

# Merge Sort

Joy Mukherjee

# Sorting

- Given an array  $A[]$  of  $n$  integers, our objective is to arrange them in non-decreasing order/non-increasing order.
- Bubble Sort/Selection Sort/Insertion Sort
- Merge Sort/Quick Sort/Heap Sort
- Counting Sort/Radix Sort
- Bucket Sort

# Merge Sort

- Given two sorted arrays of length  $m$  and  $n$ , how do we merge them into a single sorted array of size  $m+n$ ?
- We use an extra array of size  $m+n$  to merge the two sorted arrays.

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500
	i					

	1	2	3	4	5
B	-9	-1	30	60	101
	j				

[illegible]

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500
	i					

	1	2	3	4	5
B	-9	-1	30	60	101

j

[illegible]

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500
	i					

	1	2	3	4	5
B	-9	-1	30	60	101
			j		

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	0	0	0	0	0	0	0	0	0
			k								

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500
	i					

	1	2	3	4	5
B	-9	-1	30	60	101

j

[illegible]

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500

i

	1	2	3	4	5
B	-9	-1	30	60	101

j

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	0	0	0	0	0	0	0

k



# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500

i

	1	2	3	4	5
B	-9	-1	30	60	101

j

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	30	0	0	0	0	0	0

k

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500

i

	1	2	3	4	5
B	-9	-1	30	60	101

j

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	30	35	0	0	0	0	0

k

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500
				i		

	1	2	3	4	5
B	-9	-1	30	60	101
				j	

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	30	35	60	0	0	0	0
							k				

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500

i

	1	2	3	4	5
B	-9	-1	30	60	101

j

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	30	35	60	92	0	0	0

k

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500

i

	1	2	3	4	5
B	-9	-1	30	60	101

j

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	30	35	60	92	101	0	0

k

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500

i

	1	2	3	4	5
B	-9	-1	30	60	101

j

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	30	35	60	92	101	101	0

k

# How to merge two sorted array?

	1	2	3	4	5	6
A	5	26	35	92	101	500

i

	1	2	3	4	5
B	-9	-1	30	60	101

j

	1	2	3	4	5	6	7	8	9	10	11
C	-9	-1	5	26	30	35	60	92	101	101	500

k

# Merge

- A [low..high]
- A[low..mid] and A[mid+1..high]

	1	2	3	4	5	6	7	8	9	10	11
A	9	11	15	26	39	45	60	9	10	20	29
								i			j

	1	2	3	4	5	6	7	8	9	10	11
B	9	9	10	11	15	20	26	29	39	45	60



# Merge

```
void merge(int a[], int low, int mid,  
           int high)
```

```
{
```

```
    int i = low, j = mid+1, k = 0;
```

```
    int b[high-low+1];
```

```
    while(i <= mid && j <= high) { // Both  
        array are not exhausted
```

```
        if(a[i] <= a[j])
```

```
            {b[k] = a[i]; i++; k++;}
```

```
        else
```

```
            {b[k] = a[j]; j++; k++;}
```

```
    }
```

a[low] to a[mid] = A

a[mid+1] to a[high] = B

b[0] to b[high-low] = C

# Merge

```
while(j != high+1) // a[mid+1] to a[high] is not exhausted
```

```
    b[k++] = a[j++];
```

```
while(i != mid+1) // a[low] to a[mid] is not exhausted
```

```
    b[k++] = a[i++];
```

```
for(i = low, j = 0; i <= high; i++, j++)
```

```
    a[i] = b[j]; // Copy sorted b[0 .. High-low] to a[low to high]
```

```
}
```

# Merge Sort

- It is a recursive function.
  - Divide the array into almost equal two halves
  - Recursively sort left half **A[low..mid]**
  - Recursively sort right half **A[mid+1..high]**
  - Merge the two halves to get the array sorted **A[low..high]**

# Merge Sort

```
void mergesort(int a[], int low, int high)
{
    if(low < high) {
        int mid = (high+low)/2;
        mergesort(a, low, mid); // Sort the left half
        mergesort(a, mid+1, high); // Sort the right half
        merge(a, low, mid, high); // Merge the sorted halves
    }
}
```

From main(): mergesort(a, 0, n-1);

