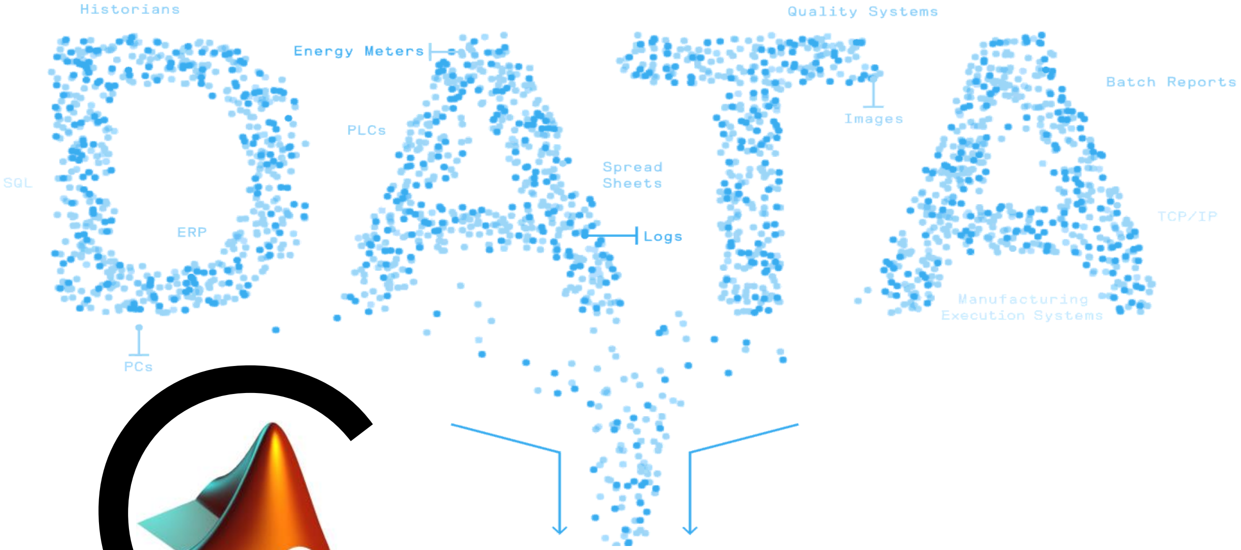
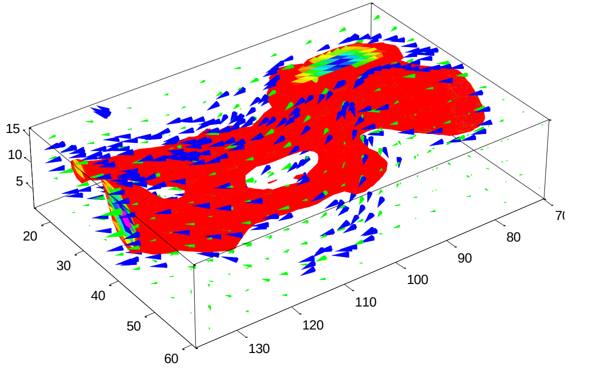
10mins Mock Lecture for Year 2 Undergraduate Students

