Data Structures

Divide and Conquer

Andres Mendez-Vazquez

November 19, 2016

Outline

Outline

Gauss and the Beginning

Carl Friedrich Gauss (1777–1855)

Outline

Gauss and the Beginning

Carl Friedrich Gauss (1777–1855)

He devised a way to multiply two imaginary numbers as

$$(a+bi)(c+di) = ac + (ad+bc)i - bd \tag{1}$$

By realizing that

$$bc + ad = (a+b)(c+d) - ac - bd \tag{2}$$

Thus minimizing the number of multiplications from four to three.

Actually

We can represent binary numbers like 1001 as $1000 + 01 = 2^2 \times 10 + 01$

Gauss and the Beginning

Carl Friedrich Gauss (1777–1855)

He devised a way to multiply two imaginary numbers as

$$(a + bi)(c + di) = ac + (ad + bc)i - bd \tag{1}$$

By realizing that

$$bc + ad = (a + b)(c + d) - ac - bd \tag{2}$$

Thus minimizing the number of multiplications from four to three.

Actually

We can represent binary numbers like 1001 as $1000 + 01 = 2^2 \times 10 + 01$

Gauss and the Beginning

Carl Friedrich Gauss (1777–1855)

He devised a way to multiply two imaginary numbers as

$$(a+bi)(c+di) = ac + (ad+bc)i - bd \tag{1}$$

By realizing that

$$bc + ad = (a+b)(c+d) - ac - bd \tag{2}$$

Thus minimizing the number of multiplications from four to three.

Actually

We can represent binary numbers like 1001 as $1000 + 01 = 2^2 \times 10 + 01$

Outline

Anatoly Alexeevitch Karatsuba (1937 - 2008)

At the Faculty of Mechanics and Mathematics of Moscow State University

He developed a trick to decrease the speed up of the multiplication of two numbers in binary representation.

Simply Realizing

$$x = x_L \circ x_R = 2^{n/2} x_L + x_R \approx i x_L + x_R$$

Anatoly Alexeevitch Karatsuba (1937 - 2008)

At the Faculty of Mechanics and Mathematics of Moscow State University

He developed a trick to decrease the speed up of the multiplication of two numbers in binary representation.

Simply Realizing

$$x = x_L \circ x_R = 2^{n/2} x_L + x_R \approx i x_L + x_R$$

Thus

We can represent numbers x, y as

- $x = x_L \circ x_R = 2^{n/2} x_L + x_R$
- $y = y_L \circ y_R = 2^{n/2} y_L + y_R$

Thus, the multiplication can be found by using

$$xy = \left(2^{n/2} x_L + x_R\right)\left(2^{n/2} y_L + y_R\right) = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \quad (3)$$

However

if we use the Gauss's trick, we only need $x_L y_L$, $x_R y_R$, $(x_L + x_R)(y_L + y_R)$ to calculate the multiplication:

- $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

Thus

We can represent numbers x, y as

- $x = x_L \circ x_R = 2^{n/2} x_L + x_R$
- $y = y_L \circ y_R = 2^{n/2} y_L + y_R$

Thus, the multiplication can be found by using

$$xy = \left(2^{n/2} x_L + x_R\right)\left(2^{n/2} y_L + y_R\right) = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \quad (3)$$

However

if we use the Gauss's trick, we only need $x_L y_L, x_R y_R, (x_L + x_R)(y_L + y_R)$ to calculate the multiplication:

- $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

Thus

We can represent numbers x, y as

- $x = x_L \circ x_R = 2^{n/2} x_L + x_R$
- $y = y_L \circ y_R = 2^{n/2} y_L + y_R$

Thus, the multiplication can be found by using

$$xy = \left(2^{n/2} x_L + x_R\right)\left(2^{n/2} y_L + y_R\right) = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R \quad (3)$$

However

if we use the Gauss's trick, we only need $x_L y_L$, $x_R y_R$, $(x_L + x_R)(y_L + y_R)$ to calculate the multiplication:

- $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$

Now, You have this...

We have that

xy can be calculated using $x_L x_L$, $x_L y_R$, $x_R y_L$ and $x_R y_R$.

An each of this multiplications

$x_L x_L$, $x_L y_R$, $x_R y_L$ and $x_R y_R$ can be calculated in a similar way

Recursion

This is know as a Recursive Procedure!!!

Now, You have this...

We have that

xy can be calculated using $x_L x_L$, $x_L y_R$, $x_R y_L$ and $x_R y_R$.

An each of this multiplications

$x_L x_L$, $x_L y_R$, $x_R y_L$ and $x_R y_R$ can be calculated in a similar way

Recursion

This is know as a Recursive Procedure!!!

Now, You have this...

We have that

xy can be calculated using $x_L x_L$, $x_L y_R$, $x_R y_L$ and $x_R y_R$.

An each of this multiplications

$x_L x_L$, $x_L y_R$, $x_R y_L$ and $x_R y_R$ can be calculated in a similar way

Recursion

This is know as a Recursive Procedure!!!

Gauss' Algorithm.

The Recursive Method

```
function multiplyGauss(x,y)
    Input: Positive integers x and y, in binary
    Output: Their product

    n = max(size of x, size of y)

    if n = 1 then return xy

    xL , xR = leftmost ceil(n/2) , rightmost floor(n/2)
                                              bits of x
    yL , yR = leftmost ceil(n/2) , rightmost floor(n/2)
                                              bits of y

    P1 = multiplyGauss(xL,yL)
    P2 = multiplyGauss(xR,yR)
    P3 = multiply(xL + xR, yL + yR)

    return 2^n*P1 + 2^(n/2)*(P3 - P1 - P2) + P2
```

Outline

Complexities

Old Multiplication

$$T(n) = 4T\left(\frac{n}{2}\right) + \text{ Some Work} \tag{4}$$

New Multiplication with Gauss Trick

$$T(n) = 3T\left(n\right) + \text{ Some Work} \tag{5}$$

We will prove that

- For old style multiplications $O\left(n^2\right)$.
- For new style multiplications $O\left(n^{\log_2 3}\right)$

Complexities

Old Multiplication

$$T(n) = 4T\left(\frac{n}{2}\right) + \text{ Some Work} \tag{4}$$

New Multiplication with Gauss Trick

$$T(n) = 3T\left(n\right) + \text{ Some Work} \tag{5}$$

We will prove that

- For old style multiplications $O\left(n^2\right)$.
- For new style multiplications $O\left(n^{\log_2 3}\right)$

Complexities

Old Multiplication

$$T(n) = 4T\left(\frac{n}{2}\right) + \text{ Some Work} \tag{4}$$

New Multiplication with Gauss Trick

$$T(n) = 3T\left(n\right) + \text{ Some Work} \tag{5}$$

We will prove that

- For old style multiplications $O\left(n^2\right)$.
- For new style multiplications $O\left(n^{\log_2 3}\right)$

What is $T(n)$?

In recursive theory

$T(n)$ is the amount of work that a procedure is doing.

