Maximum Contiguous Subsequence Sum

Recap: Problem definition: details

Problem definition: Given a sequence of n integers $A_1, A_2, \ldots, A_n$, ($n \geq 0$, some numbers may be negative). Find the maximum sum of a consecutive subsequence

$S_{i,j} = \sum_{k=i}^{j} A_k$. If i>j or if all of the numbers $A_i, \ldots, A_j$ are negative, we define the sum to be zero. We use the abbreviation $A_{i,j}$ to stand for $A_i, \ldots, A_j$. Note that $S_{i,j}$ is the sum of the numbers in $A_{i,j}$.

Examples (from the DS book):

What is $S_{2,4}$?

Can a max-sum subsequence begin with a negative number?

{-2, **11**, **-4**, **13**, -5, 2} (maximum subsequence sum is 20),
{1, -3, **4**, **-2**, **-1**, **6**} (maximum subsequence sum is 7).

When we store the sequence in a Java array, the subscripts begin with 0.

When we refer to the problem in the abstract here, the subscripts will begin with 1. (because the Weiss DS book does it this way, as do most mathematicians)

Recap: Eliminate the most obvious inefficiency, get Θ(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;
        }
    }
}
```

Can we do even better?

Observations?

- (3 minutes)
- Walk through
  - {-3, 4, 2, 1, -8, -6, 4, 5, -2}
- Any subsequences you can safely ignore?
  - Discuss with another student

```java
/**
* Linear-time maximum contiguous subsequence sum algorithm.
* seqStart and seqEnd represent the actual best sequence.
*/
public static int maxSubSum3( int [ ] a )  {
    int maxSum = 0;
    int thisSum = 0;

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

        if( thisSum > maxSum ) {
            maxSum = thisSum;
            seqStart = i;
            seqEnd   = j;
        } else if( thisSum < 0 ) {
            i = j + 1;
            thisSum = 0;
        }
    }

    return maxSum;
}
```

Observation 1

- We noted that a max-sum sequence $A_{i,j}$ cannot begin with a negative number.
- Generalizing this, it cannot begin with a prefix ($A_{i,k}$ with k<j) whose sum is negative.
  - **Proof: If $S_{i,k}$ is negative, then $S_{k+1,j} > S_{i,j}$, so $A_{i,j}$ would not be a sequence that produces the maximum sum.**

Observation 2

- All contiguous subsequences that border the maximum contiguous subsequence must have negative (or zero) sums.
- Proof: If one of them had a positive sum, we could simply append (or prepend) it to get a sum that is larger than the maximum. Impossible!

Observation 3

For any i, Let $A_{i,j}$ be the *first* subsequence (starting with A_i) whose sum $S_{i,j}$ is negative.

Then for any p and q such that $i \leq p \leq j$ and $p \leq q$, either

- $A_{p,q}$ is not a maximum contiguous subsequence of $A_1, A_2, \ldots, A_n$, **or**
- $A_{p,q}$ has the same sum as a maximum contiguous subsequence that has been previously observed.

Proof of observation 3

- If p=i,
  - then we are in the case of Observation 1.
- Otherwise p>i, and so $S_{i,q} = S_{i,p-1} + S_{p,q}$.
- Since j is the *lowest* index for which $S_{i,j} < 0$,
  - $S_{i,p-1} \geq 0$, and thus $S_{p,q} \leq S_{i,q}$.
- If q > j ,
  - then (first diagram) Observation 1 says that $A_{i,q}$ is not a maximum contiguous subsequence, and thus neither is $A_{p,q}$.
  - Otherwise (second diagram) q<=j and $A_{p,q}$ has a sum that's no bigger than sum of already-seen $A_{i,q}$.

Observation 3 (recap)

- For any i, Let $A_{i,j}$ be the *first* subsequence (starting with A_i) whose sum $S_{i,j}$ is negative. Then for any p and q such that $i \leq p \leq j$ and $p \leq q$, either $A_{p,q}$ is not a maximum contiguous subsequence of A_1, A_2, ... , An, or $A_{p,q}$ has the same sum as a maximum contiguous subsequence that has been previously observed.
- **Implication of this:** If we find that $S_{i,j}$ is negative, we can skip all sums that begin with any of A_i, A_{i+1}, …, A_j, so we can set i to be j+1.

New, improved code!

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

    if( thisSum > maxSum )
    {
        maxSum = thisSum;
        seqStart = i;
        seqEnd   = j;
    }
    else if( thisSum < 0 )
    {
        i = j + 1;
        thisSum = 0;
    }
}
```

If the current consecutive sum we're examining ever becomes negative, the sequence with the maximum sum cannot begin anywhere in the current sequence, so we can skip past the end of it.

Analyze it.

Conclusions

- The first algorithm we think of may be a lot worse than the best algorithm for a problem.
- Improvement sometimes relies on clever ideas.
- Analysis sometimes takes some serious thought.