# Merge Sort

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

# Merge Sort : Divide (Level 1)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

# Merge Sort : Divide (Level 2)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2
A	92	35

# Merge Sort : Divide (Level 3) & Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2
A	92	35

	1
A	92



# Merge Sort : Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2
A	92	35

	1	2
A	92	35

# Merge Sort : Conquer (Level 3)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2
A	35	92

	1	2
A	92	35

# Merge Sort : Divide (Level 2)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2	3	4
A	35	92	26	101

	1	2
A	92	35

# Merge Sort : Divide (Level 3) & Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2	3	4
A	35	92	26	101

	1	2	3
A	92	35	26

# Merge Sort : Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2	3	4
A	35	92	26	101

	1	2	3	4
A	92	35	26	101

# Merge Sort : Conquer (Level 3)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	92	35	26	101

	1	2	3	4
A	35	92	26	101

	1	2	3	4
A	92	35	26	101

# Merge Sort : Conquer (Level 2)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4
A	26	35	92	101

	1	2	3	4
A	35	92	26	101

	1	2	3	4
A	92	35	26	101

# Merge Sort : Divide (Level 1)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4
A	35	92		26	101

	1	2		3	4
A	92	35		26	101



# Merge Sort : Divide (Level 2)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6
A	35	92		26	101		5	500

	1	2		3	4
A	92	35		26	101

# Merge Sort : Divide (Level 3) & Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6
A	35	92		26	101		5	500

	1	2		3	4		5
A	92	35		26	101		5

# Merge Sort : Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6
A	35	92		26	101		5	500

	1	2		3	4		5	6
A	92	35		26	101		5	500

# Merge Sort : Conquer (Level 3)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6
A	35	92		26	101		5	500

	1	2		3	4		5	6
A	92	35		26	101		5	500

# Merge Sort : Divide (Level 2)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6		7	8
A	35	92		26	101		5	500		101	92

	1	2		3	4		5	6
A	92	35		26	101		5	500

# Merge Sort : Divide (Level 3) & Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6		7	8
A	35	92		26	101		5	500		101	92

	1	2		3	4		5	6		7
A	92	35		26	101		5	500		101

# Merge Sort : Conquer (Level 4)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6		7	8
A	35	92		26	101		5	500		101	92

	1	2		3	4		5	6		7	8
A	92	35		26	101		5	500		101	92

# Merge Sort : Conquer (Level 3)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	500	101	92

	1	2		3	4		5	6		7	8
A	35	92		26	101		5	500		92	101

	1	2		3	4		5	6		7	8
A	92	35		26	101		5	500		101	92



# Merge Sort : Conquer (Level 2)

	1	2	3	4	5	6	7	8
A	92	35	26	101	5	500	101	92

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	92	101	500

	1	2		3	4		5	6		7	8
A	35	92		26	101		5	500		92	101

	1	2		3	4		5	6		7	8
A	92	35		26	101		5	500		101	92

# Merge Sort : Conquer (Level 1)

	1	2	3	4	5	6	7	8
A	5	26	35	92	92	101	101	500

	1	2	3	4		5	6	7	8
A	26	35	92	101		5	92	101	500

	1	2		3	4		5	6		7	8
A	35	92		26	101		5	500		92	101

	1	2		3	4		5	6		7	8
A	92	35		26	101		5	500		101	92

# Merge Sort

```
void mergesort(int a[], int low, int high)
{
    if(low < high) {
        int mid = (high+low)/2;
        mergesort(a, low, mid); // Sort the left half
        mergesort(a, mid+1, high); // Sort the right half
        merge(a, low, mid, high); // Merge the sorted halves
    }
}
```

# Time Complexity: Merge Sort

- $T(n)$  = Time to merge sort array of size  $n = 2^k$  integers
- $T(1) = 1$

$$T(n) = 2T(n/2) + cn$$

$$2T(n/2) = 2^2T(n/2^2) + cn$$

$$2^2T(n/2^2) = 2^3T(n/2^3) + cn$$

.....

$$2^{k-1}T(n/2^{k-1}) = 2^kT(n/2^k) + cn$$

-----

$$T(n) = 2^kT(n/2^k) + cn. \quad k \quad (\text{Summation of } k \text{ equations})$$

$$T(n) = nT(1) + c.n.\log_2 n = n + c.n.\log_2 n = O(n\log_2 n)$$

# Space Complexity: Merge Sort

- Space needed to solve merge sort
- Do not consider the space for input array
- Consider the extra space needed in terms of input size
- At the bottom level, an array of size 2 is required
- Immediately higher level, an array of size 4 is required
- In the next higher level, an array of size 8 is required
- ...
- In the top level, an array of size  $2^k = n$  is required
- **Important to note:** when array in one level is used/created, the arrays in the bottom levels have already been destroyed.
- Therefore, the space complexity =  $O(\max(2, 4, 8, \dots, 2^k = n)) = O(n)$