Something Notable

We can actually use this function to represent the recursive work of any procedure!!!

In the case of the old multiplication

$$x_{LyL}\ x_{LyR}\ x_{RyL}\ x_{RyR} \tag{6}$$

What is $T(n)$?

In recursive theory

$T(n)$ is the amount of work that a procedure is doing.

Something Notable

We can actually use this function to represent the recursive work of any procedure!!!

In the case of the old multiplication

$$x_{LyL}\ x_{LyR}\ x_{RyL}\ x_{RyR} \tag{6}$$

What is $T(n)$?

In recursive theory

$T(n)$ is the amount of work that a procedure is doing.

Something Notable

We can actually use this function to represent the recursive work of any procedure!!!

In the case of the old multiplication

$$x_L y_L \; x_L y_R \; x_R y_L \; x_R y_R \tag{6}$$

We have...

Four recursive multiplications

Clearly we are using half of the input!!!

Then

Each of this multiplications costs $n/2$

And as you go deeper in the recursion, we have the following sequence

$$n \longrightarrow \frac{n}{2} \longrightarrow \frac{n}{4} \longrightarrow \frac{n}{8} \longrightarrow \frac{n}{16} \longrightarrow \ldots \longrightarrow 1$$

We have...

Four recursive multiplications

Clearly we are using half of the input!!!

Then

Each of this multiplications costs $n/2$

And as you go deeper in the recursion, we have the following sequence

$$n \longrightarrow \frac{n}{2} \longrightarrow \frac{n}{4} \longrightarrow \frac{n}{8} \longrightarrow \frac{n}{16} \longrightarrow ... \longrightarrow 1$$

We have...

Four recursive multiplications

Clearly we are using half of the input!!!

Then

Each of this multiplications costs $n/2$

And as you go deeper in the recursion, we have the following sequence

$$n \Longrightarrow \frac{n}{2} \Longrightarrow \frac{n}{4} \Longrightarrow \frac{n}{8} \Longrightarrow \frac{n}{16} \Longrightarrow ... \Longrightarrow 1$$

How do we get these complexities?

In the case of these algorithms

Some Work $=$ Summations that take constant time or $O(1)$.

First

Let me draw the recursion tree

In a similar way you can obtain the

$$O\left(n^{\log_2 3}\right)$$

How do we get these complexities?

In the case of these algorithms

Some Work $=$ Summations that take constant time or $O(1)$.

First

Let me draw the recursion tree

In a similar way you can obtain the

$$O\left(n^{\log_2 3}\right)$$

How do we get these complexities?

In the case of these algorithms

Some Work $=$ Summations that take constant time or $O(1)$.

First

Let me draw the recursion tree

In a similar way you can obtain the

$O\left(n^{\log_2 3}\right)$

Epitaph

We can do divide and conquer

In a really un-clever way!!!

Or we can go and design something better

Thus, improving speedup!!!

The difference between

- A great design...
- Or a crappy job...

Epitaph

We can do divide and conquer

In a really un-clever way!!!

Or we can go and design something better

Thus, improving speedup!!!

The difference between

- A great design...
- Or a crappy job...

Epitaph

We can do divide and conquer

In a really un-clever way!!!

Or we can go and design something better

Thus, improving speedup!!!

The difference between

- A great design...
- Or a crappy job...

Outline

Recursion is the base of Divide and Conquer

This is the natural way we do many things

We always attack smaller versions first of the large one!!!

Stephen Cole Kleene

- He defined the basics about the use of recursion.

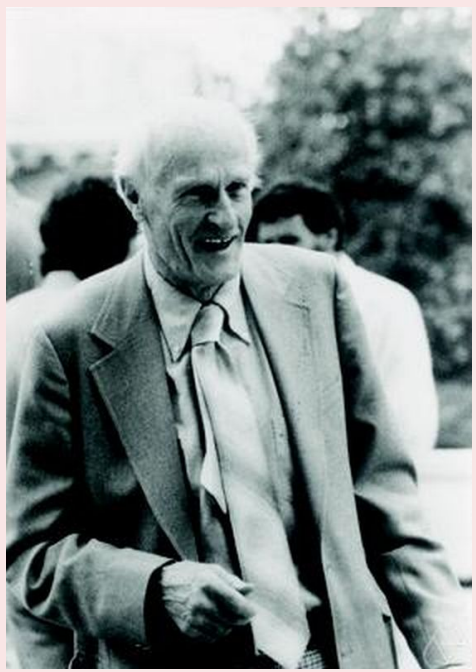
Recursion is the base of Divide and Conquer

This is the natural way we do many things

We always attack smaller versions first of the large one!!!

Stephen Cole Kleene

- He defined the basics about the use of recursion.



Kleene and Company

Some facts about him

- Stephen Cole Kleene (January 5, 1909 – January 25, 1994) was an American mathematician.
- One of the students of Alonzo Church!!!
  - Church is best known for the lambda calculus, Church–Turing thesis and proving the undecidability of the use of an algorithm to say Yes(Valid) or No(No Valid) to a first order logic statement on a FOL System (Proposed by David Hilbert).

Kleene and Company

Some facts about him

- Stephen Cole Kleene (January 5, 1909 – January 25, 1994) was an American mathematician.
- One of the students of Alonzo Church!!!
  - Church is best known for the lambda calculus, Church–Turing thesis and proving the undecidability of the use of an algorithm to say Yes(Valid) or No(No Valid) to a first order logic statement on a FOL System (Proposed by David Hilbert).

Recursion Theory

- Kleene, along with Alan Turing, Emil Post, and others, is best known as a founder of the branch of mathematical logic known as recursion theory.
- This theory subsequently helped to provide the foundations of theoretical computer science.

Kleene and Company

Some facts about him

- Stephen Cole Kleene (January 5, 1909 – January 25, 1994) was an American mathematician.
- One of the students of Alonzo Church!!!
  - Church is best known for the lambda calculus, Church–Turing thesis and proving the undecidability of the use of an algorithm to say Yes(Valid) or No(No Valid) to a first order logic statement on a FOL System (Proposed by David Hilbert).

Recursion Theory

- Kleene, along with Alan Turing, Emil Post, and others, is best known as a founder of the branch of mathematical logic known as recursion theory.
- This theory subsequently helped to provide the foundations of theoretical computer science.

Kleene and Company

Some facts about him

- Stephen Cole Kleene (January 5, 1909 – January 25, 1994) was an American mathematician.
- One of the students of Alonzo Church!!!
  - Church is best known for the lambda calculus, Church–Turing thesis and proving the undecidability of the use of an algorithm to say Yes(Valid) or No(No Valid) to a first order logic statement on a FOL System (Proposed by David Hilbert).

Recursion Theory

- Kleene, along with Alan Turing, Emil Post, and others, is best known as a founder of the branch of mathematical logic known as recursion theory.
- This theory subsequently helped to provide the foundations of theoretical computer science.

