Lecture 6

Divide and Conquer IV: integer multiplication, further examples

CS 161 Design and Analysis of Algorithms

Ioannis Panageas

Divide and Conquer (recap)

Steps of method:

- **Divide** input into parts (smaller problems)
- **Conquer** (solve) each part recursively
- **Combine** results to obtain solution of original

$$
\begin{aligned}
T(n) = {} & \text{divide time} \\
& + T(n_1) + T(n_2) + \ldots + T(n_k) \\
& + \text{combine time}
\end{aligned}
$$

Case study VI: Integer Multiplication

Problem: Given two n-digit numbers a, b in binary, compute $a \cdot b$.

Example: $a = 101, b = 111$. Answer: 100011.

Case study VI: Integer Multiplication

Problem: Given two n-digit numbers a, b in binary, compute $a \cdot b$.

Example: $a = 101$, $b = 111$. Answer: 100011.

Standard Algorithm: $\Theta(n^2)$ time. Summing two n-bit numbers takes $\Theta(n)$ time.

Addition

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Multiplication

									1	1	0	1	0	1	0	1
								*	0	1	1	1	1	1	0	1
								1	1	0	1	0	1	0	1	0
							0	0	0	0	0	0	0	0	0	
						1	1	0	1	0	1	0	1	0		
					1	1	0	1	0	1	0	1	0			
				1	1	0	1	0	1	0	1	0				
			1	1	0	1	0	1	0	1	0					
		1	1	0	1	0	1	0	1	0						
	0	0	0	0	0	0	0	0	0							
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Case study VI: Integer Multiplication

Problem: Given two n-digit numbers a, b in binary, compute $a \cdot b$.

Example: $a = 101, b = 111$. Answer: 100011.

Standard Algorithm: $\Theta\left(n^2\right)$ time. Summing two n-bit numbers takes $\Theta(n)$ time.

Can we do better?

Addition

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Multiplication

									1	1	0	1	0	1	0	1
								*	0	1	1	1	1	1	0	1
								1	1	0	1	0	1	0	1	0
							0	0	0	0	0	0	0	0	0	
						1	1	0	1	0	1	0	1	0		
					1	1	0	1	0	1	0	1	0			
				1	1	0	1	0	1	0	1	0				
			1	1	0	1	0	1	0	1	0					
		1	1	0	1	0	1	0	1	0						
	0	0	0	0	0	0	0	0	0							
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0

Case study VI: Integer Multiplication

Idea: Divide and conquer.

$$a = \underbrace{a_1 a_2 \dots a_{n/2}}_{a_L} \ \underbrace{a_{n/2+1} \dots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \dots b_{n/2}}_{b_L} \ \underbrace{b_{n/2+1} \dots b_n}_{b_R}$$

Case study VI: Integer Multiplication

Idea: Divide and conquer.

$$a = \underbrace{a_1 a_2 \ldots a_{n/2}}_{a_L} \; \underbrace{a_{n/2+1} \ldots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \ldots b_{n/2}}_{b_L} \; \underbrace{b_{n/2+1} \ldots b_n}_{b_R}$$

Recursively

$$a \cdot b = a_R \cdot b_R + 2^{\frac{n}{2}} a_L \cdot b_R + a_R \cdot 2^{\frac{n}{2}} b_L + 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L$$

Case study VI: Integer Multiplication

Idea: Divide and conquer.

$$a = \underbrace{a_1 a_2 \ldots a_{n/2}}_{a_L} \; \underbrace{a_{n/2+1} \ldots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \ldots b_{n/2}}_{b_L} \; \underbrace{b_{n/2+1} \ldots b_n}_{b_R}$$

Recursively

$$a \cdot b = a_R \cdot b_R + 2^{\frac{n}{2}} a_L \cdot b_R + a_R \cdot 2^{\frac{n}{2}} b_L + 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L$$

Running time: $T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n) \rightarrow \Theta(n^2)$ by Master thm

Case study VI: Integer Multiplication

Idea (modified): Divide and conquer.

$$a = \underbrace{a_1 a_2 \ldots a_{n/2}}_{a_L}\ \underbrace{a_{n/2+1} \ldots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \ldots b_{n/2}}_{b_L}\ \underbrace{b_{n/2+1} \ldots b_n}_{b_R}$$

$$a \cdot b = 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L + a_R \cdot b_R +$$

$$2^{\frac{n}{2}}(\,(a_L - a_R) \cdot (b_R - b_L) + a_L \cdot b_L + a_R \cdot b_R)$$

