


# CS Bridge Module 14 Sorting

## 1. Sorting Problem Definition

### 1.1 CS Bridge: Sorting




NYU TANDON  
ONLINE

CS Bridge: Sorting

Module 14  
Itay Tal

### 1.2 The Sorting Problem



NYU TANDON  
ONLINE


#### The Sorting Problem

Problem

Given an array `arr` of numbers, reorder them, so that at the end, they are in an increasing order.

Example

If `arr` is an array containing: [5, 8, 12, 7, 8, 10]  
After sorting, `arr` will look like: [5, 7, 8, 8, 10, 12]




## 1.3 Sorting Algorithms

Sorting Algorithms

NYU TANDON  
ONLINE

- Selection-Sort
- Insertion-Sort
- Bubble-Sort
- Merge-Sort
- Quick-Sort
- Heap-Sort
- ...



## 2. Selection Sort

### 2.1 Selection Sort

Selection Sort

NYU TANDON  
ONLINE

0	1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	3		5		10		14		8	
5	14	10	8	13	1	18	3	4	2	19	3	4

curr ↑ min ↑

## 2.2 Selection Sort Implementation

### Selection Sort Implementation



```
findIndexOfMin(
);

void selectionSort(int arr[], int arrSize){
    int i,
    for(i = 0; i < arrSize; i++){
        minInd =
        swap(arr[i], arr[minInd]);
    }
}
```

## 2.3 Selection Sort Implementation

### Selection Sort Implementation



```
int findIndexOfMin(int arr[], int low, int high){
    int min, minInd;
    int i;

    min = arr[low];
    minInd = low;
    for (
    if (arr[i] < min){
        min = arr[i];
        minInd = i;
    }
    return minInd;
}
```

## 2.4 Runtime Analysis

### Runtime Analysis



```
int findIndexOfMin(int arr[], int low, int high){
    int min, minInd;
    int i;

    min = arr[low];
    minInd = low;
    for (i = low+1; i <= high; i++){
        if (arr[i] < min){
            min = arr[i];
            minInd = i;
        }
    }

    return minInd;
}
```

$\theta(1)$   $\theta(n)$   $\theta(1)$   $\theta(1)$

Let:  $n = \text{high} - \text{low} + 1$

$T(n) =$

## 2.5 Runtime Analysis

Runtime Analysis

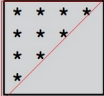
NYU TANDON  
ONLINE

```
void selectionSort(int arr[], int arrSize){  
    int i, minInd;  
    for(i = 0; i < arrSize; i++){  
        minInd = findIndexOfMin(arr, i, arrSize-1);  
        swap(arr[i], arr[minInd]);  
    }  
}
```

$\Theta(n-i)$

Let:  $n = \text{high} - \text{low} + 1$

$T(n) =$



$\downarrow$

$T(n) = \Theta(n^2)$

## 3. Merge Sort

### 3.1 Merge Sort

Merge Sort

NYU TANDON  
ONLINE

1	3	5	8	10	13	14	18
5	8	10	14	1	3	13	18
14	5	8	10	13	1	18	3

Step 1:

Step 2:

Step 3:

Notes:

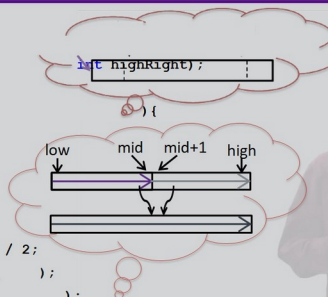
## 3.2 Implementation

Implementation

```
merge(int arr[],
      int highRight);

mergeSort(
int mid;

if (
    return;
else{
    mid = (low + high) / 2;
    mergeSort(
    mergeSort(
    merge(arr, low, mid, high);
}
}
```



## 3.3 Merge Sort

Merge Sort

min1

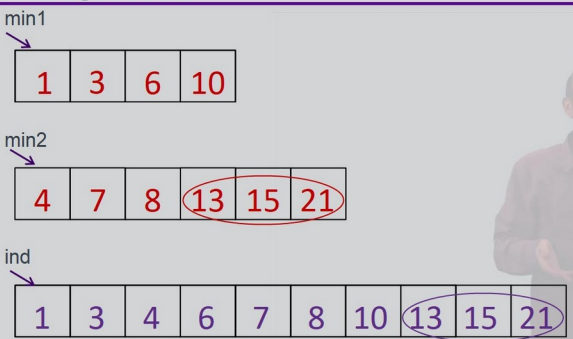
1	3	6	10
---	---	---	----

min2

4	7	8	13	15	21
---	---	---	----	----	----

ind

1	3	4	6	7	8	10	13	15	21
---	---	---	---	---	---	----	----	----	----



Notes:

