COL 351 : Analysis & Design of Algorithms

Lecture 2

Divide & Conquer I : Integer Multiplication and Merge Sort

July 24, 2024 | Rohit Vaish

ANNOUNCEMENTS

Sign up on the Teams channel

Sign up on Gradescope using @cse.iitd account

Fill Google form for tutorial conflicts by July 25 (Thurs)

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$\begin{array}{cccc} 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 \\ \hline \end{array}$$

Grade-school algorithm

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$\begin{array}{r} 5678 \\ \times\ 1234 \\ \hline 22712 \end{array}$$

Grade-school algorithm

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$\begin{array}{r} 5\ 6\ 7\ 8 \\ 1\ 2\ 3\ 4 \\ \hline 2\ 2\ 7\ 1\ 2 \\ 1\ 7\ 0\ 3\ 4 \end{array}$$

Grade-school algorithm

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$
\begin{array}{cccc}
 & 5 & 6 & 7 & 8 \\
 & 1 & 2 & 3 & 4 \\
\hline
 & 2 & 2 & 7 & 1 & 2 \\
 & 1 & 7 & 0 & 3 & 4 \\
1 & 1 & 3 & 5 & 6
\end{array}
$$

Grade-school algorithm

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$
\begin{array}{rrrrr}
 & 5 & 6 & 7 & 8 \\
 & 1 & 2 & 3 & 4 \\
\hline
 & 2 & 2 & 7 & 1 \quad 2 \\
 & 1 & 7 & 0 & 3 \quad 4 \\
1 & 1 & 3 & 5 & 6 \\
\hline
5 & 6 & 7 & 8 &
\end{array}
$$

Grade-school algorithm

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$
\begin{array}{rrrrr}
 & 5 & 6 & 7 & 8 \\
 & 1 & 2 & 3 & 4 \\
\hline
 & 2 & 2 & 7 & 1 & 2 \\
 & 1 & 7 & 0 & 3 & 4 \\
1 & 1 & 3 & 5 & 6 & \\
5 & 6 & 7 & 8 & & \\
\hline
7 & 0 & 0 & 6 & 6 & 5 & 2
\end{array}
$$

Grade-school algorithm

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$
\begin{array}{rrrrr}
 & 5 & 6 & 7 & 8 \\
 & 1 & 2 & 3 & 4 \\
\hline
 & 2 & 2 & 7 & 1 & 2 \\
 & 1 & 7 & 0 & 3 & 4 \\
1 & 1 & 3 & 5 & 6 & \\
5 & 6 & 7 & 8 & & \\
\hline
7 & 0 & 0 & 6 & 6 & 5 & 2
\end{array}
$$

basic operations: add or multiply two single-digit numbers

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$
\begin{array}{rrrrr}
 & 5 & 6 & 7 & 8 \\
 & 1 & 2 & 3 & 4 \\
\hline
 & 2 & 2 & 7 & 1 & 2 \\
 & 1 & 7 & 0 & 3 & 4 \\
1 & 1 & 3 & 5 & 6 & \\
5 & 6 & 7 & 8 & & \\
\hline
7 & 0 & 0 & 6 & 6 & 5 & 2
\end{array}
$$

basic operations: add or multiply two single-digit numbers

$$\#\text{ operations} \leq \underbrace{\text{constant}}_{\leq 9} \cdot n^2$$

Integer Multiplication

input: two n digit numbers x and y

output: the product $x \cdot y$

e.g.,

$$
\begin{array}{rrrrr}
 & 5 & 6 & 7 & 8 \\
 & 1 & 2 & 3 & 4 \\
\hline
 & 2 & 2 & 7 & 1 & 2 \\
 & 1 & 7 & 0 & 3 & 4 \\
1 & 1 & 3 & 5 & 6 & \\
5 & 6 & 7 & 8 & & \\
\hline
7 & 0 & 0 & 6 & 6 & 5 & 2
\end{array}
$$

basic operations: add or multiply two single-digit numbers

$$\#\text{ operations} \leq \underbrace{\text{constant}}_{\leq 9} \cdot n^2$$

Can we do better?