Case study VI: Integer Multiplication

Idea (modified): Divide and conquer.

$$a = \underbrace{a_1 a_2 \dots a_{n/2}}_{a_L} \; \underbrace{a_{n/2+1} \dots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \dots b_{n/2}}_{b_L} \; \underbrace{b_{n/2+1} \dots b_n}_{b_R}$$

Recursively compute

1. $(a_L - a_R)(b_R - b_L)$
2. $a_L \cdot b_L$
3. $a_R \cdot b_R$

$$a \cdot b = 2^{\frac{n}{2}} a_L \cdot 2^{\frac{n}{2}} b_L + a_R \cdot b_R +$$

$$2^{\frac{n}{2}} \left((a_L - a_R) \cdot (b_R - b_L) + a_L \cdot b_L + a_R \cdot b_R \right)$$

Case study VI: Integer Multiplication

Idea (modified): Divide and conquer.

$$a = \underbrace{a_1 a_2 \dots a_{n/2}}_{a_L} \underbrace{a_{n/2+1} \dots a_n}_{a_R}$$

$$b = \underbrace{b_1 b_2 \dots b_{n/2}}_{b_L} \underbrace{b_{n/2+1} \dots b_n}_{b_R}$$

Recursively compute

1. $(a_L - a_R)(b_R - b_L)$
2. $a_L \cdot b_L$
3. $a_R \cdot b_R$

$\Theta(n^{1.585})$

Running time: $T(n) = 3T\left(\frac{n}{2}\right) + \Theta(n) \to \Theta\left(n^{\log_2 3}\right)$ by Master thm

Case study VII: Computing powers

Problem: Given two positive integers numbers a, n compute a^n .

Example: $a = 3, n = 4$. Answer: 81.

Case study VII: Computing powers

Problem: Given two positive integers numbers a, n compute a^n .

Example: $a = 3, n = 4$. Answer: 81.

Obvious approach:

$\text{ans} \leftarrow 1$
For $i = 1$ to n **do**
  $\text{ans} \leftarrow a \cdot \text{ans}$
return ans

$\Theta(n)$ operations

Can we do better?

Case study VII: Computing powers

Problem: Given two positive integers numbers a, n compute a^n .

Example: $a = 3, n = 4$. Answer: 81.

Idea: Divide and Conquer.

Divide n in $n/2$ and $n/2$. Compute x = $a^{n/2}$ recursively. Return x^2.
Be careful on the parity of n.

Case study VII: Computing powers

Problem: Given two positive integers numbers a, n compute a^n .

Example: $a = 3, n = 4$. Answer: 81.

Idea: Divide and Conquer.

$$
\begin{array}{l}
\text{Power}(a, n) \\
\textbf{If } n == 1 \textbf{ then return } \text{a} \\
x \leftarrow \text{Pow}(a, \lfloor n/2 \rfloor) \\
\textbf{If } n \bmod 2 == 0 \textbf{ then} \\
\quad \textbf{return } x \cdot x \\
\textbf{else return } \; a \cdot x \cdot x
\end{array}
$$

Case study VII: Computing powers

Problem: Given two positive integers numbers a, n compute a^n .

Example: $a = 3, n = 4$. Answer: 81.

Idea: Divide and Conquer.

$\mathrm{Power}(a, n)$
If $n == 1$ **then return** a
$x \leftarrow \mathrm{Pow}(a, \lfloor n/2 \rfloor)$
If $n \bmod 2 == 0$ **then**
    return $x \cdot x$
else return $a \cdot x \cdot x$

Base case

Divide + Conquer

Combine

Case study VII: Computing powers

Problem: Given two positive integers numbers a, n compute a^n .

Example: $a = 3, n = 4$. Answer: 81.

Idea: Divide and Conquer.

$\mathrm{Power}(a, n)$
If $n == 1$ **then return** a
$x \leftarrow \mathrm{Pow}(a, \lfloor n/2 \rfloor)$
If $n \bmod 2 == 0$ **then**
  return $x \cdot x$
else return $a \cdot x \cdot x$

Base case

Divide + Conquer

Combine

Running time: $T(n) = T\left(\frac{n}{2}\right) + \Theta(1) \rightarrow \Theta(\log n)$ by Master thm

Case study VII: Computing powers

Problem: Given two positive integers numbers a, n compute a^n .

