



TechM Full-Stack Software Development

Lecture On: Advanced Sorting Algorithms

Instructor: Arkoprovo Dey

In Last Class, we covered....

- Basic Sorting Algorithms

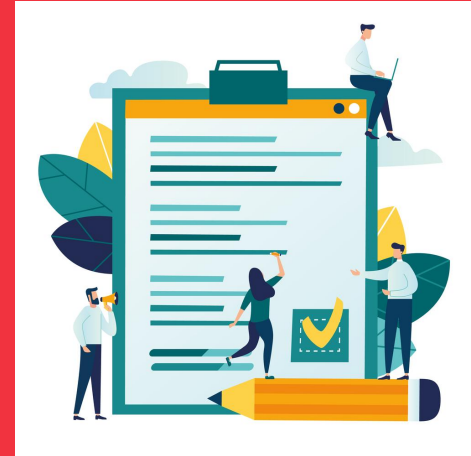


Homework Discussion

1. Write a program to sort an array in descending order using bubble sort.
2. Write a program to sort an array in descending order using selection sort.

Today's Agenda

1 Advanced Sorting Algorithms



Sorting

In the last session, we looked at some sorting algorithms that were not so efficient, apart from a few specific data sets. In this session, we will try to ensure a better worst-case performance.

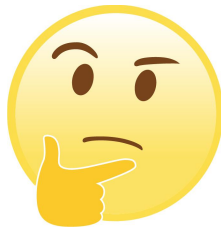
In order to do this, we will use a common design paradigm, the **'Divide and Conquer'** paradigm.

Let us first understand what we mean by 'divide and conquer' algorithms. As the term suggests, you try to **'divide'** big problems into smaller pieces that are more manageable and then **'conquer'** them.



You can draw an analogy: a team 'dividing' a big project into smaller tasks and then individuals or smaller teams 'conquering' them.





If you recall, we had used a recursive way to reduce the search space of the data while implementing Binary Search. Is Binary Search a 'divide and conquer' algorithm?

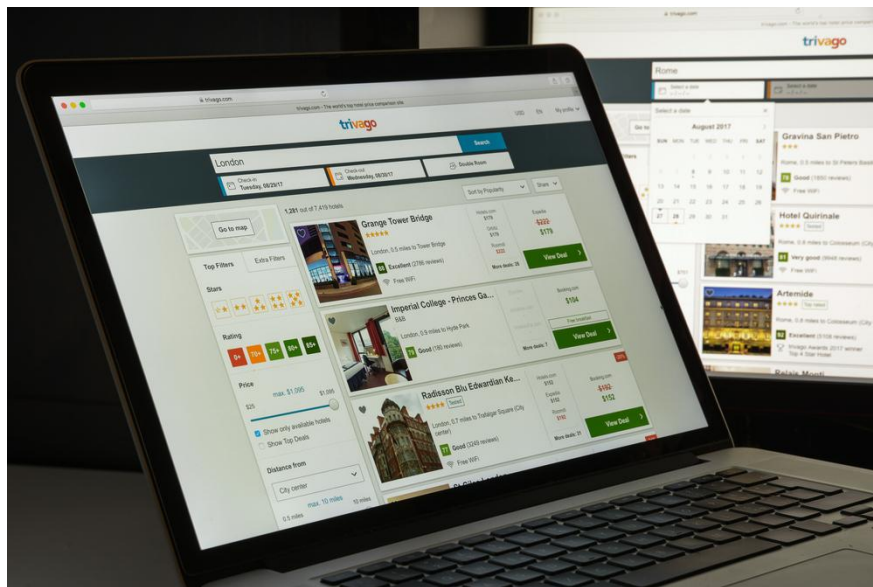
Yes

Rule of Thumb

Recursion will be used a lot in 'divide and conquer' algorithms because the algorithms will try to divide the problem multiple times into smaller chunks and then solve upon a smaller data set.

Advantages

- Break down complex problems
- Smaller problems can run in parallel, making better use of the system's capability
- Design more efficient algorithms



How do you think Trivago collects all the data from different websites and lists them down on their own website?