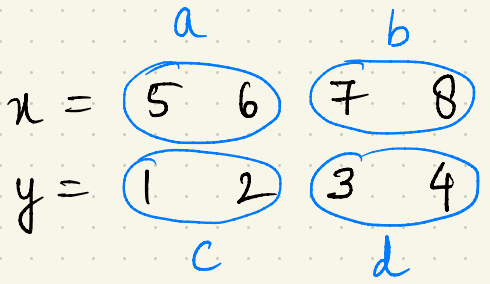
KARATSUBA MULTIPLICATION

KARATSUBA MULTIPLICATION

$x = 5\ 6\ 7\ 8$

$y = 1\ 2\ 3\ 4$

KARATSUBA MULTIPLICATION



KARATSUBA MULTIPLICATION

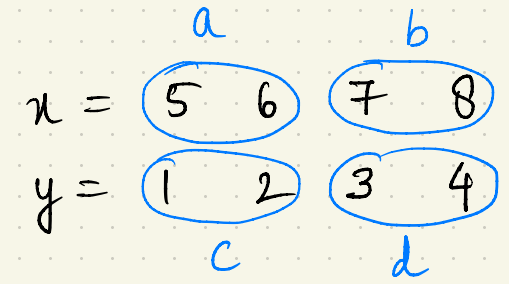
(1) compute $a \cdot c = 56 \times 12 = 672$

$x =$ (5 6) (7 8) — a, b

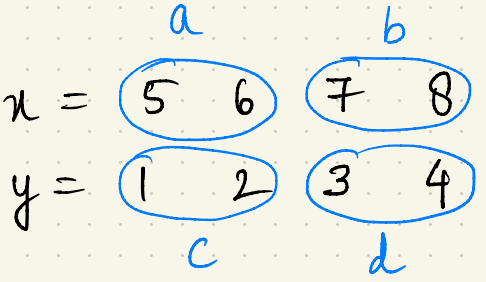
$y =$ (1 2) (3 4) — c, d

KARATSUBA MULTIPLICATION

1. compute $a \cdot c = 56 \times 12 = 672$
2. compute $b \cdot d = 78 \times 34 = 2652$



KARATSUBA MULTIPLICATION



1. compute $a \cdot c = 56 \times 12 = 672$
2. compute $b \cdot d = 78 \times 34 = 2652$
3. compute $(a+b) \cdot (c+d) = 134 \times 46 = 6164$

KARATSUBA MULTIPLICATION

$x = \overbrace{5\ 6}^{a}\ \overbrace{7\ 8}^{b}$

$y = \overbrace{1\ 2}^{c}\ \overbrace{3\ 4}^{d}$

1. compute $a \cdot c = 56 \times 12 = 672$
2. compute $b \cdot d = 78 \times 34 = 2652$
3. compute $(a+b) \cdot (c+d) = 134 \times 46 = 6164$
4. compute (3) − (2) − (1) = 2840

KARATSUBA MULTIPLICATION

$x = \overbrace{5\ 6}^{a}\ \overbrace{7\ 8}^{b}$

$y = \overbrace{1\ 2}^{c}\ \overbrace{3\ 4}^{d}$

① compute $a \cdot c = 56 \times 12 = 672$

② compute $b \cdot d = 78 \times 34 = 2652$

③ compute $(a+b) \cdot (c+d) = 134 \times 46 = 6164$

④ compute ③ − ② − ① = 2840

⑤ compute

$$\begin{array}{rl} 6\,7\,2\,0\,0\,0\,0 & \text{from } ① \\ 2\,6\,5\,2 & \text{from } ② \\ 2\,8\,4\,0\,0 & \text{from } ④ \\ \hline \end{array}$$

KARATSUBA MULTIPLICATION

$x = \overbrace{56}^{a}\ \overbrace{78}^{b}$

$y = \overbrace{12}^{c}\ \overbrace{34}^{d}$

① compute $a \cdot c = 56 \times 12 = 672$

② compute $b \cdot d = 78 \times 34 = 2652$

③ compute $(a+b) \cdot (c+d) = 134 \times 46 = 6164$

④ compute ③ − ② − ① = 2840

⑤ compute

$$
\begin{array}{rl}
6720000 & \text{from ①} \\
2652 & \text{from ②} \\
2840000 & \text{from ④} \\
\hline
7006652 & = x \cdot y
\end{array}
$$