Example: $a = 3, n = 4$. Answer: 81.

Idea: Divide and Conquer.

Remark: Same works for powers of Matrices.

$\text{Power}(a, n)$
If $n == 1$ **then return** a
$x \leftarrow \text{Pow}(a, \lfloor n/2 \rfloor)$
If $n \bmod 2 == 0$ **then**
  return $x \cdot x$
else return $a \cdot x \cdot x$

Base case

Divide + Conquer

Combine

Running time: $T(n) = T\left(\frac{n}{2}\right) + \Theta(1) \rightarrow \Theta(\log n)$ by Master thm

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Definition: $F_1 = F_2 = 1$ and $F_n \; = \; F_{n-1} \; + \; F_{n-2}$.

First 10 numbers of sequence: $1, 1, 2, 3, 5, 8, 13, 21, 34, 55$

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Definition: $F_1 = F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

First 10 numbers of sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Obvious approach:

ans1 $\leftarrow$ 1
ans2 $\leftarrow$ 1
If $n \leq 2$ **then return** 1
For $i = 3$ to n **do**
  temp $\leftarrow$ ans1
  ans1 $\leftarrow$ ans1 + ans2
  ans2 $\leftarrow$ temp
return ans

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Definition: $F_1 = F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

First 10 numbers of sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Obvious approach:

$\Theta(n)$ operations

Can we do better?

ans1 $\leftarrow$ 1
ans2 $\leftarrow$ 1
If $n \leq 2$ **then return** 1
For $i = 3$ to n **do**
  temp $\leftarrow$ ans1
  ans1 $\leftarrow$ ans1 + ans2
  ans2 $\leftarrow$ temp
return ans

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Definition: $F_1 = F_2 = 1$ and $F_n = F_{n-1} + F_{n-2}$.

First 10 numbers of sequence: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Idea: Express F_n as a power of a Matrix.

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}$$

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Idea: Express F_n as a power of a Matrix.

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix}$$

$$\begin{pmatrix} F_{n-1} \\ F_{n-2} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} F_{n-2} \\ F_{n-3} \end{pmatrix}$$

$$\vdots$$

$$\begin{pmatrix} F_3 \\ F_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} F_2 \\ F_1 \end{pmatrix}$$

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Idea: Express F_n as a power of a Matrix.

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2} \begin{pmatrix} F_2 \\ F_1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Idea: Express F_n as a power of a Matrix.

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2} \begin{pmatrix} F_2 \\ F_1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

F_n is $a + b$ and F_{n-1} is $c + d$!

Case study VIII: Fibonacci sequence

Problem: Given a positive integer numbers n, compute Fibonacci F_n .

Solution:

Compute matrix $\begin{pmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{pmatrix}^{\boldsymbol{n-2}}$ in $\boldsymbol{\Theta}(\log \boldsymbol{n})$ time.

Return the sum of the entries of first row.

Case study IX: From practice problems

Problem: Suppose you have an array A of n intervals $(x_1, y_1), \ldots, (x_n, y_n)$, where x_i, y_i are positive integers such that $x_i \leq y_i$. The interval (x_i, y_i) represents the set of integers between x_i and y_i. For example, the interval $(3, 8)$ represents the set $\{3, 4, 5, 6,7,8\}$.

Define the **overlap** of two intervals to be the number of integers that are members of both intervals. For example $(3, 8)$ and $(4, 9)$ have overlap 5 (numbers $4, 5, 6, 7, 8$) and $(1, 2)$ and $(3, 4)$ have overlap 0. Find the size of maximum overlap among all possible pairs of intervals.

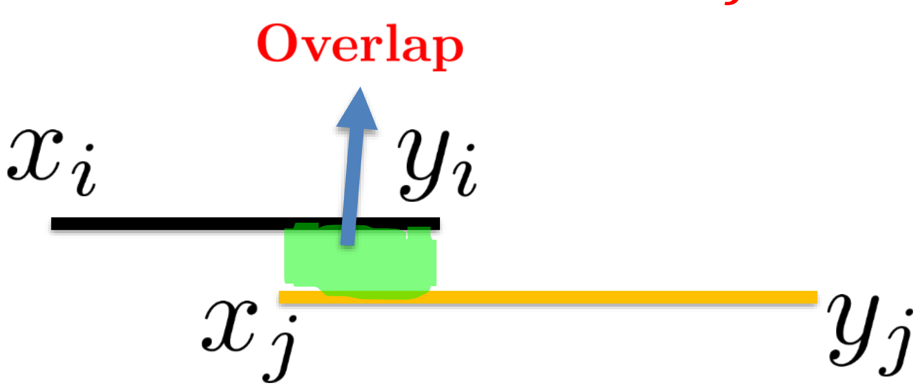
Example: $(1, 2), (3, 4), (3, 8), (4, 9)$. Answer: 5.

