# Merge Sort and Master's Theorem

Subhabrata Samajder

IIIT, Delhi

Summer Semester,

$5^{th}$ May, 2022

# Sorting

## Sorting Problem

Given $n$ numbers $x_1, x_2, \ldots, x_n$ arrange them in *increasing order*. In other words, find a sequence of distinct indices $1 \leq i_1, i_2, \ldots, i_n \leq n$, such that $x_{i_1} \leq x_{i_2} \leq \cdots \leq x_{i_n}$.

**In-place Sorting:** A sorting algorithm is called in-place if no additional memory is used besides the input array.

# Sorting

**Sorting Problem**

Given $n$ numbers $x_1, x_2, \ldots, x_n$ arrange them in *increasing order*. In other words, find a sequence of distinct indices $1 \leq i_1, i_2, \ldots, i_n \leq n$, such that $x_{i_1} \leq x_{i_2} \leq \cdots \leq x_{i_n}$.

**In-place Sorting:** A sorting algorithm is called in-place if no additional memory is used besides the input array.

# Mergesort Algorithm

# The Merge Algorithm

**Problem:** Suppose we have two lists $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$ of numbers sorted in an increasing order. Merge them to get a bigger sorted list.

# The Merge Algorithm

**Problem:** Suppose we have two lists $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$ of numbers sorted in an increasing order. Merge them to get a bigger sorted list.

**Basic Idea:** For each numbers in the second set, find it's correct place in the first set.

# The Merge Algorithm

**Problem:** Suppose we have two lists $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$ of numbers sorted in an increasing order. Merge them to get a bigger sorted list.

**Basic Idea:** For each numbers in the second set, find it's correct place in the first set.

**The Algorithm:**

- Scan the first set until the right place to insert $b_1$ is found.

# The Merge Algorithm

**Problem:** Suppose we have two lists $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$ of numbers sorted in an increasing order. Merge them to get a bigger sorted list.

**Basic Idea:** For each numbers in the second set, find it's correct place in the first set.

**The Algorithm:**

- Scan the first set until the right place to insert $b_1$ is found.
- Insert $b_1$.

# The Merge Algorithm

**Problem:** Suppose we have two lists $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$ of numbers sorted in an increasing order. Merge them to get a bigger sorted list.

**Basic Idea:** For each numbers in the second set, find it's correct place in the first set.

**The Algorithm:**

- Scan the first set until the right place to insert $b_1$ is found.
- Insert $b_1$.
- Continue the scan from that place until the right place to insert $b_2$ is found.

# The Merge Algorithm

**Problem:** Suppose we have two lists $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$ of numbers sorted in an increasing order. Merge them to get a bigger sorted list.

**Basic Idea:** For each numbers in the second set, find it's correct place in the first set.

**The Algorithm:**

- Scan the first set until the right place to insert $b_1$ is found.
- Insert $b_1$.
- Continue the scan from that place until the right place to insert $b_2$ is found.
- Repeat this for all elements of $B$.

# The Merge Algorithm (Cont.)

**Note:**

- Since the $b$'s are sorted, we never have to go back.

## The Merge Algorithm (Cont.)

**Note:**

- Since the $b$'s are sorted, we never have to go back.
- The total number of comparisons, in the worst case, is $m + n$.

# The Merge Algorithm (Cont.)

**Note:**

- Since the $b$'s are sorted, we never have to go back.
- The total number of comparisons, in the worst case, is $m + n$.

**Question:** What about data movements?

# The Merge Algorithm (Cont.)

**Note:**

- Since the $b$'s are sorted, we never have to go back.
- The total number of comparisons, in the worst case, is $m + n$.

**Question:** What about data movements?

- It is inefficient to move elements each time an insertion is performed.

# The Merge Algorithm (Cont.)

**Note:**

- Since the $b$'s are sorted, we never have to go back.
- The total number of comparisons, in the worst case, is $m + n$.

**Question:** What about data movements?

- It is inefficient to move elements each time an insertion is performed.
- Since the merge produces the elements one by one in sorted order, we copy them to a temporary array.

## The Merge Algorithm (Cont.)

**Note:**

- Since the $b$'s are sorted, we never have to go back.
- The total number of comparisons, in the worst case, is $m + n$.