A RECURSIVE ALGORITHM

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

Assuming n is even

A RECURSIVE ALGORITHM

input : two n digit numbers x and y

output : the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

Assuming n is even

NOTE : a, b, c, d are $n/2$ digit numbers

e.g., $x = 5678$ $a = 56$ $b = 78$

$y = 1234$ $c = 12$ $d = 34$

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

$$x \cdot y = (10^{n/2} a + b) \cdot (10^{n/2} c + d)$$

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

$$x \cdot y = (10^{n/2} a + b) \cdot (10^{n/2} c + d)$$

$$= 10^{n} ac + 10^{n/2} (ad + bc) + bd$$

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

$$x \cdot y = (10^{n/2} a + b) \cdot (10^{n/2} c + d)$$

$$= 10^{n} ac + 10^{n/2}(ad + bc) + bd$$

involves only the product of smaller numbers

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

$$x \cdot y = (10^{n/2} a + b) \cdot (10^{n/2} c + d)$$

$$= 10^{n} ac + 10^{n/2}(ad + bc) + bd$$

recursively solve smaller problems and combine the solutions

A RECURSIVE ALGORITHM

input: two n digit numbers x and y

output: the product $x \cdot y$

Write $x = 10^{n/2} \cdot a + b$ and $y = 10^{n/2} \cdot c + d$

$$x \cdot y = (10^{n/2} a + b) \cdot (10^{n/2} c + d)$$

$$= 10^{n} ac + 10^{n/2}(ad + bc) + bd$$

recursively solve smaller problems and combine the solutions

Base case? Easy! When $n = 1$, compute $x \cdot y$ in one step.

The Good, The Bad and The Ugly of Writing Algorithms

The Good

The Bad

The Ugly

The Good

The Bad

The recursive algorithm takes as input two n-digit numbers x and y and returns their product $x \cdot y$.

The base case of the algorithm involves checking whether the numbers are single-digit. If so, the algorithm directly returns their product. Otherwise, the algorithm computes ...

The Ugly

The Good

The Bad

The recursive algorithm takes as input two n-digit numbers x and y and returns their product $x \cdot y$.

The base case of the algorithm involves checking whether the numbers are single-digit. If so, the algorithm directly returns their product. Otherwise, the algorithm computes ...

The Ugly

```
def Karatsuba(x, y)

  if len(str(x))==1
  or len(str(y))==1:
    return x * y

  m = max( len(str(x)),
           len(str(y)))

  m2 = m // 2

  a = x // 10**(m2)

  b = x % 10**(m2)
```

The Good

input: two n digit numbers x, y

output: the product $x \cdot y$

if $n=1$ return $x \cdot y$

else

$a, b :=$ first and second halves of x

$c, d :=$ " " " y

recursively compute $a \cdot c$, $a \cdot d$, $b \cdot c$, and $b \cdot d$

Compute $10^{n} \cdot ac + 10^{n/2}(ad + bc) + bd$

using grade-school addition and return the result

The Bad

The recursive algorithm takes as input two n-digit numbers x and y and returns their product $x \cdot y$.

The base case of the algorithm involves checking whether the numbers are single-digit. If so, the algorithm directly returns their product. Otherwise, the algorithm computes ...

The Ugly

```
def Karatsuba(x, y)

  if len(str(x))==1
  or len(str(y))==1:
    return x * y

  m = max( len(str(x)),
           len(str(y)))

  m2 = m // 2

  a = x // 10**(m2)

  b = x % 10**(m2)
```

The Good

input: two n digit numbers x, y

output: the product $x \cdot y$

if $n=1$ return $x \cdot y$

else

$a, b :=$ first and second halves of x

$c, d :=$ " " " y

recursively compute $a \cdot c$, $a \cdot d$, $b \cdot c$, and $b \cdot d$

Compute $10^{n} \cdot ac + 10^{n/2}(ad + bc) + bd$

using grade-school addition and return the result

How many basic operations does the recursive algorithm perform?

The Good

input: two n digit numbers x, y

output: the product $x \cdot y$

if $n = 1$ return $x \cdot y$

else

$a, b :=$ first and second halves of x

$c, d :=$ " " " y

recursively compute $a \cdot c$, $a \cdot d$, $b \cdot c$, and $b \cdot d$

Compute $10^{n} \cdot ac + 10^{n/2}(ad + bc) + bd$

using grade-school addition and return the result

How many basic operations does the recursive algorithm perform?

Coming up shortly!