Kleene and Company

Some facts about him

- Stephen Cole Kleene (January 5, 1909 – January 25, 1994) was an American mathematician.
- One of the students of Alonzo Church!!!
  - Church is best known for the lambda calculus, Church–Turing thesis and proving the undecidability of the use of an algorithm to say Yes(Valid) or No(No Valid) to a first order logic statement on a FOL System (Proposed by David Hilbert).

Recursion Theory

- Kleene, along with Alan Turing, Emil Post, and others, is best known as a founder of the branch of mathematical logic known as recursion theory.
- This theory subsequently helped to provide the foundations of theoretical computer science.

Outline

Basics

A clear application of Divide and Conquer

First, we explain the Merging

Merging

The key to Merge Sort is merging two sorted lists into one, such that if you have two lists:

$$X = (x_1 \leq x_2 \leq ... \leq x_m) \text{ and } Y = (y_1 \leq y_2 \leq ... \leq y_n)$$

into a resulting list $Z = (z_1 \leq z_2 \leq ... \leq z_{m+n})$

Basics

A clear application of Divide and Conquer

First, we explain the Merging

Merging

The key to Merge Sort is merging two sorted lists into one, such that if you have two lists:

$$X = (x_1 \leq x_2 \leq ... \leq x_m) \text{ and } Y = (y_1 \leq y_2 \leq ... \leq y_n)$$

into a resulting list $Z = (z_1 \leq z_2 \leq ... \leq z_{m+n})$

Outline

Example: Merging

We have the following

X:

3	10	11	54

Y:

1	5	25	75

Result:

Example: Merging

We have the following

X:	3	10	11	54

Y:		5	25	75

Result:	1							

Example: Merging

We have the following

X:

	10	11	54

Y:

	5	25	75

Result:

1	3						

Example: Merging

We have the following

X:

	10	11	54

Y:

		25	75

Result:

1	3	5					

Example: Merging

We have the following

X:

		11	54

Y:

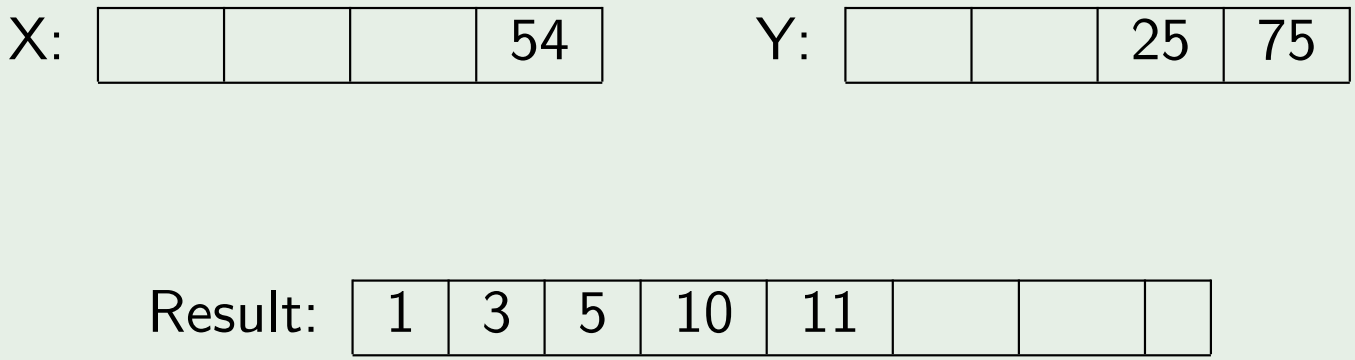
		25	75

Result:

1	3	5	10				

Example: Merging

We have the following



Example: Merging

We have the following

X:

			54

Y:

			75

Result:

1	3	5	10	11	25		

Example: Merging

We have the following

X: | | | | |

Y: | | | | 75 |

Result: | 1 | 3 | 5 | 10 | 11 | 25 | 54 | |

Example: Merging

We have the following

X: | | | | |

Y: | | | | |

Result:

1	3	5	10	11	25	54	75

Outline

Divide and Conquer

Thus, basic step

Merging a two lists of one element each is the same as sorting them.

Strategy

Merge sort divides up an unsorted list until the above condition is met and then sorts the divided parts back together in pairs.

The Recursion

Specifically this can be done by recursively dividing the unsorted list in half, merge sorting the right side then the left side and then merging the right and left back together.

Divide and Conquer

Thus, basic step

Merging a two lists of one element each is the same as sorting them.

Strategy

Merge sort divides up an unsorted list until the above condition is met and then sorts the divided parts back together in pairs.

The Recursion

Specifically this can be done by recursively dividing the unsorted list in half, merge sorting the right side then the left side and then merging the right and left back together.

Divide and Conquer

Thus, basic step

Merging a two lists of one element each is the same as sorting them.

Strategy

Merge sort divides up an unsorted list until the above condition is met and then sorts the divided parts back together in pairs.

The Recursion

Specifically this can be done by recursively dividing the unsorted list in half, merge sorting the right side then the left side and then merging the right and left back together.

Outline

Mergesort algorithm.

Pseudo-Code

MergeSort(List)

1. mid = len(alist)//2
2. lefthalf = alist[:mid]
3. righthalf = alist[mid:]
4. MergeSort(lefthalf)
5. MergeSort(righthalf)
6. Merging(List)

Mergesort algorithm.

The merge method

```
function multiplyGauss(x,y)
    Input: Positive integers x and y, in binary
    Output: Their product

    n = max(size of x, size of y)

    if n = 1 then return xy

    xL , xR = leftmost ceil(n/2) , rightmost floor(n/2)
                                              bits of x
    yL , yR = leftmost ceil(n/2) , rightmost floor(n/2)
                                              bits of y

    P1 = multiplyGauss(xL,yL)
    P2 = multiplyGauss(xR,yR)
    P3 = multiply(xL + xR, yL + yR)

    return 2^n*P1 + 2^(n/2)*(P3 - P1 - P2) + P2
```

