear, 2 its quadratic, 3 its cubic.
- Maximum contiguous subsequence sum problem
  - We have an array with numbers, we are trying to find which subsequence has the max volume.
  - Example 1: [-2, 11, -4, 13, -5, 2, -6]
    - Answer: -20
  - How would you code this?
- Brute force
  - Use 3 for loops and all nested and one goes from 0 to the end. O(N^3)
- Theorem 5.1
  - 

Theorem 5.1	The number of integer-ordered triplets (i, j, k) that satisfy $1 \le i \le k \le j \le N$ is $N(N+1)(N+2)/6$.
Proof	Place the following $N+2$ balls in a box: N balls numbered 1 to N, one unnumbered red ball, and one unnumbered blue ball. Remove three balls from the box. If a red ball is drawn, number it as the lowest of the numbered balls drawn. If a blue ball is drawn, number it as the highest of the numbered balls drawn. Notice that if we draw both a red and blue ball, then the effect is to have three balls identically numbered. Order the three balls. Each such order corresponds to a triplet solution to the equation in Theorem 5.1. The number of possible orders is the number of distinct ways to draw three balls without replacement from a collection of $N+2$ balls. This is similar to the problem of selecting three points from a group of N that we evaluated in Section 5.2, so we immediately obtain the stated result.

- Theorem 5.2

- 

  - Theorem 5.3
    - 

- General Big-oh rules
  - Big-omega
    - Upper bound, it might run this fast but will most likely be slower
  - Big-Theta
    - Upper and lower bound.
  - ^3 ^2 Nlog(N) N log(N)
- The Logarithm
  - Binary numbers grow at a logarithmic rate

- Log is how many times you can cut something in half
- Doubling or halving is the log