The Good

input: two n digit numbers x, y

output: the product $x \cdot y$

if $n = 1$ return $x \cdot y$

else

$a, b :=$ first and second halves of x

$c, d :=$ " " " y

recursively compute $a \cdot c$, $a \cdot d$, $b \cdot c$, and $b \cdot d$

Compute $10^n \cdot ac + 10^{n/2}(ad + bc) + bd$ using grade-school addition and return the result

How many basic operations does the recursive algorithm perform?

Coming up shortly!

First, let's demystify Karatsuba

KARATSUBA MULTIPLICATION

$x = \overbrace{56}^{a}\ \overbrace{78}^{b}$

$y = \overbrace{12}^{c}\ \overbrace{34}^{d}$

① compute $a \cdot c = 56 \times 12 = 672$

② compute $b \cdot d = 78 \times 34 = 2652$

③ compute $(a+b) \cdot (c+d) = 134 \times 46 = 6164$

④ compute ③ − ② − ① = 2840

⑤ compute

$$
\begin{array}{rl}
6720000 & \text{from ①} \\
2652 & \text{from ②} \\
2840000 & \text{from ④} \\
\hline
7006652 & = x \cdot y
\end{array}
$$

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n} ac + 10^{n/2}(ad + bc) + bd$$

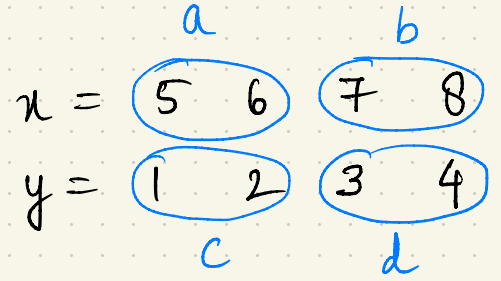
$x =$ (5 6) (7 8) — a b

$y =$ (1 2) (3 4) — c d

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n}\, ac + 10^{n/2}(ad + bc) + bd$$



Recursive algo. needs four recursive calls

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n}\, ac + 10^{n/2}(ad + bc) + bd$$

$x = 5\ 6\ 7\ 8$ (a = 5 6, b = 7 8)

$y = 1\ 2\ 3\ 4$ (c = 1 2, d = 3 4)

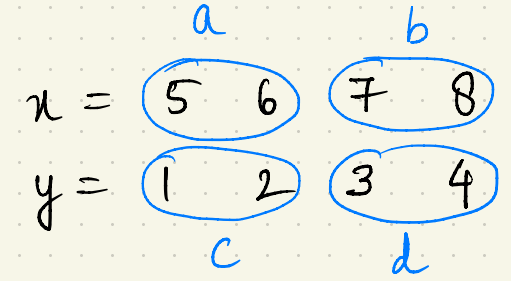
Recursive algo. needs four recursive calls

Can we get away with only three recursive calls ?

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n}\, ac + 10^{n/2}(ad + bc) + bd$$



Recursive algo. needs four recursive calls

Can we get away with only three recursive calls ?

YES !

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n} ac + 10^{n/2} (ad + bc) + bd$$

$x = 5\ 6\ 7\ 8$ (a = 5 6, b = 7 8)

$y = 1\ 2\ 3\ 4$ (c = 1 2, d = 3 4)

we only care about the Sum of ad and bc

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n} ac + 10^{n/2} (ad + bc) + bd$$

$x = 5\ 6\ 7\ 8$ (a = 5 6, b = 7 8)

$y = 1\ 2\ 3\ 4$ (c = 1 2, d = 3 4)

we only care about the sum of ad and bc

$$ad + bc = (a+b)(c+d) - ac - bd$$

Gauss' trick

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n} ac + 10^{n/2} (ad + bc) + bd$$

$x = \overbrace{5\ 6}^{a}\ \overbrace{7\ 8}^{b}$

$y = \overbrace{1\ 2}^{c}\ \overbrace{3\ 4}^{d}$

we only care about the sum of ad and bc

$$\underbrace{ad + bc}_{\text{two recursive calls}} = \underbrace{(a+b)(c+d)}_{\text{one recursive call}} - ac - bd$$

Gauss' trick

already have these

KARATSUBA MULTIPLICATION

Recall from recursive algorithm :

$$x \cdot y = 10^{n} ac + 10^{n/2}(ad + bc) + bd$$

$x = 5\ 6\ 7\ 8$ (a = 5 6, b = 7 8)

$y = 1\ 2\ 3\ 4$ (c = 1 2, d = 3 4)

we only care about the sum of ad and bc

$$\underbrace{ad} + \underbrace{bc} = \underbrace{(a+b)(c+d)} - ac - bd$$

Gauss' trick

two recursive calls

one recursive call

already have these

Karatsuba only needs three recursive calls!