Case study IX: From practice problems

Obvious approach: For every pair i, j of intervals, find the overlap. Keep the maximum.

Suppose $x_i \leq x_j$.
(x_i, y_i) and (x_j, y_j) have overlap

$$\max(\min(y_i, y_j) - x_j + 1, 0).$$



Case study IX: From practice problems

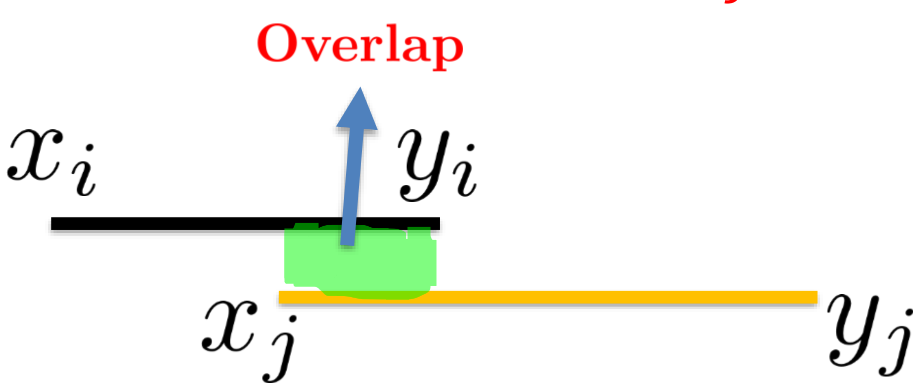
Obvious approach: For every pair i, j of intervals, find the overlap. Keep the maximum.

Suppose $x_i \leq x_j$.

(x_i, y_i) and (x_j, y_j) have overlap

$\Theta(n^2)$ running time

$$\max(\min(y_i, y_j) - x_j + 1, 0).$$



Can we do better?

Case study IX: From practice problems

Idea: Use divide and conquer. Suppose we first sort the intervals in increasing order of x-coordinate.

Case study IX: From practice problems

Idea: Use divide and conquer. Suppose we first sort the intervals in increasing order of x-coordinate.

- Divide the intervals in two parts L and R.
- Recursively find max overlap for each part maxL and maxR.
- Combine step?

Case study IX: From practice problems

Idea: Use divide and conquer. Suppose we first sort the intervals in increasing order of x-coordinate.

- Divide the intervals in two parts L and R.
- Recursively find max overlap for each part maxL and maxR.
- Combine step: maximum of maxL and maxR?

Case study IX: From practice problems

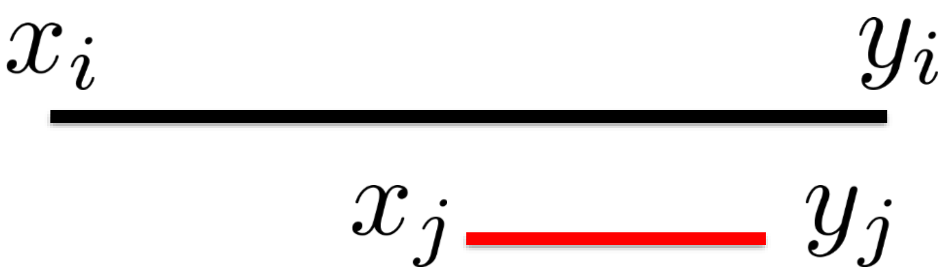
Idea: Use divide and conquer. Suppose we first sort the intervals in increasing order of x-coordinate.

- Divide the intervals in two parts L and R.
- Recursively find max overlap for each part maxL and maxR.
- Combine step: Check overlap between an interval in L and an interval in R. This should be in $\Theta(n)$.

We will scan the intervals once. One index for L and one index for R.