Poll 1 (15 Sec.)

Which of the following is correct about divide and conquer?

1. Every time, take the top half of the array and perform the operation.
2. Every time, split the array into two halves.
3. First split the whole set into two halves and then later merge them again.
4. None of these

Poll 1 (Answer)

Which of the following is correct about divide and conquer?

1. Every time, take the top half of the array and perform the operation.
2. Every time, split the array into two halves.
3. **First split the whole set into two halves and then later merge them again.**
4. None of these

Poll 2 (15 Sec.)

In which of the following can the divide and conquer algorithm be applied?

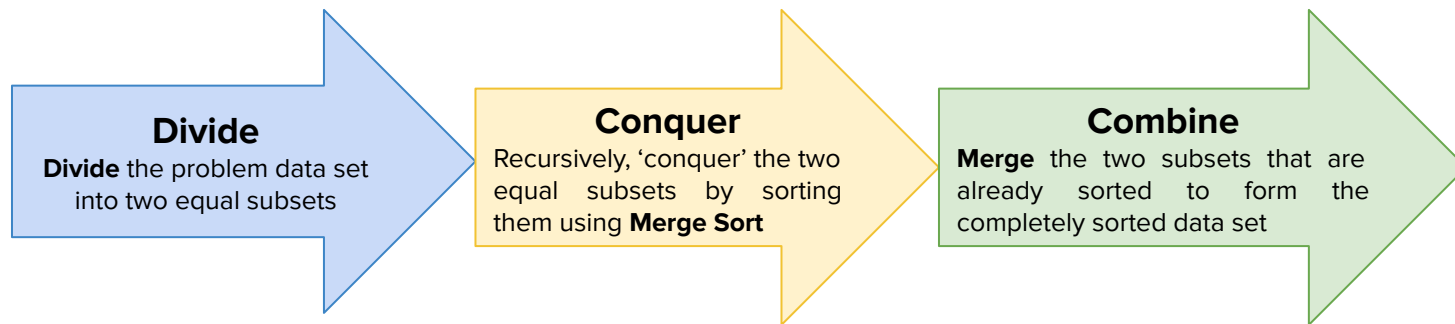
1. Linear search
2. Bubble sort
3. Finding an element in an unordered array
4. Finding an element in an ordered array

Poll 2 (Answer)

In which of the following can the divide and conquer algorithm be applied?

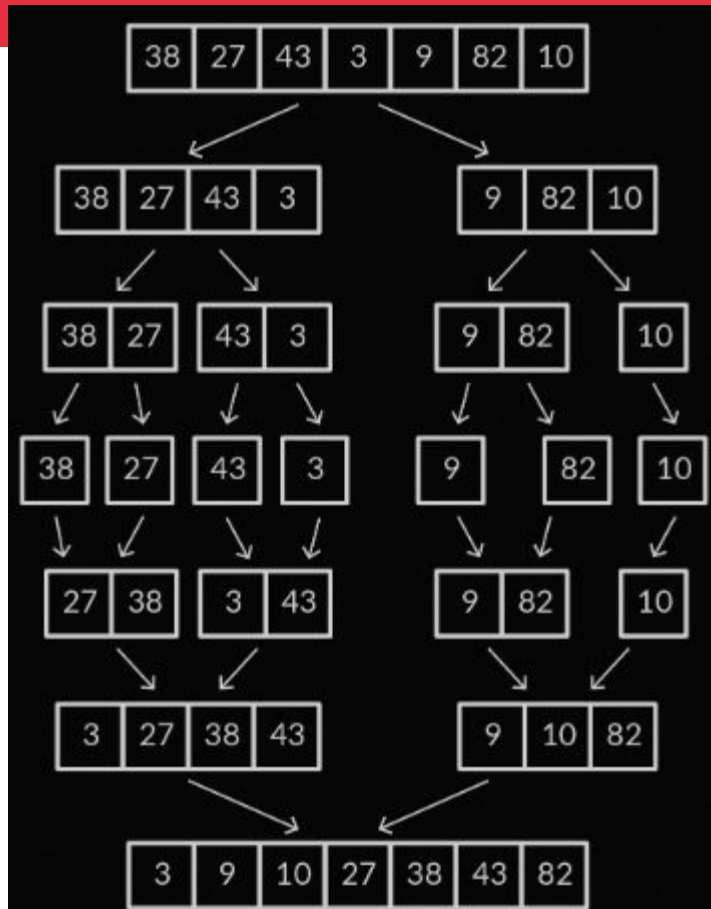
1. Linear search
2. Bubble sort
3. Finding an element in an unordered array
4. **Finding an element in an ordered array (Binary Search)**