Outline

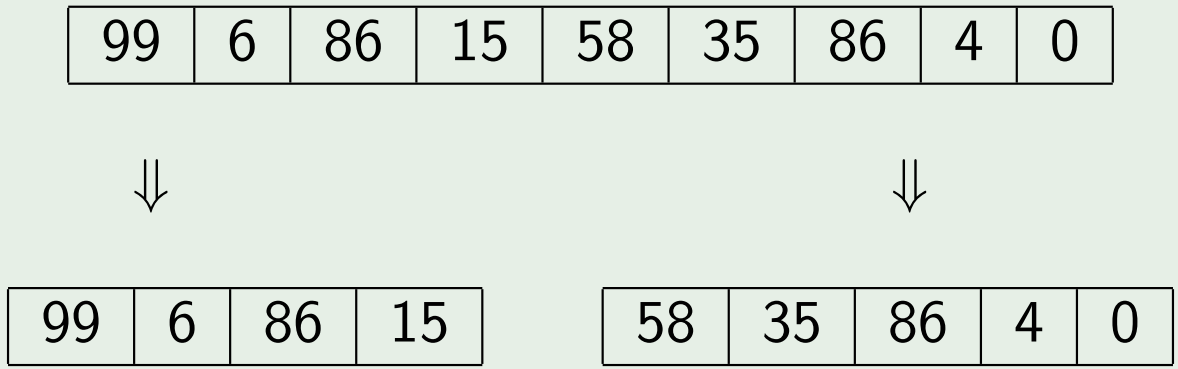
Example

The array

99	6	86	15	58	35	86	4	0

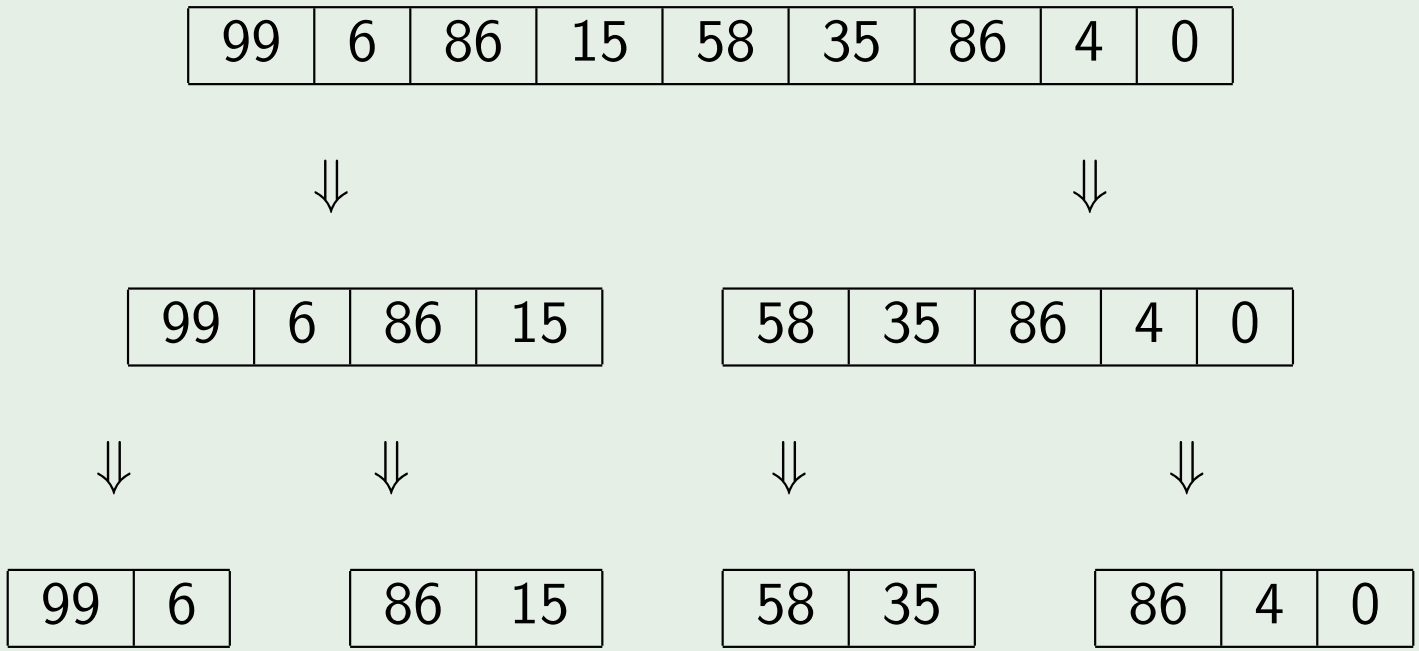
Example

First Split



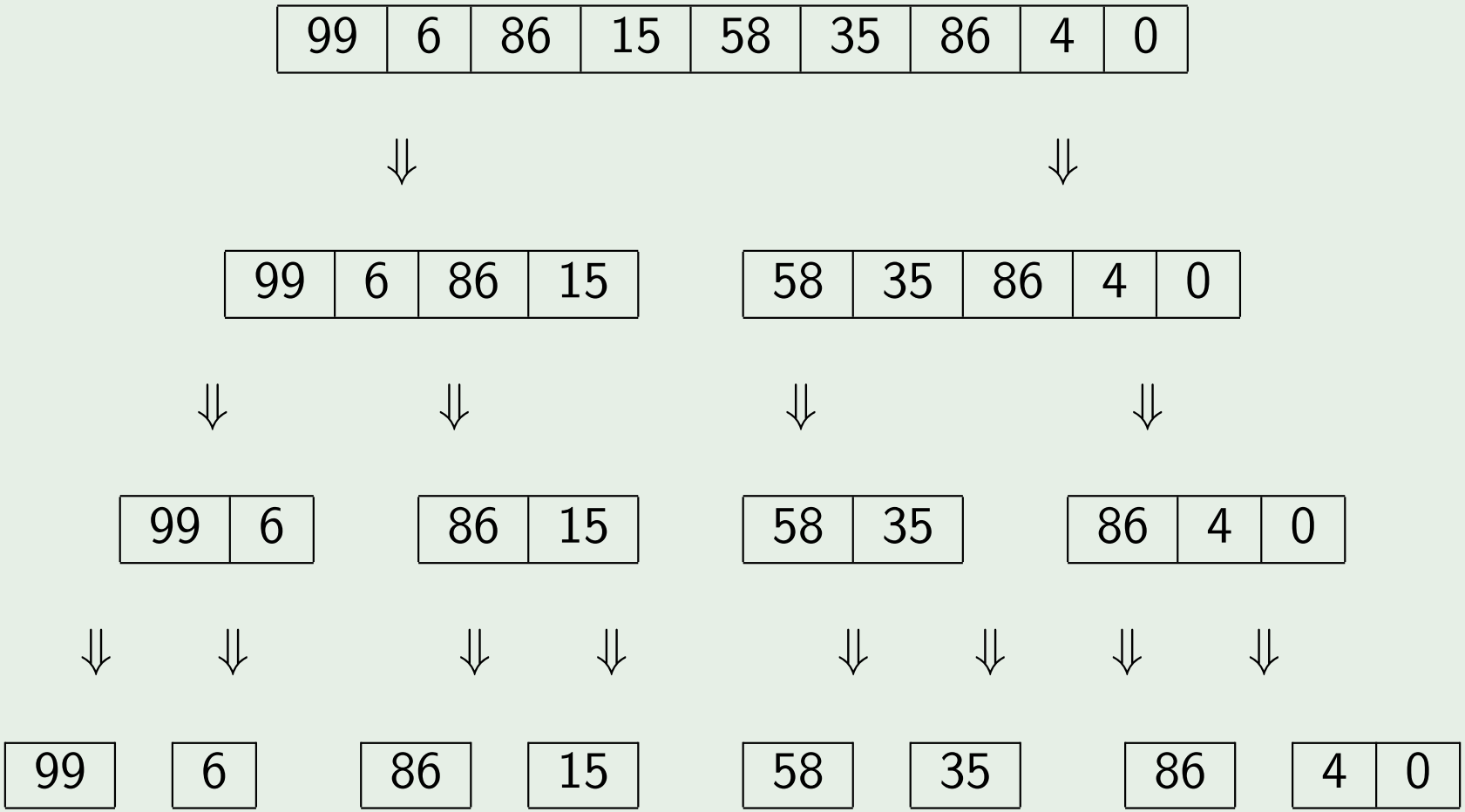
Example

Second Split



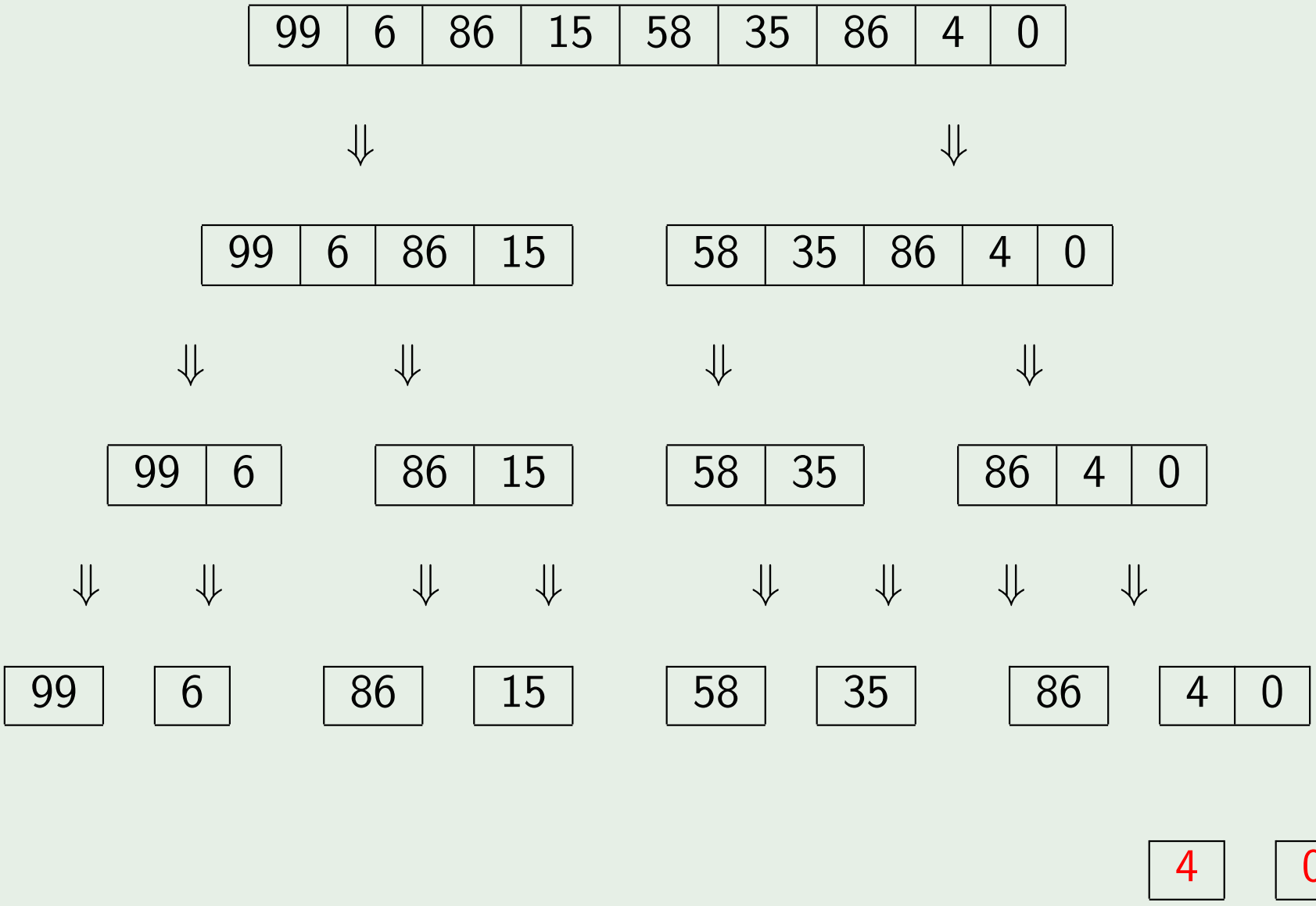
Example

Third Split



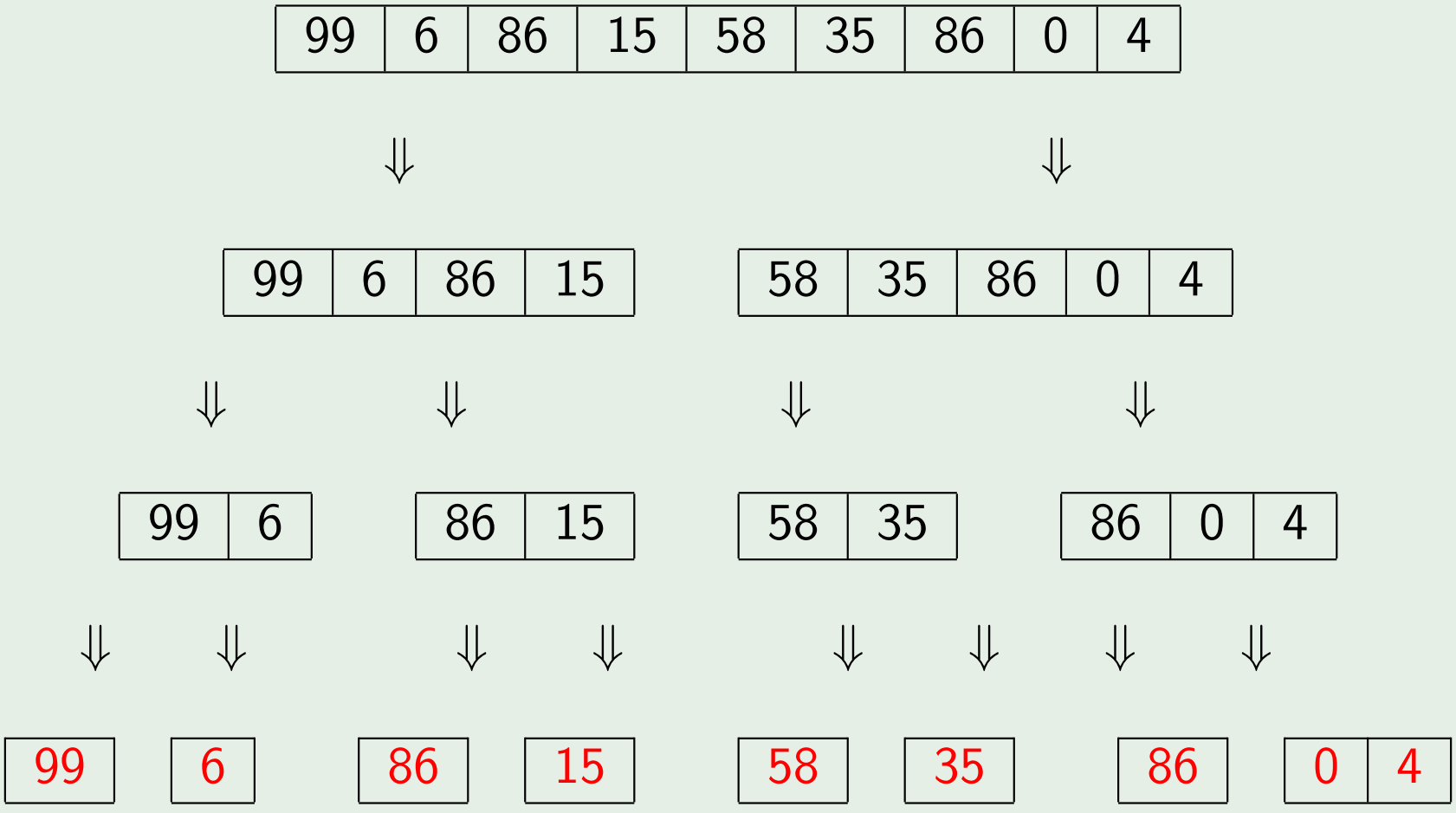
Example

Fourth Split



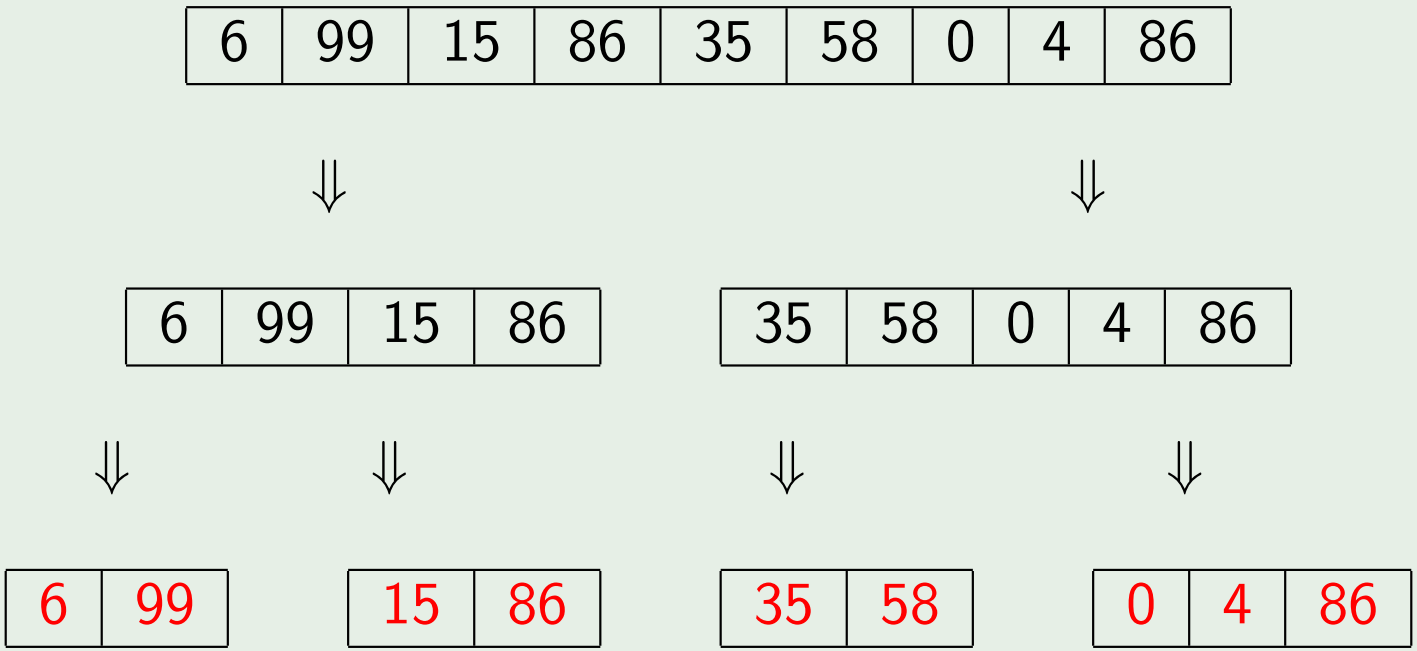
Example

Merging



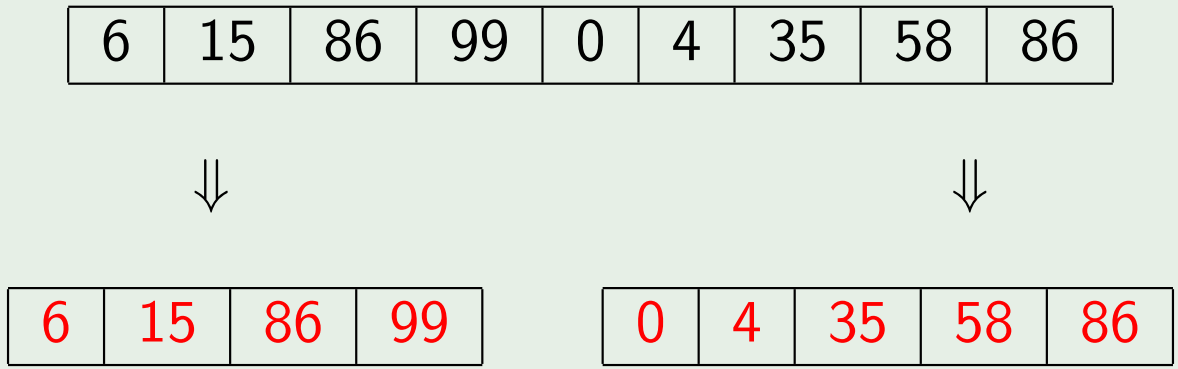
Example

Merging



Example

Merging



Example

Merging

0	4	6	15	35	58	86	86	99

Outline

Complexity

We have the following

The Double Memory Merge Sort runs $O(N \log N)$ for all cases? (N being the number of numbers to be sorted) because of its Divide and Conquer approach.

Actually, we can prove the following recursion

$$T(N) = 2T\left(\frac{N}{2}\right) + N = O(N \log N)$$

Complexity

We have the following

The Double Memory Merge Sort runs $O(N \log N)$ for all cases? (N being the number of numbers to be sorted) because of its Divide and Conquer approach.

Actually, we can prove the following recursion

$$T(N) = 2T\left(\frac{N}{2}\right) + N = O(N \log N)$$

Outline

Who invented Quicksort?

Imagine this

The quicksort algorithm was developed in 1960 by Tony Hoare (He has a postgraduate certificate in Statistics) while in the Soviet Union, as a visiting student at Moscow State University.

Why?

At that time, Hoare worked in a project on machine translation for the National Physical Laboratory.

To do

He developed the algorithm in order to sort the words to be translated.

Who invented Quicksort?