**Question:** What about data movements?

- It is inefficient to move elements each time an insertion is performed.
- Since the merge produces the elements one by one in sorted order, we copy them to a temporary array.
- Each element is copied exactly once.

# The Merge Algorithm (Cont.)

**Note:**

- Since the $b$'s are sorted, we never have to go back.
- The total number of comparisons, in the worst case, is $m + n$.

**Question:** What about data movements?

- It is inefficient to move elements each time an insertion is performed.
- Since the merge produces the elements one by one in sorted order, we copy them to a temporary array.
- Each element is copied exactly once.

**Complexity:**

- **Time:** $\mathcal{O}(n + m)$ comparisons.
- **Space:** $\mathcal{O}(n + m)$ data.

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | 1 | 2 | 5 | 6 | 8 | 9 | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | 3 | 4 | 7 | 11 | 13 | 14 | 15 | 16 |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | 1 | 2 | 5 | 6 | 8 | 9 | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | 3 | 4 | 7 | 11 | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | | | | | | | | | | | | | | | | |

# The Merge Algorithm (Cont.)

$B$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 5 | 6 | 8 | 9 | 10 | 12 |

$A$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 4 | 7 | 11 | 13 | 14 | 15 | 16 |

$M$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | ~~1~~ | 2 | 5 | 6 | 8 | 9 | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | 3 | 4 | 7 | 11 | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | 1 | | | | | | | | | | | | | | | |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | $\cancel{1}$ | $\cancel{2}$ | 5 | 6 | 8 | 9 | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | 3 | 4 | 7 | 11 | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | 1 | 2 | | | | | | | | | | | | | | |

# The Merge Algorithm (Cont.)

$B$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~1~~ | ~~2~~ | 5 | 6 | 8 | 9 | 10 | 12 |

$A$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~3~~ | 4 | 7 | 11 | 13 | 14 | 15 | 16 |

$M$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | | | | | | | | | | | | | |

# The Merge Algorithm (Cont.)

$B$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\cancel{1}$ | $\cancel{2}$ | 5 | 6 | 8 | 9 | 10 | 12 |

$A$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\cancel{3}$ | $\cancel{4}$ | 7 | 11 | 13 | 14 | 15 | 16 |

$M$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | | | | | | | | | | | | |

# The Merge Algorithm (Cont.)

| $B$ : | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ~~1~~ | ~~2~~ | ~~5~~ | 6 | 8 | 9 | 10 | 12 |

| $A$ : | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ~~3~~ | ~~4~~ | 7 | 11 | 13 | 14 | 15 | 16 |

| $M$ : | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | | | | | | | | | | | |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | 8 | 9 | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | ~~3~~ | ~~4~~ | 7 | 11 | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | 1 | 2 | 3 | 4 | 5 | 6 | | | | | | | | | | |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | 8 | 9 | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | ~~3~~ | ~~4~~ | ~~7~~ | 11 | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | | | | | | | | |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | 9 | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | ~~3~~ | ~~4~~ | ~~7~~ | 11 | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | | | | | | | |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | ~~9~~ | 10 | 12 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | ~~3~~ | ~~4~~ | ~~7~~ | 11 | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | | | | | | |

# The Merge Algorithm (Cont.)

$B$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | ~~9~~ | ~~10~~ | 12 |

$A$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~3~~ | ~~4~~ | ~~7~~ | 11 | 13 | 14 | 15 | 16 |

$M$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | | | | | |

# The Merge Algorithm (Cont.)

B :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | ~~9~~ | ~~10~~ | 12 |

A :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~3~~ | ~~4~~ | ~~7~~ | ~~11~~ | 13 | 14 | 15 | 16 |

M :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | | | | | |

# The Merge Algorithm (Cont.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $B$ : | ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | ~~9~~ | ~~10~~ | ~~12~~ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ : | ~~3~~ | ~~4~~ | ~~7~~ | ~~11~~ | 13 | 14 | 15 | 16 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | | | |

# The Merge Algorithm (Cont.)

$B$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | ~~9~~ | ~~10~~ | ~~12~~ |

$A$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~3~~ | ~~4~~ | ~~7~~ | ~~11~~ | ~~13~~ | 14 | 15 | 16 |