Case study IX: From practice problems

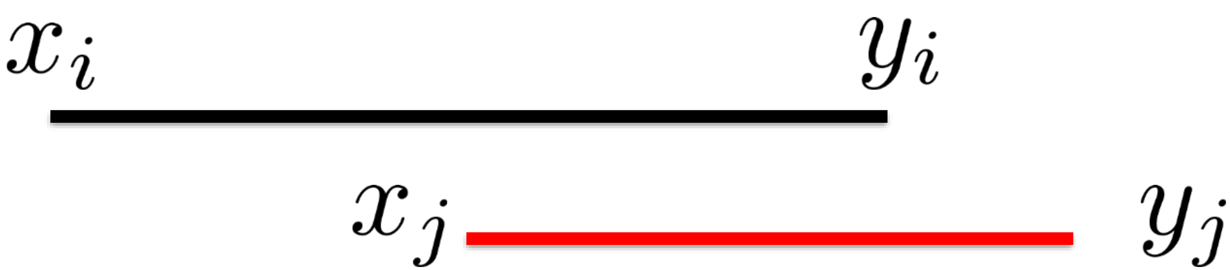
Combine step: Black is in L, red in R.



Overlap is $(y_j - x_j + 1)$. We can remove interval j from R.

Case study IX: From practice problems

Combine step: Black is in L, red in R.



Overlap is $(y_i - x_j + 1)$. We can remove interval i from L.

Case study IX: From practice problems

Combine step: Black is in L, red in R.

Overlap is $(y_i - x_j + 1)$. We can remove interval i from L.

All intervals after j in R will not give larger overlap with interval i.

Case study IX: From practice problems

Pseudocode:

$\text{Maxoverlap}(A[1:n])$

$\quad$ **If** $n == 1$ **return** 0

$\quad$ **maxL** $\leftarrow \text{Maxoverlap}(A[1:n/2])$

$\quad$ **maxR** $\leftarrow \text{Maxoverlap}(A[n/2+1:n])$

$\quad$ **maxComb** $\leftarrow 0$

$\quad$ $i \leftarrow 1,\ j \leftarrow n/2+1$

$\quad$ **While** $i \leq n/2$ and $j \leq n$ **do**

$\quad\quad$ **If** $\text{maxComb} < \text{overlap}(i,j)$ **then**

$\quad\quad\quad$ $\text{maxComb} = \text{overlap}(i,j)$

$\quad\quad$ **If** case 1 **then** $j \leftarrow j+1$

$\quad\quad$ **else If** case 2 **then** $i \leftarrow i+1$

$\quad$ **return** maximum of maxL, maxR and maxComb

Case study IX: From practice problems

Pseudocode:

Maxoverlap($A[1:n]$)

If $n == 1$ **return** 0

maxL $\leftarrow$Maxoverlap($A[1:n/2]$) — $T(n/2)$ Running time

maxR $\leftarrow$Maxoverlap($A[n/2+1:n]$) — $T(n/2)$ Running time

maxComb $\leftarrow 0$

$i \leftarrow 1,\ j \leftarrow n/2+1$

While $i \leq n/2$ and $j \leq n$ **do** — $\Theta(n)$ Running time

    If maxComb $<$ overlap(i, j) **then**

        maxComb = overlap(i, j)

    If case 1 **then** $j \leftarrow j+1$

    else If case 2 **then** $i \leftarrow i+1$

return maximum of maxL, maxR and maxComb

Case study IX: From practice problems

Pseudocode: $\Theta(n \log n)$ Running time

$\text{Maxoverlap}(A[1:n])$

$\textbf{If } n == 1 \textbf{ return } 0$

$\textbf{maxL} \leftarrow \text{Maxoverlap}(A[1:n/2])$ — $T(n/2)$ Running time

$\textbf{maxR} \leftarrow \text{Maxoverlap}(A[n/2+1:n])$ — $T(n/2)$ Running time

$\textbf{maxComb} \leftarrow 0$

$i \leftarrow 1,\ j \leftarrow n/2+1$

$\textbf{While } i \le n/2 \text{ and } j \le n \textbf{ do}$ — $\Theta(n)$ Running time

$\quad \textbf{If } \text{maxComb} < \text{overlap}(i,j) \textbf{ then}$

$\quad\quad \text{maxComb} = \text{overlap}(i,j)$

$\quad \textbf{If } \text{case 1 } \textbf{then } j \leftarrow j+1$

$\quad \textbf{else If } \text{case 2 } \textbf{then } i \leftarrow i+1$

$\textbf{return } \text{maximum of maxL, maxR and maxComb}$