Merge Sort technique works in three steps:



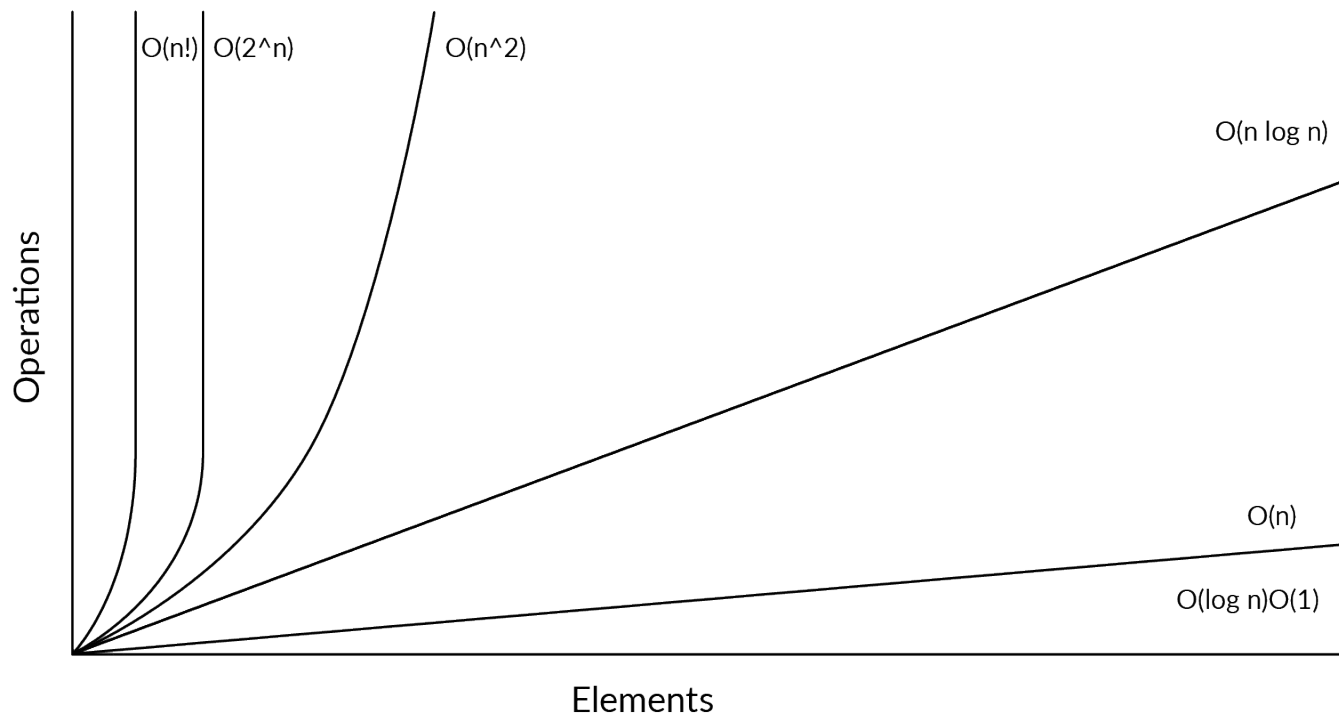
Merge Sort is based on a few basic ideas:

- It is easier to sort smaller lists than larger ones.
- It is easier to come up with a merged sorted list from two sorted arrays than from two unsorted ones.