$M$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | | | |

# The Merge Algorithm (Cont.)

| $B$ : | ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | ~~9~~ | ~~10~~ | ~~12~~ |
|---|---|---|---|---|---|---|---|---|

| $A$ : | ~~3~~ | ~~4~~ | ~~7~~ | ~~11~~ | ~~13~~ | ~~14~~ | 15 | 16 |
|---|---|---|---|---|---|---|---|---|

| $M$ : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

# The Merge Algorithm (Cont.)

$B$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~1~~ | ~~2~~ | ~~5~~ | ~~6~~ | ~~8~~ | ~~9~~ | ~~10~~ | ~~12~~ |

$A$ :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ~~3~~ | ~~4~~ | ~~7~~ | ~~11~~ | ~~13~~ | ~~14~~ | ~~15~~ | 16 |

$M$ :

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |

# The Merge Algorithm (Cont.)

| $B$ : | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\cancel{1}$ | $\cancel{2}$ | $\cancel{5}$ | $\cancel{6}$ | $\cancel{8}$ | $\cancel{9}$ | $\cancel{10}$ | $\cancel{12}$ |

| $A$ : | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $\cancel{3}$ | $\cancel{4}$ | $\cancel{7}$ | $\cancel{11}$ | $\cancel{13}$ | $\cancel{14}$ | $\cancel{15}$ | $\cancel{16}$ |

| $M$ : | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

The merge procedure is used as a basis for the divide-and-conquer sorting algorithm, known as the **mergesort**.

# Mergesort

The merge procedure is used as a basis for the divide-and-conquer sorting algorithm, known as the **mergesort**.

**The Algorithm:**

- Divide the sequence into two equal or close-to-equal parts.
- Sort each part separately using *recursion*.
- Merge the two sorted parts into one sorted sequence, using the *merge algorithm*.

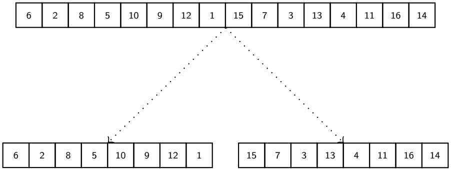
# Mergesort: An Example

**Splitting Phase:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 2 | 8 | 5 | 10 | 9 | 12 | 1 | 15 | 7 | 3 | 13 | 4 | 11 | 16 | 14 |

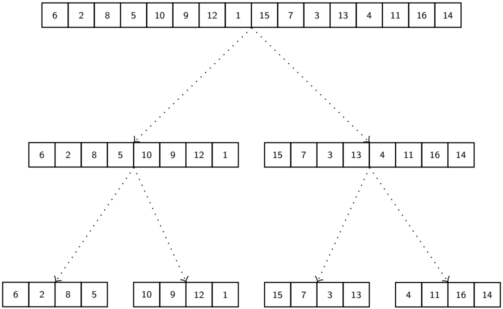
# Mergesort: An Example

**Splitting Phase:**



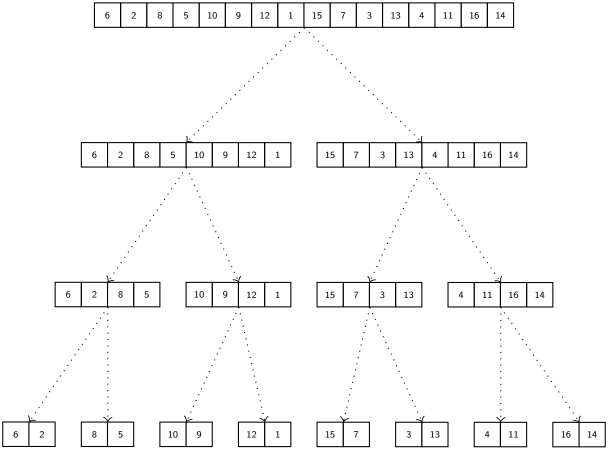
# Mergesort: An Example

**Splitting Phase:**



# Mergesort: An Example

**Splitting Phase:**



# Mergesort: An Example

**Merging Phase:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 2 | 8 | 5 | 10 | 9 | 12 | 1 | 15 | 7 | 3 | 13 | 4 | 11 | 16 | 14 |

