

The Maximum Subsequence Sum Problem (MSSP)

Section 2.4.3

Ali Erkan
Ithaca College

MSSP: The Quadratic Algorithm, Code

```

1 public static
2 int msspQuadratic( int [] a ) {
3     int maxSum = 0;
4
5     for( int i = 0; i < a.length; i++ ) {
6         int thisSum = 0;
7         for( int j = i; j < a.length; j++ ) {
8             thisSum += a[ j ];
9             if( thisSum > maxSum )
10                maxSum = thisSum;
11        }
12    }
13    return maxSum;
14 }

```

MSSP: The Quadratic Algorithm

► Max of the five intervals:

-2 11 -4 13 -5 -2

i j
i j
i j
i j
i j

► Max of the four intervals:

-2 11 -4 13 -5 -2

i j
i.....j
i j
i j

► Max of the three intervals:

-2 11 -4 13 -5 -2

i j
i j
i j

► Max of the two intervals:

-2 11 -4 13 -5 -2

i j
i j

► Max of the one interval:

-2 11 -4 13 -5 -2

i j

MSSP: The Quadratic Algorithm, Analysis

Max of the values from the five previous steps: 11 + -4 + 13 = 20

So how much work does this algorithm take?

$$\sum_{i=0}^{N-2} \left(\sum_{j=i+1}^{N-1} 1 \right) = \dots$$

= whatever; first figure out the inner part

$$\sum_{j=i+1}^{N-1} 1 = \textcircled{A}$$

$$= \textcircled{A}$$

$$= \textcircled{A}$$

$$= \textcircled{A}$$

MSSP: The Quadratic Algorithm, Analysis

$$\begin{aligned}\sum_{i=0}^{N-2} \left(\sum_{j=i+1}^{N-1} 1 \right) &= \sum_{i=0}^{N-2} (N - i - 1) \\ &= \left(\sum_{i=0}^{N-2} N \right) - \left(\sum_{i=0}^{N-2} i \right) - \left(\sum_{i=0}^{N-2} 1 \right) \\ &= \textcircled{A} \\ &= \textcircled{A}\end{aligned}$$

MSSP: The Quadratic Algorithm, Analysis

$$\begin{aligned}\sum_{i=0}^{N-2} \left(\sum_{j=i+1}^{N-1} 1 \right) &= (N^2 - 2N + 1) - \sum_{i=0}^{N-2} i \\ &= \textcircled{A} \\ &= N^2 - 2N + 1 - \frac{N^2 - 3N + 2}{2} \\ &= \frac{2N^2 - 4N + 2 - N^2 + 3N - 2}{2} \\ &= \frac{N^2 - N}{2} \\ f(N) &= \frac{N^2 - N}{2}\end{aligned}$$

MSSP: The Quadratic Algorithm, Analysis

$$\begin{aligned}\sum_{i=0}^{N-2} \left(\sum_{j=i+1}^{N-1} 1 \right) &= ((N-1)N) - \left(\sum_{i=0}^{N-2} i \right) - ((N-1)1) \\ &= (N^2 - N) - (N-1) - \left(\sum_{i=0}^{N-2} i \right) \\ &= (N^2 - 2N + 1) - \sum_{i=0}^{N-2} i\end{aligned}$$

MSSP: The LogLinear Algorithm

- We saw that the second algorithm takes $\frac{N^2 - N}{2}$ steps which is $O(N^2)$. The third algorithm is **recursive**. Is it better, worse, or the same as the second one in asymptotic terms?
- **Recurrence relation** for the number of steps:

$$f(n) = 2f\left(\frac{n}{2}\right) + n$$

$$f(1) = 1$$

MSSP: The LogLinear Algorithm, Code

```
1 public static
2 int msspLogLinear( int [] a, int left, int right ) {
3     if( left == right )
4         if( a[ left ] > 0 )
5             return a[ left ];
6         else
7             return 0;
8
9     ...recursive calls...
10    ...linear block...
11    ...picking max of three...
12 }
```

MSSP: The LogLinear Algorithm, Code

```
1 public static
2 int msspLogLinear( int [] a, int left, int right ) {
3     ...initializations...
4     ...recursive calls...
5
6     int maxLeftBorderSum = 0;
7     int leftBorderSum = 0;
8     for( int i = center; i >= left; i-- ) {
9         leftBorderSum += a[ i ];
10        if( leftBorderSum > maxLeftBorderSum )
11            maxLeftBorderSum = leftBorderSum;
12    }
13
14    ...picking max of three...
15 }
```

MSSP: The LogLinear Algorithm, Code

```
1 public static
2 int msspLogLinear( int [] a, int left, int right ) {
3     ...initializations...
4
5     int center = ( left + right ) / 2;
6     int maxLeftSum = msspLogLinear( a, left, center );
7     int maxRightSum = msspLogLinear( a, center + 1, right );
8
9     ...linear block...
10    ...picking max of three...
11 }
```

