

- ▶ Proof by Contradiction. Suppose there is such a MaxCSS, namely $S_{p,q}$, where $i+1 \leq p \leq j$.

i $S_{i,j}$ just became negative! j

Case 1. $q > j$

p MaxCSS q

Case 2. $q \leq j$

p MaxCSS q

CSSE 230 Day 4

Maximum Contiguous Subsequence Sum

After today's class you will be able to:

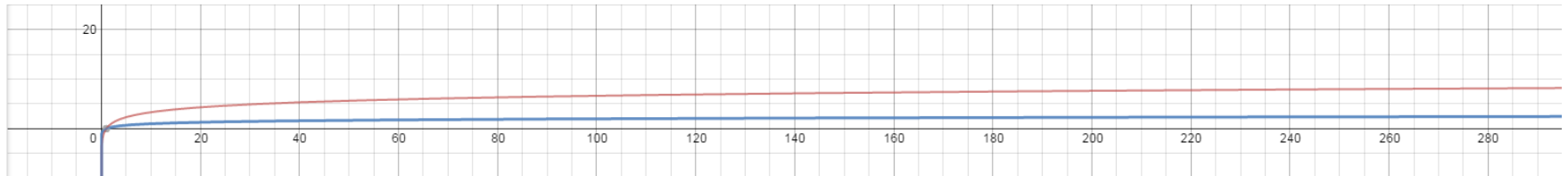
provide an example where an insightful algorithm can be much more efficient than a naive one.

Announcements

- ▶ Sit with your StacksAndQueues partner now
- ▶ Why Math?
- ▶ HW2
 - Recommended: complete #1 (UML Diagram of Collections) by next class, as we will discuss Collections
 - You should be ready to work on all of the problems

Homework 2

- ▶ Is it true that $\log_a(n)$ is $\theta(\log_b(n))$?
- ▶ Complete homework 2 to find out the exciting conclusion!
- ▶ Here is the graph for $a=2$ and $b=10$:



- ▶ Is it true that 3^n is $\theta(2^n)$?



Andrew Hettlinger ► **Matt Boutell**

November 6 at 12:30pm · 🌐

In your class, I never thought I'd actually use big O notation, but now I find myself using it in my complaints to coworkers about how a previous developer would sort a list before doing a binary search to find a single element $O(n \log n) + O(\log n)$ instead of just doing a linear search $O(n)$. I feel really nerdy now (as if I didn't before 😊)

Like · Comment

So why would we ever sort first to do binary search?

Recap: MCSS

Problem definition: Given a non-empty sequence of n (possibly negative) integers A_1, A_2, \dots, A_n , find the maximum consecutive subsequence $S_{i,j} = \sum_{k=i}^j A_k$, and the corresponding values of i and j .

Reminder: we use 0-based indexing.

Recap: Eliminate the most obvious inefficiency, get $\Theta(N^2)$

```
for( int i = 0; i < a.length; i++ ) {  
    int thisSum = 0;  
    for( int j = i; j < a.length; j++ ) {  
        thisSum += a[ j ];  
  
        if( thisSum > maxSum ) {  
            maxSum = thisSum;  
            seqStart = i;  
            seqEnd    = j;  
        }  
    }  
}
```

- ▶ Exhaustive search: find every $S_{i,j}$

MCSS is $O(n^2)$

▶ Is MCSS $\theta(n^2)$?

- Showing that a problem is $\Omega(g(n))$ is much tougher. How do you prove that it is impossible to solve a problem more quickly than you already can?
- Can we find a yet faster algorithm?
 - If so, it can't use exhaustive search. (Why?)

$f(n)$ is $O(g(n))$ if $f(n) \leq cg(n)$ for all $n \geq n_0$

- So O gives an upper bound

$f(n)$ is $\Omega(g(n))$ if $f(n) \geq cg(n)$ for all $n \geq n_0$

- So Ω gives a lower bound

$f(n)$ is $\theta(g(n))$ if $c_1g(n) \leq f(n) \leq c_2g(n)$ for all $n \geq n_0$

- So θ gives a tight bound

- $f(n)$ is $\theta(g(n))$ if it is both $O(g(n))$ and $\Omega(g(n))$

Observation 1

- ▶ We noted that a max-sum sequence $S_{i,j}$ cannot begin with a negative number.
- ▶ Generalizing this, it cannot begin with a prefix $S_{i,k}$ with $k < j$ whose sum is negative.
 - **Proof by contradiction.** Suppose that $S_{i,j}$ is a max-sum sequence and that $S_{i,k}$ is negative. In that case, a larger-sum contiguous sequence can be created by removing $S_{i,k}$. However, this violates our assumption that $S_{i,j}$ is a max-sum contiguous sequence.