6 5 3 1 8 7 2 4

Though it might not seem like it, the difference between $O(n^2)$ and $O(n \log n)$ is **HUGE**.



```
mergesort(A, l, r){  
  if (r>l) {  
    mid = (l+r)/2  
    mergesort(A, l, mid)  
    mergesort(A, mid+1, r)  
    merge(A, l, mid, r)  
  }  
}
```



Is Merge Sort a stable sorting algorithm?

Yes

Is Merge Sort an in-place sorting algorithm?

No

Hands-on Coding

- Implement merge sort in Java
- Merge a sorted array of size n into another sorted array of size $m+n$ (having m elements and these m elements are in the starting m positions of the second array).

Poll 3 (15 Sec.)

What is the space complexity of merge sort?

1. $O(n)$
2. $O(n^2)$
3. $O(n \log n)$
4. None of these

Poll 3 (Answer)

What is the space complexity of merge sort?

1. **$O(n)$**
2. $O(n^2)$
3. $O(n \log n)$
4. None of these

Poll 4 (15 Sec.)

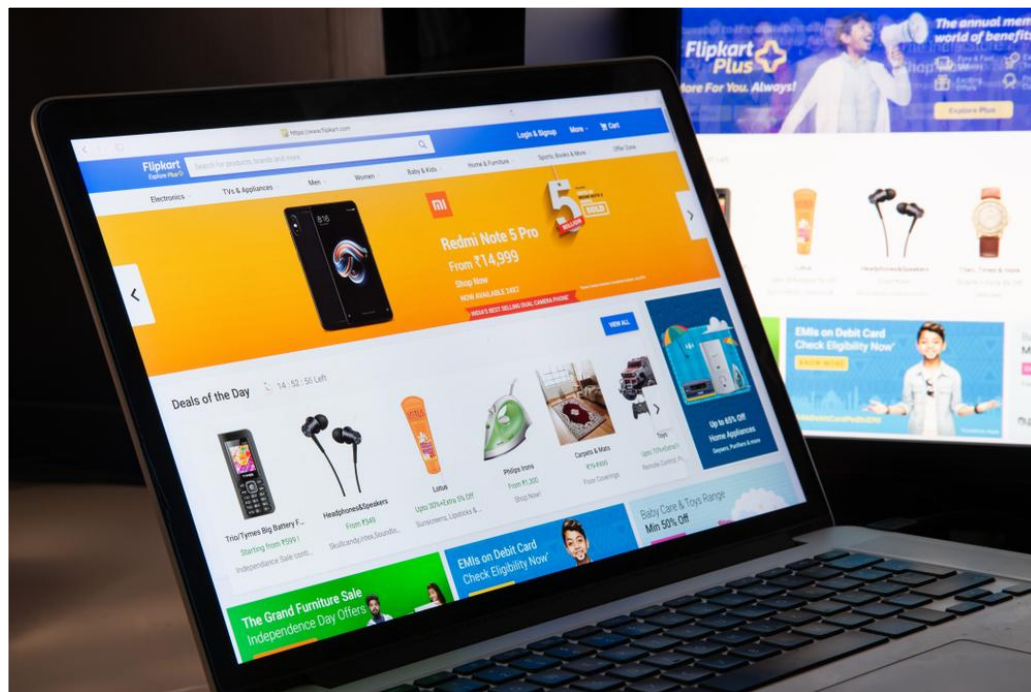
Which among the following options is a step in merge sort?

1. Selection
2. Partitioning
3. Pivoting
4. Merging

Poll 4 (Answer)

Which among the following options is a step in merge sort?

1. Selection
- 2. Partitioning**
3. Pivoting
- 4. Merging**



Have you ever tried to filter and sort your data on Flipkart?

What do you think happens here?

Let us now look at another smart sorting algorithm that makes use of the ‘divide and conquer’ paradigm. This sorting algorithm is known as the **Quick Sort**.

Quicksort takes inspiration from the working of Insertion Sort, introducing the divide-and-conquer function in the process. *The basic idea behind Quicksort is that it chooses a data element (the ‘pivot’) and places it in its correct place in the sorted version of the data set.*

While doing this, it also ensures that all the elements that are ‘lesser’ than the pivot are to its left side while the elements ‘greater’ than the pivot are to its right side.