Imagine this

The quicksort algorithm was developed in 1960 by Tony Hoare (He has a postgraduate certificate in Statistics) while in the Soviet Union, as a visiting student at Moscow State University.

Why?

At that time, Hoare worked in a project on machine translation for the National Physical Laboratory.

To do

He developed the algorithm in order to sort the words to be translated.

Who invented Quicksort?

Imagine this

The quicksort algorithm was developed in 1960 by Tony Hoare (He has a postgraduate certificate in Statistics) while in the Soviet Union, as a visiting student at Moscow State University.

Why?

At that time, Hoare worked in a project on machine translation for the National Physical Laboratory.

To do

He developed the algorithm in order to sort the words to be translated.

Outline

Definitions

Divide

Partition (rearrange) the array $A[p, ..., r]$ into two (possibly empty) sub-arrays $A[p, ..., q-1]$ and $A[q+1, ..., r]$ such that each element of $A[p, ..., q-1]$ is less than or equal to $A[q]$, which is, in turn, less than or equal to each element of $A[q+1, ..., r]$. Compute the index q as part of this partitioning procedure.

Conquer

Sort the two sub-arrays $A[p, ..., q-1]$ and $A[q+1, ..., r]$ by recursive calls to quicksort.

Combine

Since the sub-arrays are sorted in place, no work is needed to combine them: the entire array $A[p, ..., r]$ is now sorted.

Definitions

Divide

Partition (rearrange) the array $A[p, ..., r]$ into two (possibly empty) sub-arrays $A[p, ..., q-1]$ and $A[q+1, ..., r]$ such that each element of $A[p, ..., q-1]$ is less than or equal to $A[q]$, which is, in turn, less than or equal to each element of $A[q+1, ..., r]$. Compute the index q as part of this partitioning procedure.

Conquer

Sort the two sub-arrays $A[p, ..., q-1]$ and $A[q+1, ..., r]$ by recursive calls to quicksort.

Combine

Since the sub-arrays are sorted in place, no work is needed to combine them: the entire array $A[p, ..., r]$ is now sorted.

Definitions

Divide

Partition (rearrange) the array $A[p, ..., r]$ into two (possibly empty) sub-arrays $A[p, ..., q-1]$ and $A[q+1, ..., r]$ such that each element of $A[p, ..., q-1]$ is less than or equal to $A[q]$, which is, in turn, less than or equal to each element of $A[q+1, ..., r]$. Compute the index q as part of this partitioning procedure.

Conquer

Sort the two sub-arrays $A[p, ..., q-1]$ and $A[q+1, ..., r]$ by recursive calls to quicksort.

Combine

Since the sub-arrays are sorted in place, no work is needed to combine them: the entire array $A[p, ..., r]$ is now sorted.

Outline

Quicksort Algorithm

Quicksort Algorithm

Quicksort(A, p, r)

1. if $p < r$
2. $\quad q = Partition\,(A, p, r)$
3. $\quad$ **Quicksort**$(A, p, q-1)$
4. $\quad$ **Quicksort**$(A, q+1, r)$

Partition Algorithm

Quicksort Partition

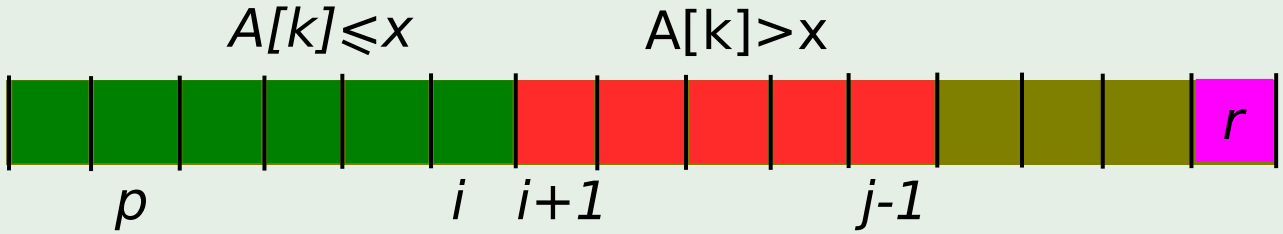
Partition(A, p, r)

1. $x = A[r]$
2. $i = p - 1$
3. for $j = p$ to $r - 1$
4. if $A[j] \leq x$
5. $i = i + 1$
6. exchange $A[i]$ with $A[j]$
7. exchange $A[i + 1]$ with $A[r]$
8. return $i + 1$

Quicksort: What is the Invariance?

Loop Invariance

1. **If** $p \leq k \leq$ **i, then** $A[k] \leq x$**.**
2. **If** $i+1 \leq k \leq j-1$**, then** $A[k] > x$**.**
3. **If** $k = r$ **, then** $A[k] = x$**.**
4. **UNKNOWN**



