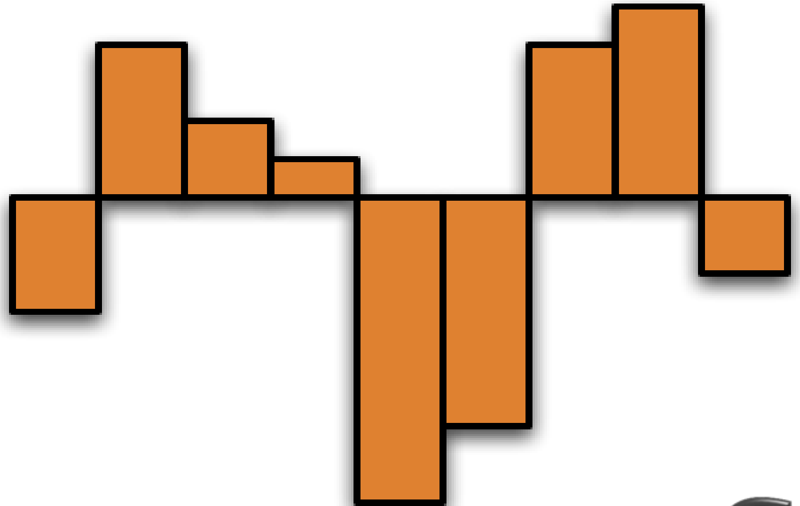


CSSE 230 Day 3

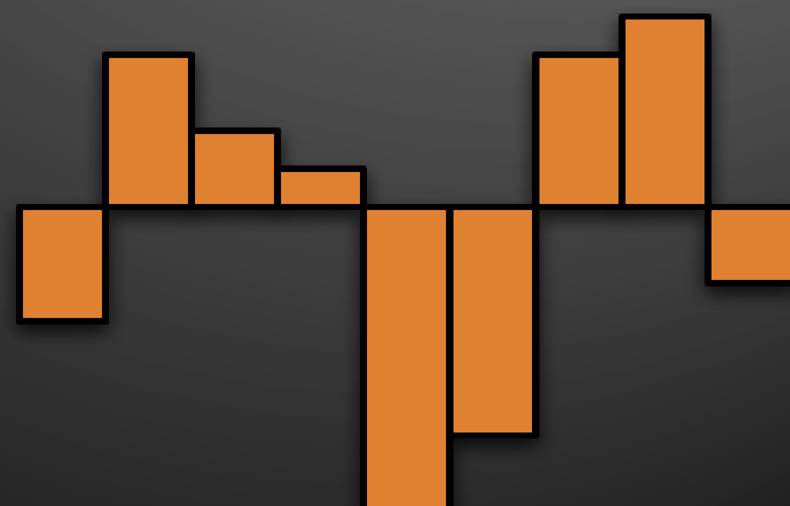
Maximum Contiguous Subsequence Sum

Q1

Maximum Contiguous Subsequence Sum

A deceptively deep problem with a surprising solution.

{−3, 4, 2, 1, −8, −6, 4, 5, −2}

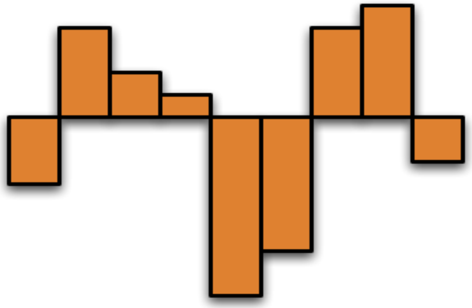


Why do we look at this problem?

- It’s interesting
- Analyzing the obvious solution is instructive:
- We can make the program more efficient

A Nice Algorithm Analysis Example

- **Problem**: Given a sequence of numbers, find the maximum sum of a contiguous subsequence.



- **Consider**:
  - What if all the numbers were positive?
  - What if they all were negative?
  - What if we left out “contiguous”?

Q2-4

Formal Definition: Maximum Contiguous Subsequence Sum

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

- In {−2, **11, −4, 13**, −5, 2}, $S_{2,4}$ = ?
- In {1, −3, 4, −2, −1, 6}, what is MCSS?
- If every element is negative, what's the MCSS?