(and some additions)

KARATSUBA MULTIPLICATION

$x = \overbrace{56}^{a}\ \overbrace{78}^{b}$

$y = \overbrace{12}^{c}\ \overbrace{34}^{d}$

① compute $a \cdot c = 56 \times 12 = 672$

② compute $b \cdot d = 78 \times 34 = 2652$

③ compute $(a+b) \cdot (c+d) = 134 \times 46 = 6164$

④ compute ③ − ② − ① = 2840

⑤ compute

$$
\begin{array}{rl}
6720000 & \text{from ①} \\
2652 & \text{from ②} \\
2840000 & \text{from ④} \\
\hline
7006652 & = x \cdot y
\end{array}
$$

KARATSUBA MULTIPLICATION

a b

$x = 5\ 6\ 7\ 8$

$y = 1\ 2\ 3\ 4$

c d

① compute $a \cdot c = 56 \times 12 = 672$

② compute $b \cdot d = 78 \times 34 = 2652$

③ compute $(a+b) \cdot (c+d) = 134 \times 46 = 6164$

④ compute ③ − ② − ① = 2840

Gauss' trick!

⑤ compute

$$
\begin{array}{rl}
6\,7\,2\,0\,0\,0\,0 & \text{from } ① \\
2\,6\,5\,2 & \text{from } ② \\
2\,8\,4\,0\,0\,0 & \text{from } ④ \\
\hline
7\,0\,0\,6\,6\,5\,2 & = x \cdot y
\end{array}
$$

KARATSUBA PSEUDOCODE

input : two n-digit positive integers x and y

output : product $x \cdot y$

if $n = 1$

    Compute $x \cdot y$ in one step and return the result

else

    a, b := first and second halves of x

    c, d := " " " y

    recursively compute $a \cdot c$, $b \cdot d$, and $(a+b) \cdot (c+d)$

    Compute $z := (a+b) \cdot (c+d) - ac - bd$ via grade-school addition

    return $10^{n} \cdot a \cdot c + 10^{n/2} \cdot z + bd$

STORY SO FAR

Grade-school multiplication $\leq 9n^2$ basic operations

Recursive algorithm (4 calls) ?

Karatsuba algorithm (3 calls) ?

STORY SO FAR

Grade-school multiplication $\leq 9n^2$ basic operations

Recursive algorithm (4 calls) ?

Karatsuba algorithm (3 calls) ?

Is Karatsuba faster than recursive and grade-school algorithms?

STORY SO FAR

Grade-school multiplication $\leq 9n^2$ basic operations

Recursive algorithm (4 calls) ?

Karatsuba algorithm (3 calls) ?

Is Karatsuba faster than recursive and grade-school algorithms?

We will come back to this question.

MERGE SORT

MERGE SORT

Canonical divide-and-conquer algorithm

- break the problem into smaller subproblems
- solve the subproblems recursively
- combine the solutions of subproblems

MERGE SORT

Canonical divide-and-conquer algorithm

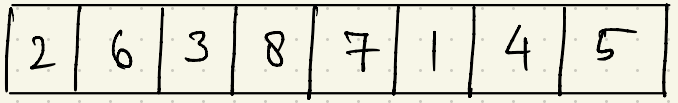
- break the problem into smaller subproblems
- solve the subproblems recursively
- combine the solutions of subproblems

Warm up for recursion tree and master method

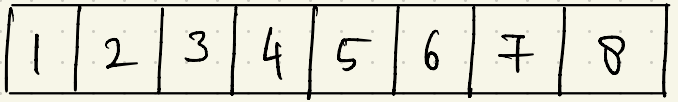
- useful in analyzing recursive multiplication and Karatsuba's algorithms

SORTING

input : an array of numbers

eg., 

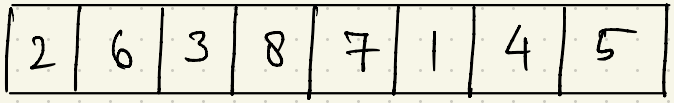
output : Same numbers, sorted in increasing order