Outline

Complexity

It is possible to prove that the expected complexity

$$O(N \log N)$$

However, the worst case is

$$O\left(N^2\right)$$

Complexity

It is possible to prove that the expected complexity

$$O(N \log N)$$

However, the worst case is

$$O\left(N^2\right)$$

Outline

Example

Lets Sort the following sequence

99	6	86	15	58	35	86	4	10

Example

Lets Sort the following sequence

99	6	86	15	58	35	86	4	**10**

Example

Lets Sort the following sequence

6	99	86	15	58	35	86	4	10

Example

Lets Sort the following sequence

6	4	86	15	58	35	86	99	10

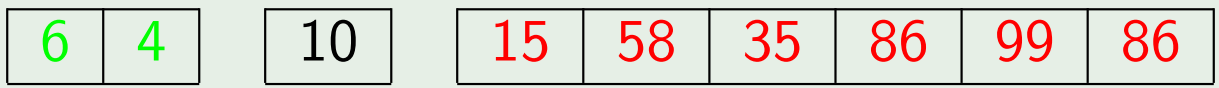
Example

Exchange pivot with the element at $i + 1$

6	4	10	15	58	35	86	99	86

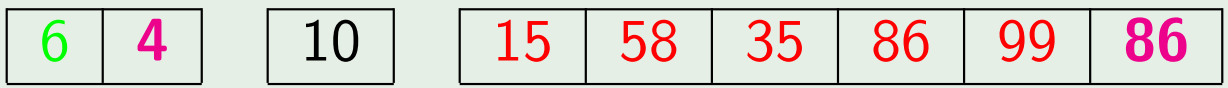
Example

Call the two recursions



Example

Thus, we have that



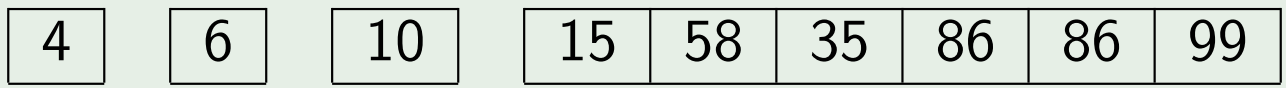
Example

Thus, we have that a simple exchange for the left part

4	6	10	15	58	35	86	99	**86**

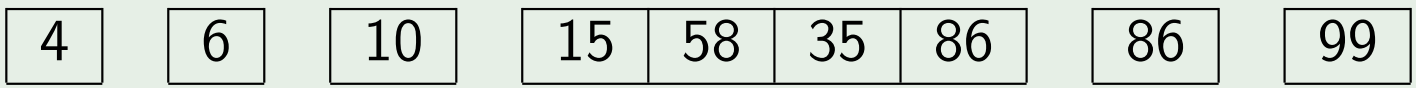
Example

The right part after exchanging the pivot



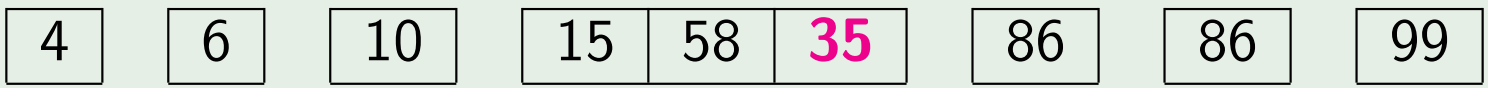
Example

Split



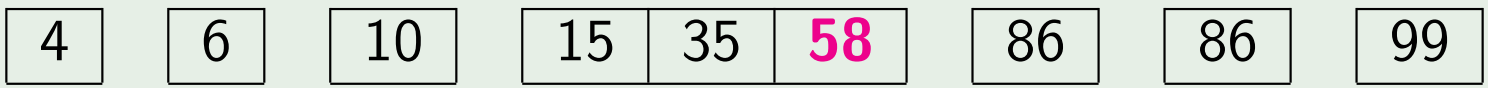
Example

Split



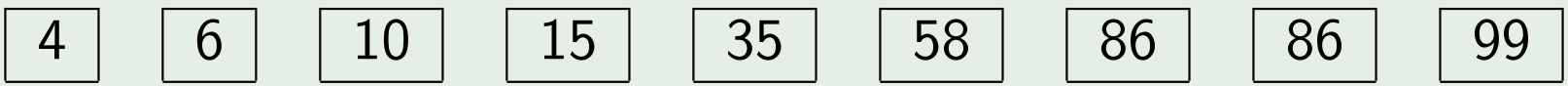
Example

Exchange



Example

Split



Example

Finally

4	6	10	15	35	58	86	86	99

Outline

Choosing the Pivot

Something Notable

- The popular, uninformed choice is to use the first or last element as the pivot.
- This is acceptable if the input is random.

Choosing the Pivot

Something Notable

- The popular, uninformed choice is to use the first or last element as the pivot.
- This is acceptable if the input is random.

However

if the input is presorted or in reverse order, then the pivot provides a poor partition, because either all the elements go into $LeftPartition$ or they go into $RightPartiton$.

Choosing the Pivot

Something Notable

- The popular, uninformed choice is to use the first or last element as the pivot.
- This is acceptable if the input is random.

However

if the input is presorted or in reverse order, then the pivot provides a poor partition, because either all the elements go into $LeftPartition$ or they go into $RightPartiton$.

This leads to

Quadratic Time!!! $O(n^2)$.

Choosing the Pivot

Something Notable

- The popular, uninformed choice is to use the first or last element as the pivot.
- This is acceptable if the input is random.

However

if the input is presorted or in reverse order, then the pivot provides a poor partition, because either all the elements go into $LeftPartition$ or they go into $RightPartiton$.

This leads to

Quadratic Time!!! $O\left(n^2\right)$.

What do we do?

A simpler strategy

- You can choose each of the pivots in a random way!!!
- Thus, you minimize the probability of having the worst behavior.
- However, a random number generator is an expansive commodity.

What do we do?

A simpler strategy

- You can choose each of the pivots in a random way!!!
- Thus, you minimize the probability of having the worst behavior.
- However, a random number generator is an expansive commodity.

A Better Strategy

Median-of Three Partitioning

What do we do?

A simpler strategy

- You can choose each of the pivots in a random way!!!
- Thus, you minimize the probability of having the worst behavior.
- However, a random number generator is an expansive commodity.

A Better Strategy

Median-of Three Partitioning

What do we do?

A simpler strategy

- You can choose each of the pivots in a random way!!!
- Thus, you minimize the probability of having the worst behavior.
- However, a random number generator is an expansive commodity.

A Better Strategy

Median-of Three Partitioning

Thus, we have...

First

The median of a group of N numbers is the $\lceil N/2 \rceil$ th largest number.

Thus, we have...

First

The median of a group of N numbers is the $\lceil N/2 \rceil$ th largest number.

The median is a good choice

But is expansive to calculate.

Thus, we have...

First

The median of a group of N numbers is the $\lceil N/2 \rceil$ th largest number.

The median is a good choice

But is expansive to calculate.

Thus

A good estimate can be obtained by picking three elements randomly and using the median of these three as the pivot.

- Again the problem of using a random number generator.

Thus, we have...

First

The median of a group of N numbers is the $\lceil N/2 \rceil$ th largest number.

The median is a good choice

But is expansive to calculate.

Thus

A good estimate can be obtained by picking three elements randomly and using the median of these three as the pivot.

- Again the problem of using a random number generator.

Better Strategy

Use the Median of the left, right and center elements

For example, with input 8, 1, 4, 9, 6, 3, 5, 2, 7, 0

Better Strategy

Use the Median of the left, right and center elements

For example, with input 8, 1, 4, 9, 6, 3, 5, 2, 7, 0

We have that

1. Left $= 1$ and the element is 8
2. Right $= 10$ and the element is 0

Better Strategy

Use the Median of the left, right and center elements

For example, with input 8, 1, 4, 9, 6, 3, 5, 2, 7, 0

We have that

1. Left $= 1$ and the element is 8
2. Right $= 10$ and the element is 0

We calculate

$\left\lfloor \frac{(Left+Right)}{2} \right\rfloor$ which has element 6

Better Strategy

Use the Median of the left, right and center elements

For example, with input 8, 1, 4, 9, 6, 3, 5, 2, 7, 0

We have that

1. Left $= 1$ and the element is 8
2. Right $= 10$ and the element is 0

We calculate

$\left\lfloor \frac{(Left+Right)}{2} \right\rfloor$ which has element 6

Better Strategy

Thus the Pivot

It is $r = 6$

Better Strategy

Thus the Pivot

It is $r = 6$

We have that

Using median-of-three partitioning helps to eliminates the bad case for sorted inputs.

- The partitions become equal in this case.

Better Strategy

Thus the Pivot

It is $r = 6$

We have that

Using median-of-three partitioning helps to eliminates the bad case for sorted inputs.

- The partitions become equal in this case.

Actually Statistical Testing has proven that

The method reduces the number of comparisons by 14 percent.

Better Strategy

Thus the Pivot

It is $r = 6$

We have that

Using median-of-three partitioning helps to eliminates the bad case for sorted inputs.

- The partitions become equal in this case.

Actually Statistical Testing has proven that

The method reduces the number of comparisons by 14 percent.