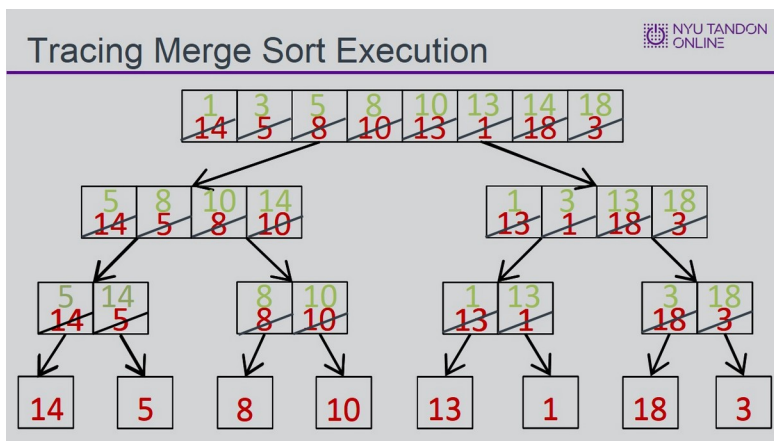
### 3.4 Merge Sort Implementation

```

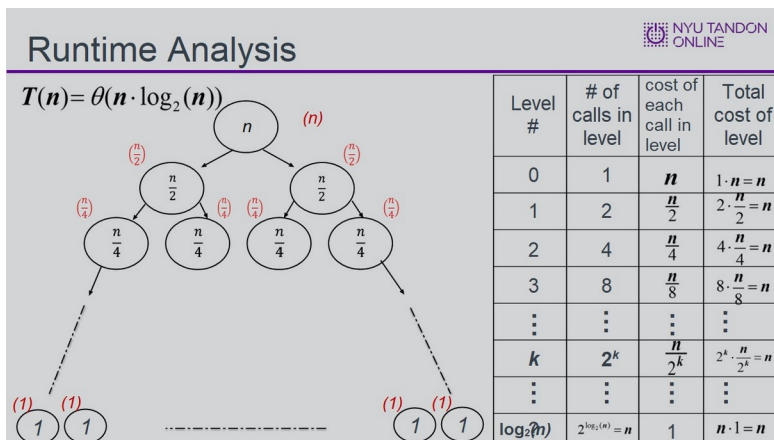
1 #include <iostream>
2 using namespace std;
3
4 void printArray(int arr[], int arrSize);
5 void mergeSort(int arr[], int low, int high);
6 void merge(int arr[], int lowLeft, int highLeft, int highRight);
7
8 int main() {
9     int arr[8] = {14, 5, 8, 10, 13, 1, 18, 3};
10    int arrSize = 8;
11
12    mergeSort(arr, 0, arrSize - 1);
13    printArray(arr, arrSize);
14
15    return 0;
16 }
17
18 void printArray(int arr[], int arrSize){
19     int i;
20
21     for (i = 0; i < arrSize; i++) {
22         cout<<arr[i]<<" ";
23     }
24     cout<<endl;
25 }
26
27

```

### 3.5 Tracing Merge Sort Execution




### 3.6 Runtime Analysis



### 3.7 Knowledge Check

(Sequence Drag-and-Drop, 10 points, 3 attempts permitted)

Knowledge Check



List the running times for Linear Search, Binary Search, Selection Sort, and Merge Sort in increasing order (from lowest to highest).

1. Binary Search (  $O(\log n)$  )

2. Linear Search (  $O(n)$  )

3. Merge Sort (  $O(n \log n)$  )

4. Selection Sort (  $O(n^2)$  )

Correct Order
Binary Search ( $O(\log n)$ )
Linear Search ( $O(n)$ )
Merge Sort ( $O(n \log n)$ )
Selection Sort ( $O(n^2)$ )

**Feedback when correct:**

That's right! You selected the correct response.

**Feedback when incorrect:**

You did not select the correct response.

### Correct (Slide Layer)

Knowledge Check

NYU TANDON  
ONLINE

List the running times for Linear Search, Binary Search, Selection Sort, and Merge Sort in increasing order (from lowest to highest).

Correct

That's right! You selected the correct response.

Continue

1. Binary Search
2. Linear Search
3. Merge Sort
4. Selection Sort (  $O(n^2)$  )

### Incorrect (Slide Layer)

Knowledge Check

NYU TANDON  
ONLINE

List the running times for Linear Search, Binary Search, Selection Sort, and Merge Sort in increasing order (from lowest to highest).

Incorrect

You did not select the correct response.

Continue

1. Binary Search
2. Linear Search
3. Merge Sort
4. Selection Sort (  $O(n^2)$  )

### Try Again (Slide Layer)

Knowledge Check

NYU TANDON  
ONLINE

List the running times for Linear Search, Binary Search, Selection Sort, and Merge Sort in increasing order (from lowest to highest).

Incorrect

That is incorrect. Please try again.

Try Again

1. Binary Search
2. Linear Search
3. Merge Sort
4. Selection Sort (  $O(n^2)$  )



### 3.8 Knowledge Check

(Matching Drag-and-Drop, 10 points, 2 attempts permitted)

Knowledge Check

NYU TANDON  
ONLINE

The following sorting algorithms have running times as follows:

Bubble Sort :  $O(n^2)$   
Heap Sort:  $O(n \log n)$

Bubble Sort

Selection Sort

Heap Sort

Merge Sort

Correct	Choice
Bubble Sort	Selection Sort
Heap Sort	Merge Sort

#### Feedback when correct:

That's right! You selected the correct response.

#### Feedback when incorrect:

You did not select the correct response.

### Correct (Slide Layer)

NYU TANDON  
ONLINE

## Knowledge Check

The following sorting algorithms have running times as follows:

Bubble Sort :  $O(n^2)$   
Heap Sort:  $O(n \log n)$

Bubble Sort

Heap Sort

Correct

That's right! You selected the correct response.

Continue

### Incorrect (Slide Layer)

NYU TANDON  
ONLINE

## Knowledge Check

The following sorting algorithms have running times as follows:

Bubble Sort :  $O(n^2)$   
Heap Sort:  $O(n \log n)$

Bubble Sort

Heap Sort

Incorrect

You did not select the correct response.

Continue

### Try Again (Slide Layer)

NYU TANDON  
ONLINE

## Knowledge Check

The following sorting algorithms have running times as follows:

Bubble Sort :  $O(n^2)$   
Heap Sort:  $O(n \log n)$

Bubble Sort


Heap Sort

Incorrect

That is incorrect. Please try again.

Try Again

### 3.9 End of Module



NYU TANDON  
ONLINE

End of Module

Exit