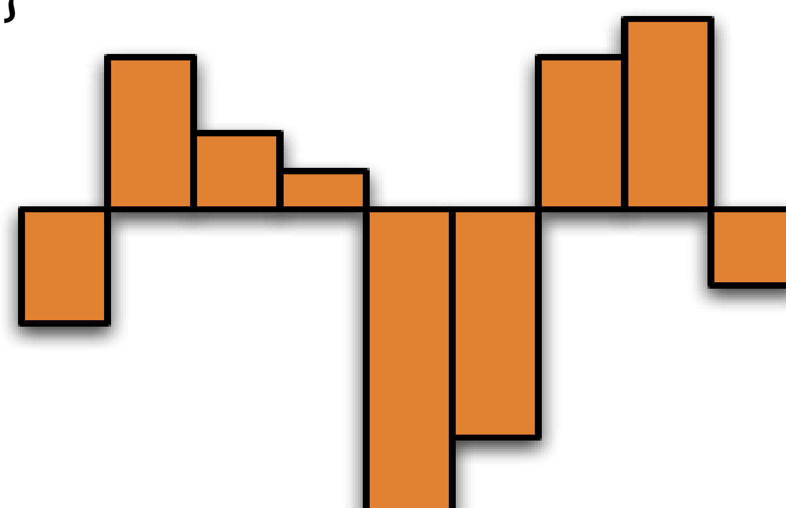
1-based indexing

Q5

A quick-and-dirty algorithm

- Design one right now.
  - Efficiency doesn't matter.
  - It has to be easy to understand.
  - 3 minutes

- Examples to consider:
  - {–3, 4, 2, 1, –8, –6, 4, 5, –2}
  - {5, 6, –3, 2, 8, 4, –12, 7, 2}



First Algorithm

Find the sums of *all* subsequences

```
public final class MaxSubTest {
    private static int seqStart = 0;
    private static int seqEnd = 0;

    /* First maximum contiguous subsequence sum algorithm.
     * seqStart and seqEnd represent the actual best sequence.
     */
    public static int maxSubSum1( int [ ] a ) {
        int maxSum = 0;
          //In the analysis we use "n" as a shorthand for "a.length
        for( int i = 0; i < a.length; i++ )   "
            for( int j = i; j < a.length; j++ ) {
                int thisSum = 0;

                for( int k = i; k <= j; k++ )
                    thisSum += a[ k ];

                if( thisSum > maxSum ) {
                    maxSum   = thisSum;
                    seqStart = i;
                    seqEnd   = j;
                }
            }
        return maxSum;
    }
```

i: beginning of subsequence

j: end of subsequence

k: steps through each element of subsequence

Where will this algorithm spend the most time?

How many times (exactly, as a function of N = a.length) will that statement execute?

Analysis of this Algorithm

- What statement is executed the most often?
- How many times?
- How many triples, *(i,j,k)* with $1 \le i \le k \le j \le n$?

```
//In the analysis we use "n" as a shorthand for "a.length "
for( int i = 0; i < a.length; i++ )
    for( int j = i; j < a.length; j++ ) {
        int thisSum = 0;

        for( int k = i; k <= j; k++ )
            thisSum += a[ k ];
```

Outer numbers could be 0 and n – 1, and we'd still get the same answer.

Three ways to find the sum

- By hand
- Using Maple
- A tangent (Related to urns and probabilities?)

Q6, Q7

Counting is (surprisingly) hard!

- How many triples, ***(i,j,k)*** with ***1≤i≤k≤j≤n*** ?
- What is that as a summation?

$$\sum_{i=1}^{n}\left(\sum_{j=i}^{n}\left(\sum_{k=i}^{j}1\right)\right)$$

- Let's solve it by hand to practice with sums

Hidden: One part of the process will be

$$\sum_{j=1}^{n} j = \sum_{j=1}^{i-1} j + \sum_{j=i}^{n} j$$

We have seen this idea before

Then we can solve for the last term to get a formula that we need on the next slide:

$$\sum_{j=i}^{n} j = \sum_{j=1}^{n} j - \sum_{j=1}^{i-1} j = \frac{n(n+1)}{2} - \frac{(i-1)i}{2}$$

Hidden

$$\sum_{i=1}^{n}\left(\sum_{j=i}^{n}\left(\sum_{k=i}^{j}1\right)\right)=\sum_{i=1}^{n}\left(\sum_{j=i}^{n}(j-i+1)\right)=\sum_{i=1}^{n}\left(\sum_{j=i}^{n}j-\sum_{j=i}^{n}i+\sum_{j=i}^{n}1\right)$$

$$=\sum_{i=1}^{n}\left(\frac{n(n+1)}{2}-\frac{(i-1)i}{2}-i(n-i+1)+(n-i+1)\right)$$

$$=\sum_{i=1}^{n}\left(\frac{n(n+1)}{2}+n+1-i(n+\tfrac{3}{2})+\tfrac{1}{2}i^2\right)=\left(\frac{n(n+1)}{2}+n+1\right)\sum_{i=1}^{n}1-(n+\tfrac{3}{2})\sum_{i=1}^{n}i+\tfrac{1}{2}\sum_{i=1}^{n}i^2$$

$$=\left(\frac{n^2+3n+2}{2}\right)n-(n+\tfrac{3}{2})\frac{n(n+1)}{2}+\tfrac{1}{2}\frac{n(n+1)(2n+1)}{6}$$