The basic pattern of Quick Sort is similar to that of Merge Sort. Once again, we will essentially do the following:

- **Divide:** We choose a 'pivot' and then divide the array into two portions: the 'left' side, with all the data elements lesser than or equal to the 'pivot', and the 'right' side, which has all the elements that are greater than the 'pivot'. We must keep track of the index that the 'pivot' is put into.
- **Conquer:** We recursively sort the left and the right subsets of the 'pivot' by recursive calls to Quicksort.

Does anyone remember how we had arrived at a time complexity of $O(\log n)$ for Binary Search? We also saw how we were doing something similar for Merge Sort.

What would be the best case time complexity of Quicksort?
 $O(n \log n)$

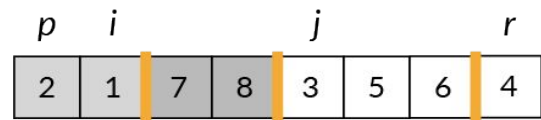
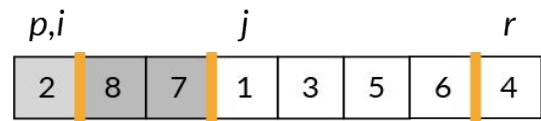
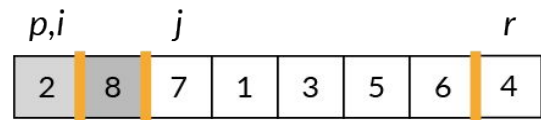
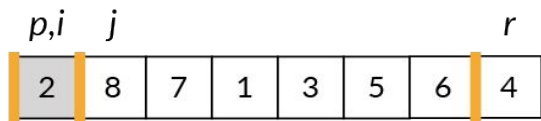
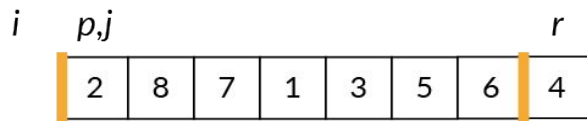


What would happen if we ran Quicksort on an already sorted array (or an array with only one element out of place)? What would be the time complexity in such case?
When, quicksort is applied on an sorted array, the worst case of quick sort occurs and the time complexity of it is $O(n^2)$

This shows us that there is an edge case where the worst-case performance of Quicksort degrades to $O(n^2)$. This is one of the major pitfalls of Quicksort. In order for us to yield better performance for these specific data sets, **choosing the pivot element** becomes extremely important.

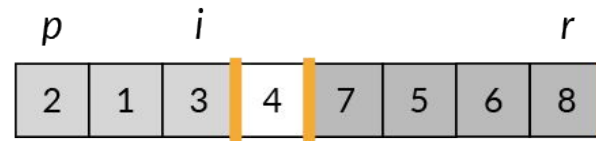
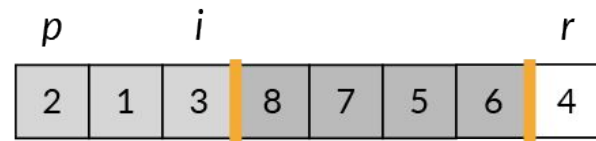
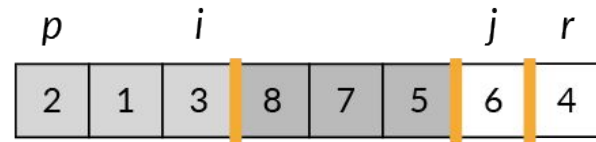
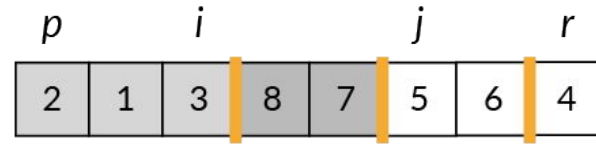
You can use several approaches in order to choose a pivot element while implementing Quicksort:

- The **first** element
- The **middle** element
- The **last** element, or even
- A **random** element



Here,

p is the lower element
 i is the index of smaller element
 j is the index of current element
and the the pivot element is 4



To sort an entire array A, the initial call is

```
quicksort(A, 1, length[A])
```

```
quicksort(A, p, r) {
```

```
    if (p<r) {
```

```
        q=partition(A,p,r)
```

```
        quicksort(A,p,q-1)
```

```
        quicksort(A,q+1,r)
```

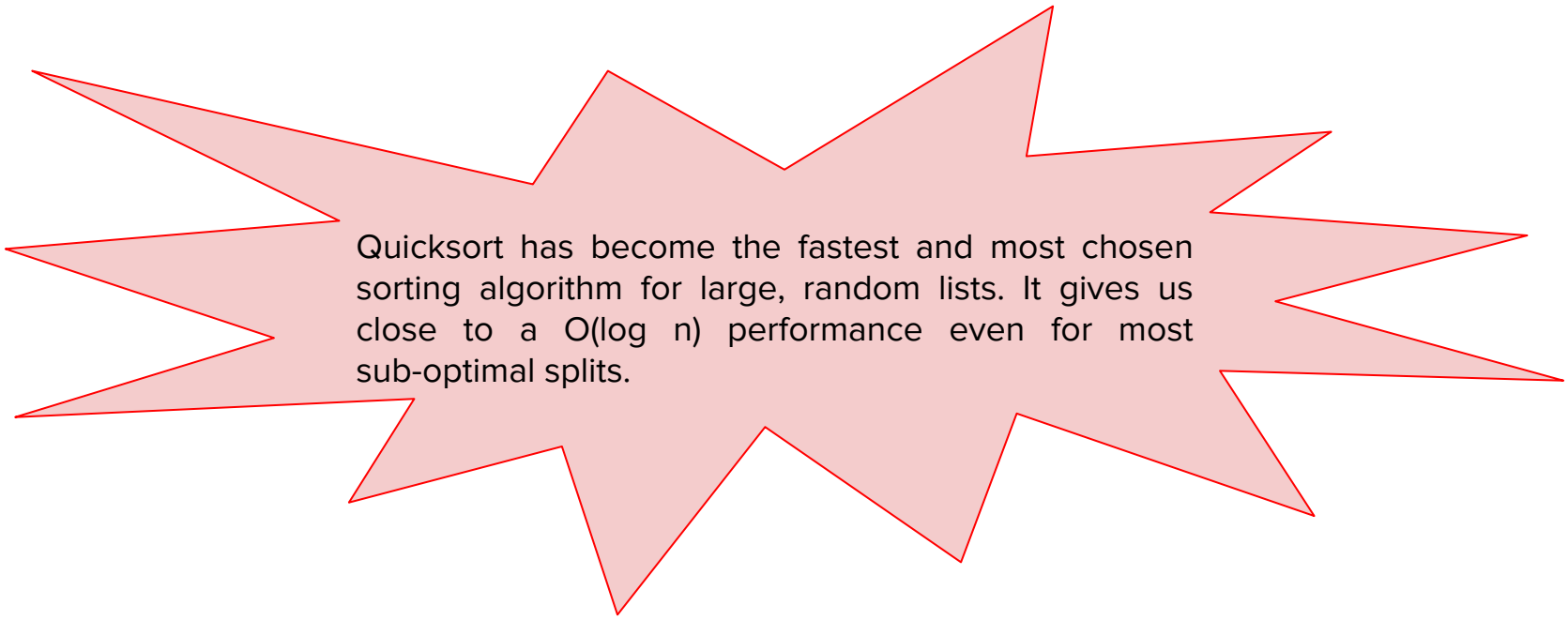
```
    }
```

```
}
```

```
partition(A,p,r) {  
    x=A[r]  
    i=p-1  
    for(j=p; j<=(r-1); j++) {  
        if (A[j] < x) {  
            i=i++;  
            exchange A[i] and A[j]  
        }  
    }  
    exchange A[i+1] and A[r]  
    return i+1  
}
```

Hands-on Coding

- [Implement quick sort in Java](#)

A red starburst shape with multiple points, containing text.