MSSP: The LogLinear Algorithm, Code

```
1 public static
2 int msspLogLinear( int [] a, int left, int right ) {
3     ...initializations...
4     ...recursive calls...
5
6     int maxRightBorderSum = 0;
7     int rightBorderSum = 0;
8     for( int i = center+1; i <= right; i++ ) {
9         rightBorderSum += a[ i ];
10        if( rightBorderSum > maxRightBorderSum )
11            maxRightBorderSum = rightBorderSum;
12    }
13
14    ...picking max of three...
15 }
```

MSSP: The LogLinear Algorithm, Code

```
1 public static
2 int mspLogLinear( int [] a, int left, int right ) {
3     ...initializations...
4     ...recursive calls...
5     ...linear block...
6
7     return max3( maxLeftSum,
8                 maxRightSum,
9                 maxLeftBorderSum + maxRightBorderSum
10                );
11 }
```

MSSP: The LogLinear Algorithm, Analysis

► One more “iteration”:

$$\begin{aligned}f(n) &= 4f\left(\frac{n}{4}\right) + 2n \\f\left(\frac{n}{4}\right) &= 2f\left(\frac{n}{8}\right) + \frac{n}{4} \\f(n) &= 4\left(2f\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n \\&= 8f\left(\frac{n}{8}\right) + 4\frac{n}{4} + 2n \\&= 8f\left(\frac{n}{8}\right) + 3n\end{aligned}$$

MSSP: The LogLinear Algorithm, Analysis

► Let’s define $f(n)$ in terms of “smaller versions” of the same problem:

$$\begin{aligned}f(n) &= 2f\left(\frac{n}{2}\right) + n \\f\left(\frac{n}{2}\right) &= 2f\left(\frac{n}{4}\right) + \frac{n}{2} \\f(n) &= 2\left(2f\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n \\&= 4f\left(\frac{n}{4}\right) + 2\frac{n}{2} + n \\&= 4f\left(\frac{n}{4}\right) + 2n\end{aligned}$$

MSSP: The LogLinear Algorithm, Analysis XYZ

► One last iteration:

$$\begin{aligned}f(n) &= 8f\left(\frac{n}{8}\right) + 3n \\f\left(\frac{n}{8}\right) &= \textcircled{A} \\f(n) &= \textcircled{A} \\&= \textcircled{A} \\&= \textcircled{A}\end{aligned}$$

MSSP: The LogLinear Algorithm, Analysis

► Summary of what we found so far:

- At step 1, we have $f(n) = 2f(\frac{n}{2}) + n$
- At step 2, we have $f(n) = 4f(\frac{n}{4}) + 2n$
- At step 3, we have $f(n) = 8f(\frac{n}{8}) + 3n$
- At step 4, we have $f(n) = 16f(\frac{n}{16}) + 4n$

► Generalization:

- At step i , we have $f(n) = \textcircled{A}$

► Seeing where this goes:

- What is the largest value 2^i could have? ...
- What is the largest value i could have? ...

MSSP: The Linear Algorithm

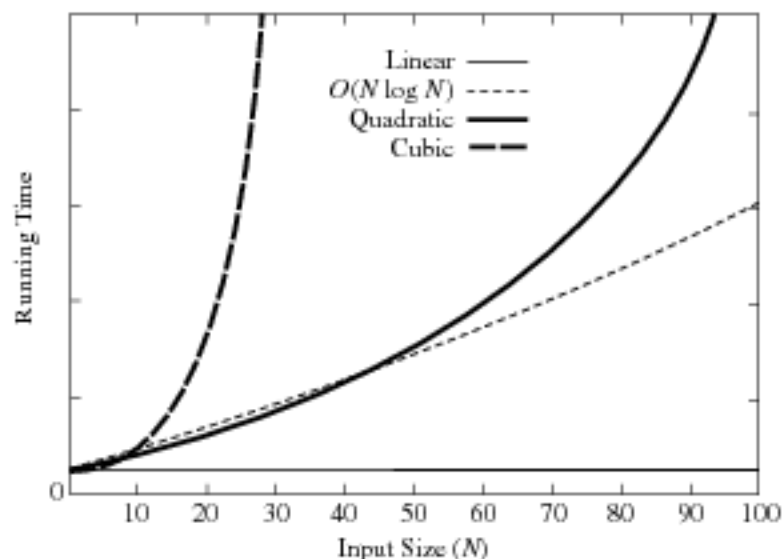
```
1 public static
2 int masplinear( int [ ] a ) {
3     int maxSum = 0, thisSum = 0;
4
5     for( int j = 0; j < a.length; j++ ) {
6         thisSum += a[ j ];
7
8         if( thisSum > maxSum )
9             maxSum = thisSum;
10        else if( thisSum < 0 )
11            thisSum = 0;
12    }
13
14    return maxSum;
15 }
```

MSSP: The LogLinear Algorithm, Analysis

► We now try $i = \lg(n)$ and do some algebraic simplification:

$$\begin{aligned} f(n) &= 2^i f\left(\frac{n}{2^i}\right) + (n)i \\ &= \dots \\ &= O(n \lg n) \end{aligned}$$

MSSP: Comparison Of Performance



MSSP: Comparison Of Performance