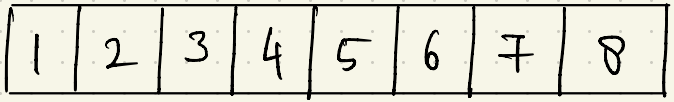
SORTING

input : an array of numbers

eg.,

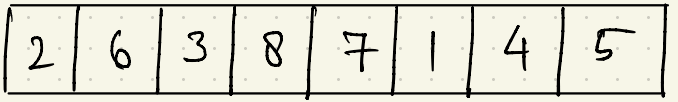


output : same numbers, sorted in increasing order

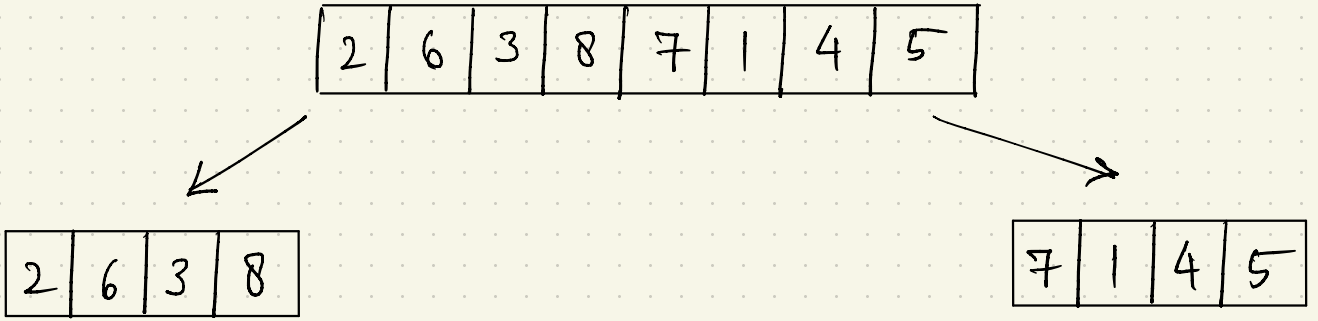


for simplicity, assume numbers to be distinct

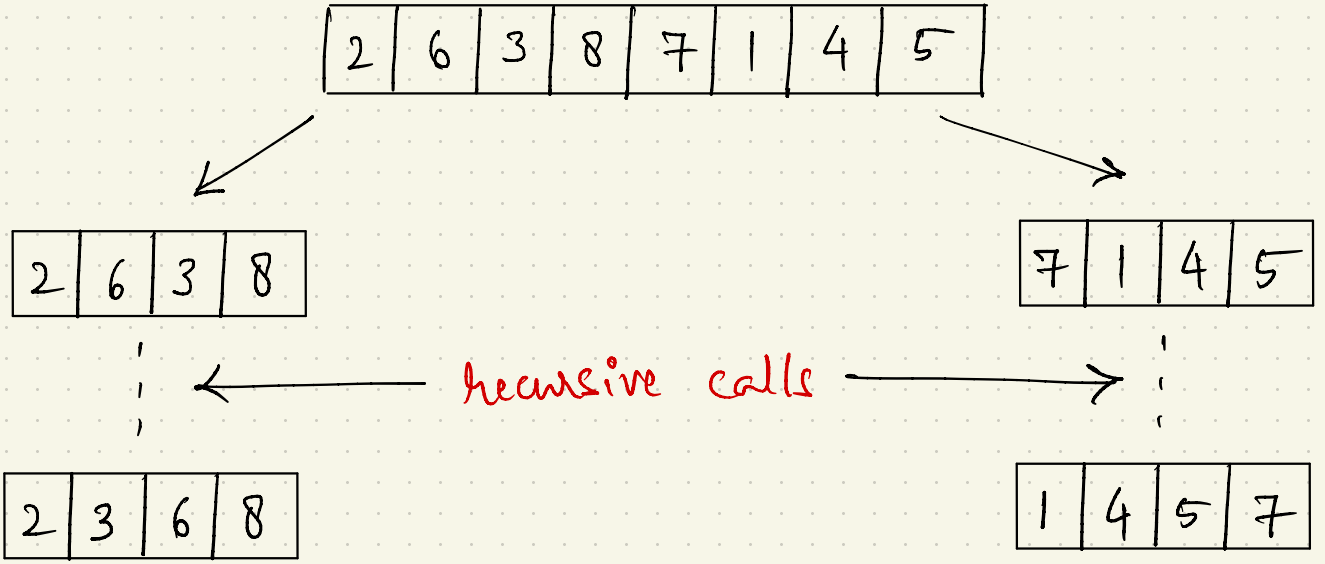
MERGE SORT : EXAMPLE



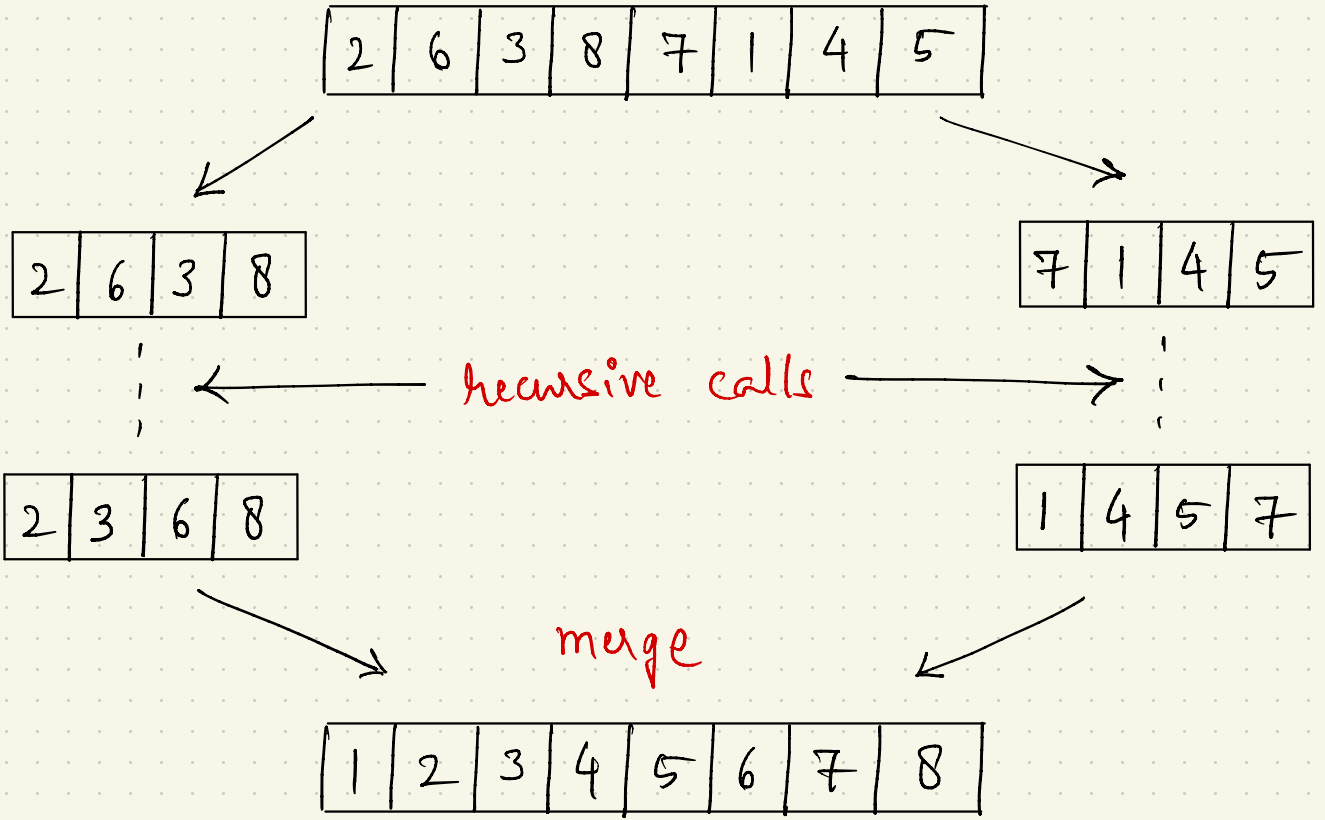
MERGE SORT : EXAMPLE



MERGE SORT : EXAMPLE



MERGE SORT : EXAMPLE



MERGE SORT : PSEUDOCODE

input : array A of n distinct integers

output: array with the same numbers sorted in increasing order

MERGE SORT : PSEUDOCODE

input : array A of n distinct integers

output: array with the same numbers sorted in increasing order

```
if n ≤ 1
        return A
else
        L := left half of A sorted recursively
        R := right    "          "          "

        return MERGE(L, R)
```