# Mergesort: An Example

**Merging Phase:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 2 | 8 | 5 | 10 | 9 | 12 | 1 | 15 | 7 | 3 | 13 | 4 | 11 | 16 | 14 |

**Merging Phase:**

swap

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 8 | 5 | 10 | 9 | 12 | 1 | 15 | 7 | 3 | 13 | 4 | 11 | 16 | 14 |

# Mergesort: An Example

**Merging Phase:**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 8 | 5 | 10 | 9 | 12 | 1 | 15 | 7 | 3 | 13 | 4 | 11 | 16 | 14 |

# Mergesort: An Example

**Merging Phase:**

swap

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 5 | 8 | 10 | 9 | 12 | 1 | 15 | 7 | 3 | 13 | 4 | 11 | 16 | 14 |

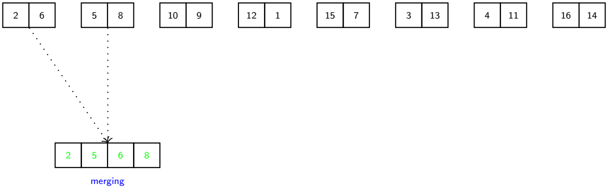
# Mergesort: An Example

**Merging Phase:**

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | | 5 | 8 | | 10 | 9 | | 12 | 1 | | 15 | 7 | | 3 | 13 | | 4 | 11 | | 16 | 14 |

# Mergesort: An Example

## Merging Phase:



# Mergesort: An Example

**Merging Phase:**

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 9 | | 12 | 1 | | 15 | 7 | | 3 | 13 | | 4 | 11 | | 16 | 14 |

| | | | |
|---|---|---|---|
| 2 | 5 | 6 | 8 |

# Mergesort: An Example

## Merging Phase:

swap

[9, 10] [12, 1] [15, 7] [3, 13] [4, 11] [16, 14]

[2, 5, 6, 8]

# Mergesort: An Example

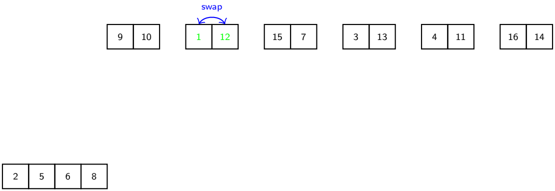
## Merging Phase:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9 | 10 | 12 | 1 | 15 | 7 | 3 | 13 | 4 | 11 | 16 | 14 |

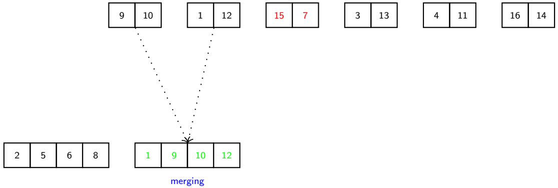
| | | | |
|---|---|---|---|
| 2 | 5 | 6 | 8 |

# Mergesort: An Example

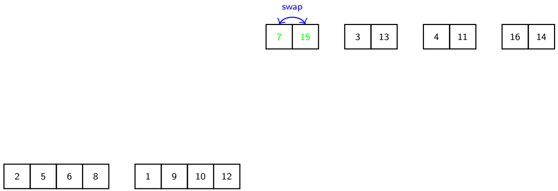
**Merging Phase:**



**Merging Phase:**



**Merging Phase:**



# Mergesort: An Example

**Merging Phase:**

| | |
|---|---|
| 15 | 7 |

| | |
|---|---|
| 3 | 13 |

| | |
|---|---|
| 4 | 11 |

| | |
|---|---|
| 16 | 14 |

| | | | |
|---|---|---|---|
| 2 | 5 | 6 | 8 |

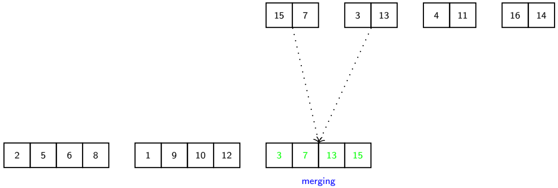
| | | | |
|---|---|---|---|
| 1 | 9 | 10 | 12 |