Simplify the sum

$$\sum_{i=1}^{n}\left(\sum_{j=i}^{n}\left(\sum_{k=i}^{j}1\right)\right)$$

- When it gets down to “just Algebra”, Maple is our friend

Help from Maple, part 1

Simplifying the last step of the monster sum

```
> simplify((n^2+3*n+2)/2*n
  -(n+3/2)*n*(n+1)/2+1/2*n*(n+1)*(2*n+1)/6);
```

$$\frac{1}{6}n^3+\frac{1}{2}n^2+\frac{1}{3}n$$

```
> factor(%);
```

$$\frac{1}{6}(n+2)\,n\,(n+1)$$

Help from Maple, part 2

Letting Maple do the whole thing for us:

```
sum(sum(sum(1, k=i..j), j=i..n), i=1..n);
```

$$\frac{1}{2}(n+1)n^2+2(n+1)n+\frac{1}{3}n+\frac{5}{6}-\frac{1}{2}n(n+1)^2-(n+1)^2+\frac{1}{6}(n+1)^3-\frac{1}{2}n^2$$

```
> factor(simplify(%));
```

$$\frac{1}{6}(n+2)n(n+1)$$

We get same answer if we sum from 0 to n-1, instead of 1 to n

```
factor(simplify(sum(sum(sum(1,k=i..j), j=i..n),
i=1..n)));
```

$$\frac{n\,(n+2)\,(n+1)}{6}$$

```
factor(simplify(sum(sum(sum(1,k=i..j),j=i..n-1),
i=0..n-1)));
```

$$\frac{n\,(n+2)\,(n+1)}{6}$$

Interlude

- Computer Science is no more about computers than astronomy is about _______.

Donald Knuth

Interlude

- Computer Science is no more about computers than astronomy is about <u>telescopes</u>.

Donald Knuth

"Magic" Tangent: Another (clever) way to count it

- How many triples, (i,j,k) with $1 \le i \le k \le j \le n$?

- The trick:
  - Find a set that's **easier to count** that has a **one-to-one correspondence** with the original

Q8

The "equivalent count" set

- **We want to count** the number of triples, *(i,j,k)* with $1 \le i \le k \le j \le n$
- First get an urn
  - Put in n white balls labeled 1,2,…,n
  - Put in one red ball and one blue one
- Choose 3 balls
  - If red drawn, = min of other 2
  - If blue drawn, = max of other 2
- What numbers do we get?

http://www.talaveraandmore.com, for all your urn needs!

The Correspondence with $1 \leq i \leq k \leq j \leq n$

- Choose 3 balls
  - If red drawn, = min of other 2
  - If blue drawn, = max of other 2

Triple of balls	Corresponding triple of numbers
(i, k, j)	(i, k, j)
(red, i, j)	(i, i, j)
(blue i, j)	(i, j, j)
(red, blue, i)	(i, i, i)

How does this help?!?

- There's a formula!
- It counts the ways to choose M items from a set of P items "without replacement"
- "P choose M" written ${}_PC_M$ or $\binom{P}{M}$ is:

$$\binom{P}{M} = \frac{P!}{M!(P-M)!}$$

- So ${}_{n+2}C_3$ is

$$\binom{n+2}{3} = \frac{(n+2)!}{3!(n-1)!} = \frac{n(n+1)(n+2)}{6}$$

What is the main source of the simple algorithm's inefficiency?

```
//In the analysis we use "n" as a shorthand for "a.length "
for( int i = 0; i < a.length; i++ )
    for( int j = i; j < a.length; j++ ) {
        int thisSum = 0;

        for( int k = i; k <= j; k++ )
            thisSum += a[ k ];
```

- The performance is bad!

Eliminate the most obvious inefficiency…

```
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;
        }
    }
}
```

This is Θ(?)

Q9, Q10

Can we do even better?

Tune in next time for the exciting conclusion!

http://www.etsu.edu/math/gardner/batman