MERGE SUBROUTINE

input : sorted arrays L and R of length $n/2$ each

output: sorted array B of length n

MERGE SUBROUTINE

input : sorted arrays L and R of length $n/2$ each

output: sorted array B of length n

assume n is even

MERGE SUBROUTINE

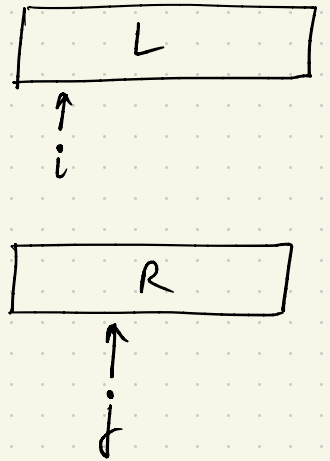
input : sorted arrays L and R of length $n/2$ each

output: sorted array B of length n

assume n is even

initialize $i := 1$

initialize $j := 1$



MERGE SUBROUTINE

input : sorted arrays L and R of length $n/2$ each

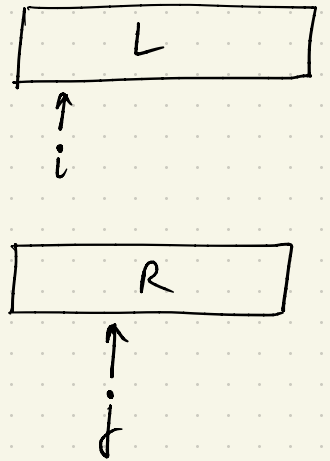
output: sorted array B of length n

assume n is even

initialize $i := 1$

initialize $j := 1$

for $k := 1$ to n



MERGE SUBROUTINE

input : sorted arrays L and R of length $n/2$ each

output: sorted array B of length n

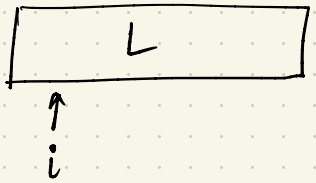
assume n is even

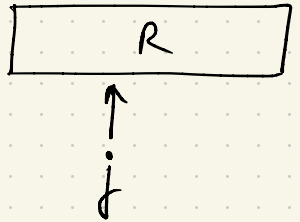
initialize $i := 1$

initialize $j := 1$

for $k := 1$ to n

if $L[i] < R[j]$ then





else

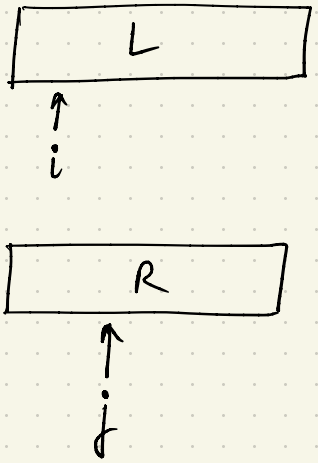
MERGE SUBROUTINE

input : sorted arrays L and R of length $n/2$ each

output: sorted array B of length n

assume n is even

```
initialize i := 1
initialize j := 1
for k := 1 to n
        if L[i] < R[j] then
                B[k] := L[i]
                increment i by 1
        else
                B[k] := R[j]
                increment j by 1
```



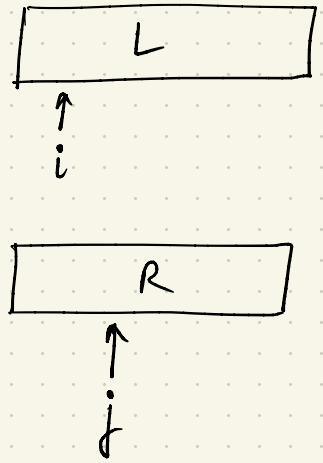
MERGE SUBROUTINE

input : sorted arrays L and R of length $n/2$ each

output: sorted array B of length n

assume n is even

```
initialize i := 1
initialize j := 1
for k := 1 to n
        if L[i] < R[j] then
                B[k] := L[i]
                increment i by 1
        else
                B[k] := R[j]
                increment j by 1
```



ignoring end cases when i or j falls off

MERGE SORT RUNNING TIME

What is the basic operation ?

MERGE SORT RUNNING TIME

What is the basic operation ?

- commonly, any pairwise comparison
- for now, a line-by-line implementation using a debugger