**Merging Phase:**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 7 | | 3 | 13 | | 4 | 11 | | 16 | 14 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 6 | 8 | | 1 | 9 | 10 | 12 |

**Merging Phase:**



# Mergesort: An Example

**Merging Phase:**

| | |
|---|---|
| 4 | 11 |

| | |
|---|---|
| 16 | 14 |

| | | | |
|---|---|---|---|
| 2 | 5 | 6 | 8 |

| | | | |
|---|---|---|---|
| 1 | 9 | 10 | 12 |

| | | | |
|---|---|---|---|
| 3 | 7 | 13 | 15 |

**Merging Phase:**

| | | | | |
|---|---|---|---|---|
| 4 | 11 | | 16 | 14 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 6 | 8 | | 1 | 9 | 10 | 12 | | 3 | 7 | 13 | 15 |

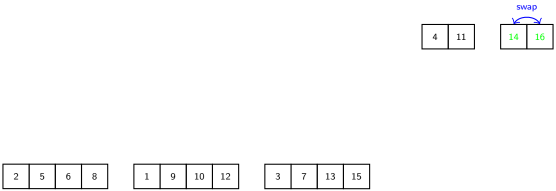
# Mergesort: An Example

**Merging Phase:**

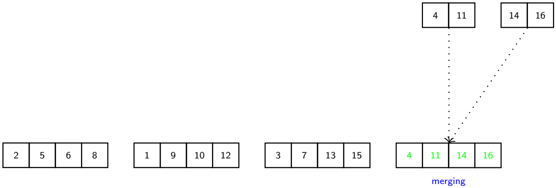
| | | | | | |
|---|---|---|---|---|---|
| 4 | 11 | | | 16 | 14 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 6 | 8 | | 1 | 9 | 10 | 12 | | 3 | 7 | 13 | 15 |

**Merging Phase:**

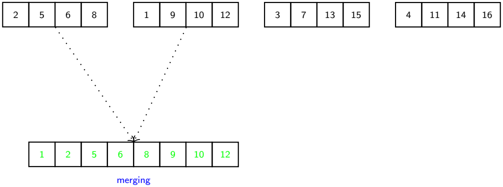


**Merging Phase:**



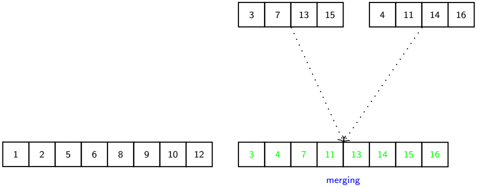
# Mergesort: An Example

## Merging Phase:



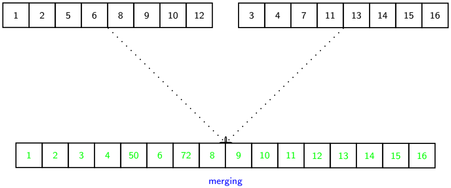
# Mergesort: An Example

**Merging Phase:**



# Mergesort: An Example

## Merging Phase:



# Recurrences: Divide and Conquer

# Divide and Conquer Relations: The Basic Idea

- The original problem is divided into smaller subproblems.

# Divide and Conquer Relations: The Basic Idea

- The original problem is divided into smaller subproblems.
- Each subproblem is solved recursively.

# Divide and Conquer Relations: The Basic Idea

- The original problem is divided into smaller subproblems.
- Each subproblem is solved recursively.
- A *combine* algorithm is used to solve the original problem.

## Divide and Conquer Relations: Problem Statement

**Assumptions:**

- # Subproblems: $a$.
- Size of Each Subproblem: $1/b$ of the original problem.
- Combine Algorithm: Takes time $cn^k$.

where $a$, $b$, $c$, and $k$ are some constant.

# Divide and Conquer Relations: Problem Statement

**Assumptions:**

- # Subproblems: $a$.
- Size of Each Subproblem: $1/b$ of the original problem.
- Combine Algorithm: Takes time $cn^k$.

where $a, b, c$, and $k$ are some constant.

Then,

$$T(n) = aT(n/b) + cn^k.$$

# Divide and Conquer Relations: Problem Statement

**Assumptions:**

- \# Subproblems: $a$.
- Size of Each Subproblem: $1/b$ of the original problem.
- Combine Algorithm: Takes time $cn^k$.

where $a, b, c$, and $k$ are some constant.