Observation 2

- ▶ Every contiguous subsequence that borders a maximum contiguous subsequence must have a negative or zero sum.
 - **Proof by contradiction.** Consider a contiguous subsequence that borders an MCSS sequence. Suppose it has a positive sum. We can then create a larger max-sum sequence by combining the two adjacent sequences. This contradicts our assumption that the original sequence has the maximum sum.

Observation 3

- ▶ Imagine we are growing subsequences from a fixed left index i . That is, we compute the sums $S_{i,j}$ for increasing j .
- ▶ Claim: If there is such an $S_{i,j}$ that “just became negative” (for the first time, with the inclusion of the j^{th} term), any subsequence starting in between $i + 1$ and j cannot be a MaxCSS (unless its sum equals an already-found MaxCSS)!
- ▶ In other words, as soon as we find that $S_{i,j}$ is negative, we can skip all sums that begin with any of A_{i+1}, \dots, A_j .
- ▶ I.e., we can “skip i ahead” to be $j + 1$.

Proof of Observation 3

- ▶ Proof by Contradiction. Suppose there is such a MaxCSS, namely $S_{p,q}$, where $i+1 \leq p \leq j$.

i $S_{i,j}$ just became negative! j

- ▶ Key point. What must be true of the following sums?

$S_{i,p-1} \geq 0$ | $S_{p,j} < 0$ $S_{i,j}$ just became negative!

Case 1. $q > j$

p MaxCSS q

Starts with a negative prefix. Violates Obs. 1!

Case 2. $q \leq j$

p MaxCSS q

Borders a subsequence with nonnegative sum. Violates Obs. 2, or there is a previous MaxCSS with the same sum.

New, improved code!

```

public static Result mcSSLinear(int[] seq) {
    Result result = new Result();
    result.sum = 0;
    int thisSum = 0;

    int i = 0;
    for (int j = 0; j < seq.length; j++) {
        thisSum += seq[j];

        if (thisSum > result.sum) {
            result.sum = thisSum;
            result.startIndex = i;
            result.endIndex = j;
        } else if (thisSum < 0) {
            // advances start to where end
            // will be on NEXT iteration
            i = j + 1;
            thisSum = 0;
        }
    }
    return result;
}

```

$S_{i,j}$ is negative. So, skip ahead per Observation 3

Running time is $O(?)$
How do we know?

What have we shown?

- ▶ MCSS is $O(n)$!
- ▶ Is MCSS $\Omega(n)$ and thus $\theta(n)$?
 - Yes, intuitively: we must at least examine all n elements

Time Trials!

- ▶ From personal repo, checkout **MCSSRaces**
- ▶ Study code in **MCSS.main()**
- ▶ For each algorithm, **how large a sequence** can you process on your machine **in less than 1 second?**

MCSS Conclusions

- ▶ The first algorithm we think of may be a lot worse than the best one for a problem
- ▶ Sometimes we need clever ideas to improve it
- ▶ Showing that the faster code is correct can require some serious thinking
- ▶ Programming is more about careful consideration than fast typing!

Interlude

- ▶ If GM had kept up with technology like the computer industry has, we would all be driving \$25 cars that got 1000 miles to the gallon.
 - Bill Gates

- ▶ If the automobile had followed the same development cycle as the computer, a Rolls-Royce would today cost \$100, get a million miles per gallon, and explode once a year, killing everyone inside.
 - Robert X. Cringely

Interlude



Stacks and Queues

A preview of Abstract Data
Types and Java Collections

This week's major program

Stacks and Queues assignment

Intro: Ideas for how to implement stacks and queues using arrays and linked lists

How to write your own growable circular queue:

1. Grow it as needed (like day 1 exercise)
2. Wrap-around the array indices for more efficient dequeuing

Stacks and Queues implementation

Analyze implementation choices for Queues – much more interesting than stacks! (See HW)

Application: An exercise in writing cool algorithms that evaluate mathematical expressions:

Evaluate Postfix: $6\ 7\ 8\ *\ +$
(62. How?)

Convert Infix to Postfix: $6\ +\ 7\ *\ 8$
($6\ 7\ 8\ *\ +$ You'll figure out how)

Both using **stacks**.

Read assignment for hints on *how*.

Meet your partner

- ▶ Plan when you'll be working. We suggest that your first meeting should be today or tomorrow
- ▶ Review the pair programming video as needed
- ▶ Check out the code and read the specification together