Quicksort has become the fastest and most chosen sorting algorithm for large, random lists. It gives us close to a $O(\log n)$ performance even for most sub-optimal splits.

Poll 5 (15 Sec.)

Is quick sort stable?

1. True
2. False

Poll 5 (Answer)

Is quick sort stable?

1. True
2. **False**

Poll 6 (15 Sec.)

Which of the following is correct about the merge sort?

1. Not in place
2. Comparison based
3. Unstable
4. Both 1 and 2

Poll 6 (Answer)

Which of the following is correct about the Merge Sort

1. Not in place
2. Comparison based
3. Unstable
4. **Both 1 and 2**

Hands-on Coding

- Sort an array in wave form. Elements at odd index should be smaller than the elements at even index.

e.g. [7,1,3,4,8,5] -> [7,1,4,3,8,5]

Poll 7 (15 Sec.)

What is the worst case time complexity of Quick Sort?

1. $O(n^2)$
2. $O(n)$
3. $O(n \log n)$
4. none

Poll 7 (Answer)

What is the worst case time complexity of Quick Sort

1. $O(n^2)$
2. $O(n)$
3. $O(n \log n)$
4. none

Poll 8(15 Sec.)

Which amongst the below options is a step in quick sort?

1. Selection
2. Partitioning
3. Pivoting
4. Merging

Poll 8(Answer)

Which amongst the below options is a step in quick sort?

1. Selection
2. Partitioning
3. Pivoting
4. Merging

What do you think are the disadvantages of using Divide and Conquer algorithm?

Disadvantages

The major limitation of DnC is that you may end up solving the same sub-problem multiple times. (ex: fibonacci numbers).

It quickly uses up the system stack and comes with all the inherent downsides of using recursion. If the 'base case' is not chosen carefully, you might run into stack overflow issues.

You can use this only if the subproblems have the same structure as the original problem. Otherwise, you cannot recursively solve it.

Poll 9 (30 Sec.)

Which of the following is not in-place sorting algorithm?

1. Bubble sort
2. Selection sort
3. Merge Sort
4. All

Poll 9 (Answer)

Which of the following is not in-place sorting algorithm

1. Bubble sort
2. Selection sort
- 3. Merge Sort**
4. All

Poll 10(15 Sec.)

Which of the following is not a stable sorting algorithm

1. Bubble sort
2. Insertion sort
3. Merge Sort
4. Quick Sort

Poll 10 (Answer)

Which of the following is not a stable sorting algorithm

1. Bubble sort
2. Insertion sort
3. Merge Sort
4. **Quick Sort**

Homework

1. Sort an array such that the first element is largest and then the smallest element. It has to be followed by second largest and second smallest element in array.

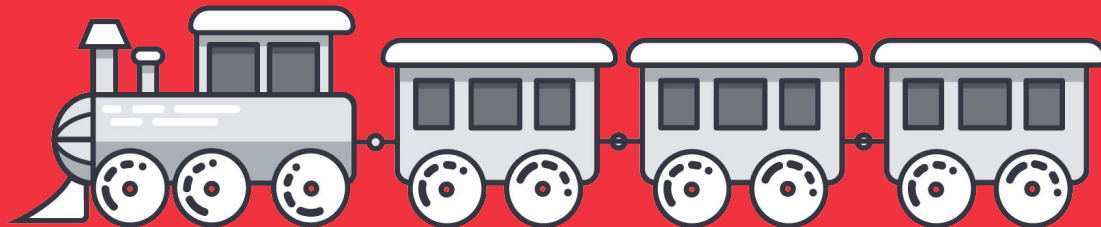
e.g. [6,1,4,2,9,3] -> [9,1,6,2,4,3]

Tasks to complete after the session

| |
|--------------------|
| Homework Questions |
| MCQs |
| Coding Questions |

In the next class...

- Introduction to Linked List
- Solve some problems involving singly Linked List





Thank You!