Then,

$$T(n) = aT(n/b) + cn^k.$$

**For Simplicity:** Further assume that $n = b^m$, so that $n/b$ is always an integer ($b$ is an integer greater than 1).

# Divide and Conquer Relations (Cont.)

**Expand:**

$$T(n) = a\{aT(n/b^2) + c(n/b)^k\} + c(n)^k$$

## Divide and Conquer Relations (Cont.)

**Expand:**

$$
\begin{aligned}
T(n) &= a\{aT(n/b^2) + c(n/b)^k\} + c(n)^k \\
&= a\{a\{aT(n/b^3) + c(n/b^2)^k\} + c(n/b)^k\} + cn^k
\end{aligned}
$$

**Expand:**

$$
\begin{aligned}
T(n) &= a\{aT(n/b^2) + c(n/b)^k\} + c(n)^k \\
&= a\{a\{aT(n/b^3) + c(n/b^2)^k\} + c(n/b)^k\} + cn^k \\
&\vdots \\
&= a\{a\{a\{\cdots a\{T(n/b^m) + c(n/b^{m-1})^k\} + \cdots\} + cn^k,
\end{aligned}
$$

where $n/b^m = 1$.

# Divide and Conquer Relations (Cont.)

**Expand:**

$$\begin{aligned}
T(n) &= a\{aT(n/b^2) + c(n/b)^k\} + c(n)^k \\
&= a\{a\{aT(n/b^3) + c(n/b^2)^k\} + c(n/b)^k\} + cn^k \\
&\vdots \\
&= a\{a\{a\{\cdots a\{T(n/b^m) + c(n/b^{m-1})^k\} + \cdots\} + cn^k,
\end{aligned}$$

where $n/b^m = 1$.

**Assume:** $T(1) = c$.

# Divide and Conquer Relations (Cont.)

**Expand:**

$$
\begin{aligned}
T(n) &= a\{aT(n/b^2) + c(n/b)^k\} + c(n)^k \\
&= a\{a\{aT(n/b^3) + c(n/b^2)^k\} + c(n/b)^k\} + cn^k \\
&\vdots \\
&= a\{a\{a\{\cdots a\{T(n/b^m) + c(n/b^{m-1})^k\} + \cdots\} + cn^k,
\end{aligned}
$$

where $n/b^m = 1$.

**Assume:** $T(1) = c$.

**Remark:** A different value would change the end result by only a constant.

## Divide and Conquer Relations (Cont.)

$$\therefore \; T(n) = ca^m + ca^{m-1}b^k + ca^{m-2}b^{2k} + \cdots + cb^{mk}$$

# Divide and Conquer Relations (Cont.)

$$\begin{aligned} \therefore\ T(n) &= ca^m + ca^{m-1}b^k + ca^{m-2}b^{2k} + \cdots + cb^{mk} \\ &= c\sum_{i=0}^{m} a^{m-i}b^{ik} = ca^m \sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i, \end{aligned}$$

# Divide and Conquer Relations (Cont.)

$$\begin{aligned}
\therefore\ T(n) &= ca^m + ca^{m-1}b^k + ca^{m-2}b^{2k} + \cdots + cb^{mk} \\
&= c\sum_{i=0}^{m} a^{m-i}b^{ik} = ca^m \sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i,
\end{aligned}$$

which is a simple geometric series.

## Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$:
  - The factor of the geometric series is less than 1.

## Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$:
  - The factor of the geometric series is less than 1.
  - So the series converges to a constant as $m \to \infty$.

# Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$:
  - The factor of the geometric series is less than 1.
  - So the series converges to a constant as $m \to \infty$.
  - Therefore,

$$T(n) = \mathcal{O}(a^m) = \mathcal{O}(a^{\log_b n}) = \mathcal{O}(n^{\log_b a}),$$

  as $m = \log_b n$.

## Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$:
- $a = b^k$:
  - The factor of the geometric series is equal to 1.

# Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$**:**
- $a = b^k$**:**
    - The factor of the geometric series is equal to 1.
    - Thus

$$T(n) = \mathcal{O}(a^m m) = \mathcal{O}(n^k \log n),$$

since, $a = b^k \;\Rightarrow\; \log_b a = k$ and $m = \log_b n$.

## Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$**:**
- $a = b^k$**:**
- $a < b^k$**:**
  - The factor of the geometric series is greater than 1.

# Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$**:**
- $a = b^k$**:**
- $a < b^k$**:**
  - The factor of the geometric series is greater than 1.
  - Let $F = b^k/a$ ($F$ is a constant).

# Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$:
- $a = b^k$:
- $a < b^k$:
  - The factor of the geometric series is greater than 1.
  - Let $F = b^k/a$ ($F$ is a constant).
  - First element of the series is $a^m$, therefore we obtain

$$T(n) = \frac{a^m(F^{m+1} - 1)}{F - 1}$$

# Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$:
- $a = b^k$:
- $a < b^k$:
  - The factor of the geometric series is greater than 1.
  - Let $F = b^k/a$ ($F$ is a constant).
  - First element of the series is $a^m$, therefore we obtain

$$\begin{aligned} T(n) &= \frac{a^m(F^{m+1}-1)}{F-1} \\ &= \mathcal{O}(a^m F^m) = \mathcal{O}((b^k)^m) = \mathcal{O}((b^m)^k) \end{aligned}$$

# Divide and Conquer Relations (Cont.)

The following cases may arise:

- $a > b^k$:
- $a = b^k$:
- $a < b^k$:
  - The factor of the geometric series is greater than 1.
  - Let $F = b^k/a$ ($F$ is a constant).
  - First element of the series is $a^m$, therefore we obtain

$$\begin{aligned} T(n) &= \frac{a^m(F^{m+1}-1)}{F-1} \\ &= \mathcal{O}(a^m F^m) = \mathcal{O}((b^k)^m) = \mathcal{O}((b^m)^k) \\ &= \mathcal{O}(n^k). \end{aligned}$$

# Master's Theorem: A Simpler Version

**Theorem**

*The solution of the recurrence relation* $T(n) = aT(n/b) + cn^k$, *where* $a$ *and* $b$ *are integer constants,* $a \geq 1, b \geq 2$, *and* $c$ *and* $k$ *are positive constants, is*

$$T(n) = \begin{cases} \mathcal{O}(n^{\log_b a}) & \textit{if } a > b^k \\ \mathcal{O}(n^k \log n) & \textit{if } a = b^k \\ \mathcal{O}(n^k) & \textit{if } a < b^k \end{cases}$$

Merge Sort: Cost Analysis

# Cost Analysis

$$T(n) = 2T(\lceil n/2 \rceil) + \mathcal{O}(n) = \mathcal{O}(n \log n) \quad \text{[By Master's theorem].}$$

## Cost Analysis

$$T(n) = 2T(\lceil n/2 \rceil) + \mathcal{O}(n) = \mathcal{O}(n \log n) \quad \text{[By Master's theorem].}$$

**Note:** The number of data movements is $\mathcal{O}(n \log n)$!!

# Cost Analysis

$$T(n) = 2T(\lceil n/2 \rceil) + \mathcal{O}(n) = \mathcal{O}(n \log n) \quad \text{[By Master's theorem]}.$$

**Note:** The number of data movements is $\mathcal{O}(n \log n)$!!

**Drawbacks:**

- Not as easy to implement.
- Additional storage required during each merge step.
- Thus, mergesort is not an in-place algorithm.
- This copying must be done every time two smaller sets are merged, making the procedure slower.

## Cost Analysis

$$T(n) = 2T(\lceil n/2 \rceil) + \mathcal{O}(n) = \mathcal{O}(n \log n) \quad \text{[By Master's theorem]}.$$

**Note:** The number of data movements is $\mathcal{O}(n \log n)$!!

**Drawbacks:**

- Not as easy to implement.
- Additional storage required during each merge step.
- Thus, mergesort is not an in-place algorithm.
- This copying must be done every time two smaller sets are merged, making the procedure slower.

**Home Work:** Write the algorithm for Mergesort and implement it in C.

# Books Consulted

1. *Introduction to Algorithms: A Creative Approach* by Udi Manber.

2. *Introduction to Algorithms* by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.

Thank You for your kind attention!