Divide and Conquer Sorting Algorithms
- An Introduction



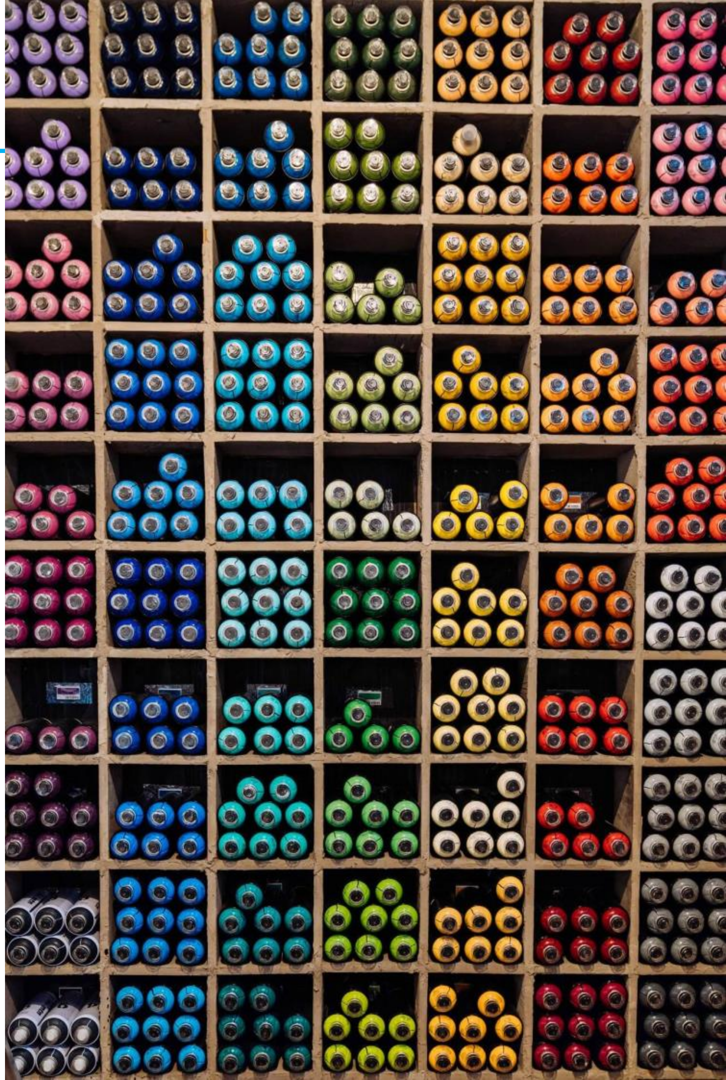


Dr Leo Chen
leo.chen@ieee.org
26/Oct/2021

Contents

1. What is ‘Divide and Conquer’
2. Basic Steps
3. Examples
4. Class Exercises

- FAQ
- Useful Links



What is ‘Divide and Conquer’

- Divide and conquer (D&C) is an algorithm **design paradigm** based on **multi-branched recursion**.
- A D&C algorithm works by **recursively** **breaking down** a problem into **two or more sub-problems** of the same or related type, **until** these become simple enough to be solved directly.

❑ **Idea 1: Merge sort -**

__Divide__ array into __two__ halves, recursively sort __left__ and __right__ halves, then __merge__ two halves as one array.

❑ **Idea 2: Quicksort -**

__Partition__ array into __small items__ and __large__ items, then recursively sort the two sets.

Basic Steps 01: Merge Sort

D&C steps:

1 Divide, Break up problem **into several parts**

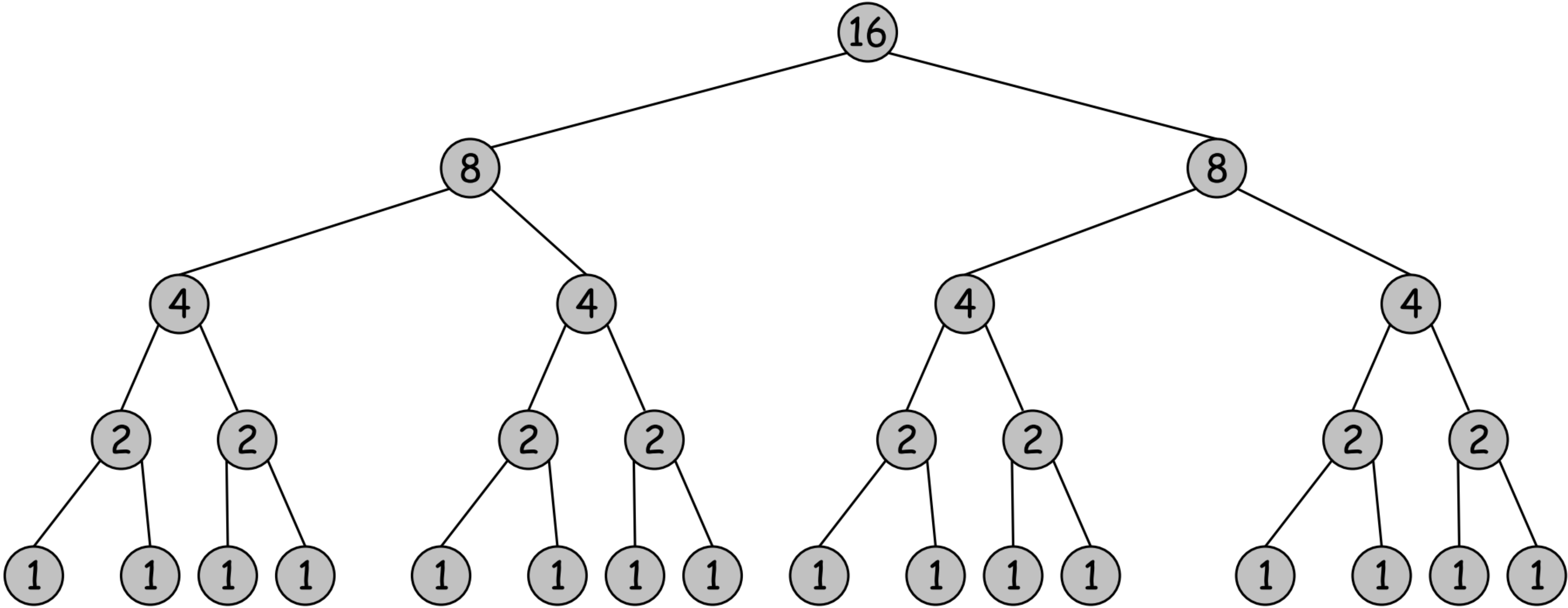
2 Sort, Solve each part recursively

3 Merge, Combine solutions to sub-problems **into** overall solution

Merge Sort :

1. **Divide**: break up problem of size **n** into **two** equal parts of size n/2.
2. **Sort**: solve **two** parts **recursively**
3. **Merge**: combine **two** solutions into **overall** solution in linear time.

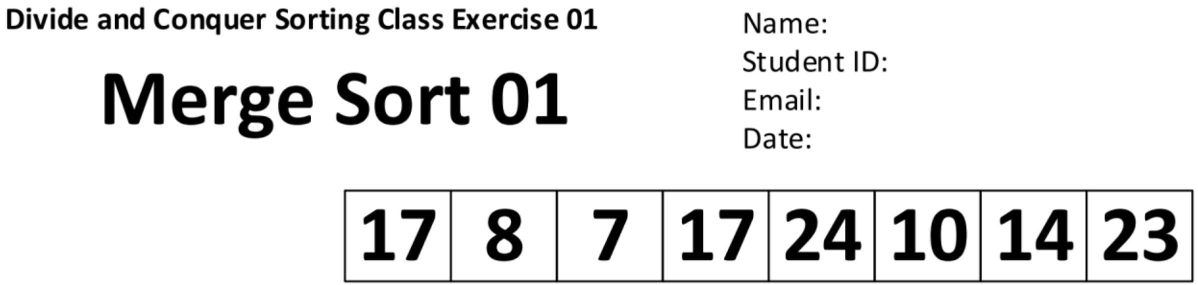
Call graph of merge sort of a string of length 16



Example 01: Merge Sort

6 5 3 1 8 7 2 4

Class Exercise 01 – Merge Sort 01



Divide and Conquer Sorting Class Exercise 01

Merge Sort 01

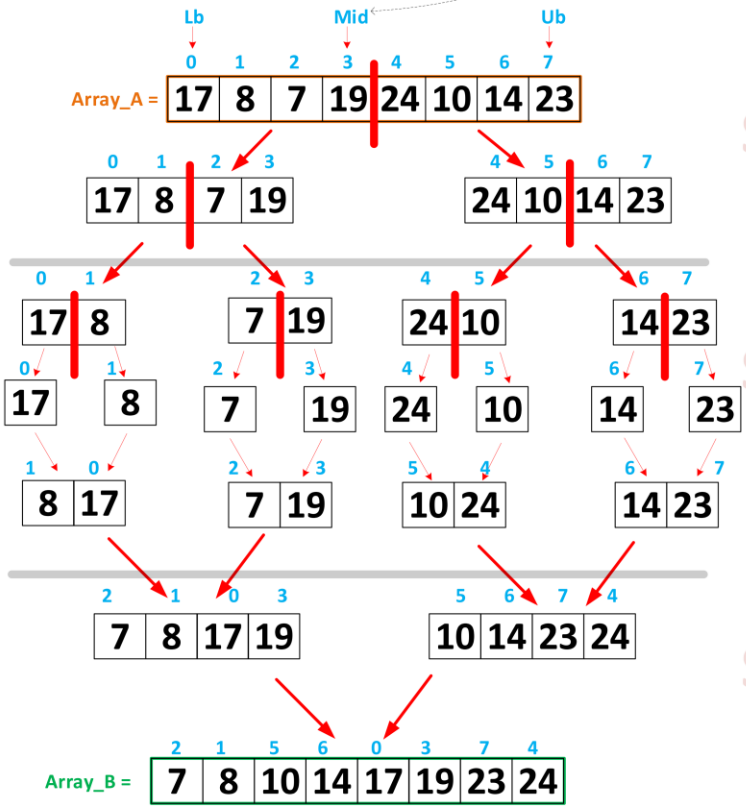
Name:
Student ID:
Email:
Date:

17	8	7	17	24	10	14	23

Divide and Conquer Sorting Class Exercise 01

Merge Sort 01

Name:
Student ID:
Email:
Date:



```
Mid = floor( (Lb + Ub)/2 )
    = floor( (0+7)/2 )
    = floor(3.5)
    = 3
```

```
MergeSort(Array_A, Lb, Ub)
```

Step1

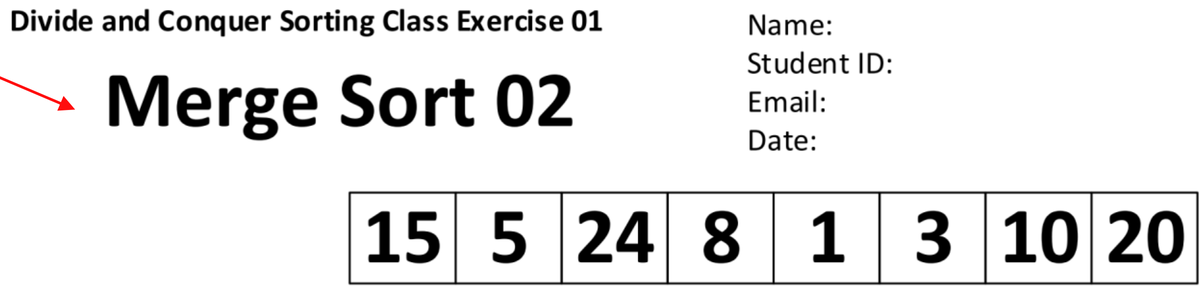
```
MergeSort(Array_A, Lb, Mid)

MergeSort(Array_A, Mid+1, Ub)
```

Step2

Step3

```
[ Array_B ] = MergeSort( Array_A, Lb, Ub )
```

Class Exercise 01 – Merge Sort 02 / 03

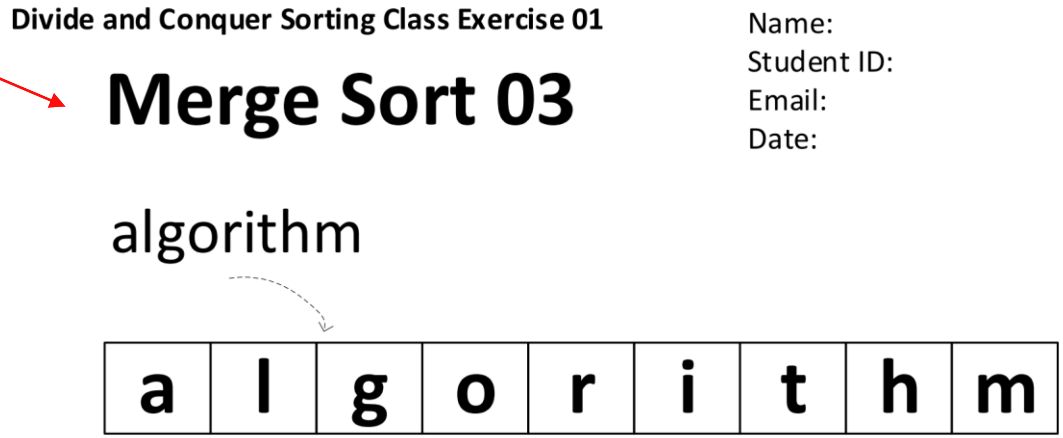


Divide and Conquer Sorting Class Exercise 01

Name:
Student ID:
Email:
Date:

Merge Sort 02

15	5	24	8	1	3	10	20



Divide and Conquer Sorting Class Exercise 01

Name:
Student ID:
Email:
Date:

Merge Sort 03

algorithm

a	l	g	o	r	i	t	h	m

Basic Steps 02: Quick Sort

1 Partition the elements into **three** categories based on a chosen **pivot** element:

- Elements **smaller**/**equal** to/**larger** ***than*** *the pivot*

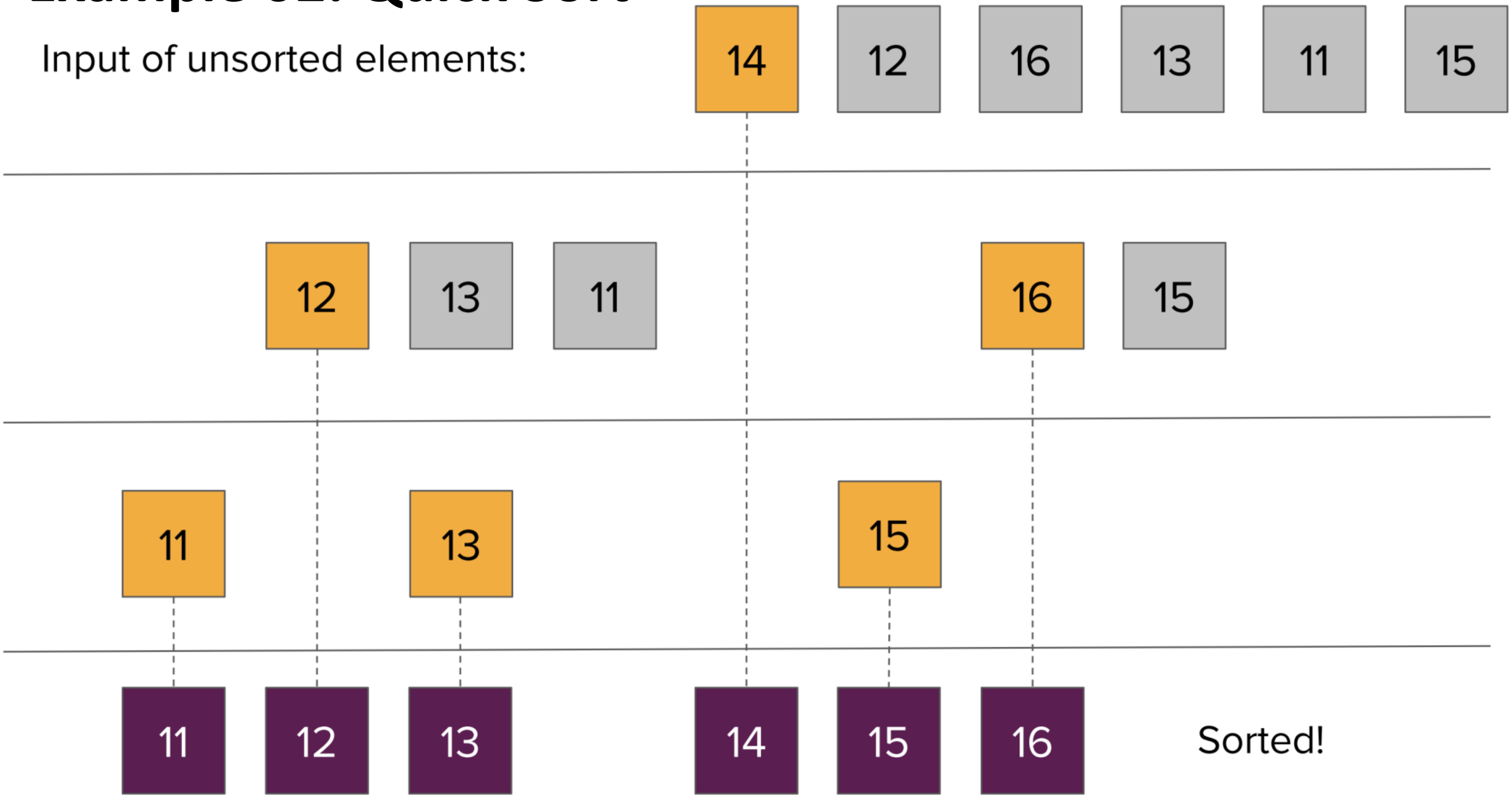
2 **Recursively** sort the two partitions that are **not** equal to the pivot (smaller and larger elements).

- Now our smaller elements are in sorted order, and our larger elements are also in **sorted** order!

3 **Connect** the three now-sorted partitions together.

Example 02: Quick Sort

Input of unsorted elements:



Sorted!

Class Exercise 02 – Quick Sort 01 / 02

Divide and Conquer Sorting Class Exercise 01

Quick Sort 01

17	8	7	17	24	10	14	23

Divide and Conquer Sorting Class Exercise 01

Quick Sort 02

Name:
Student ID:
Email:
Date:

15	5	24	8	1	3	10	20

FAQ

- FAQ 01 What is the difference between divide and conquer, and branch and reduce?
- FAQ 02 What does O(log n) mean exactly?
- FAQ 03 Sorting Definition
- FAQ 04 Merge Sort Coding
- FAQ 05 Complexity Chart

FAQ 01

Q1: What is the difference between divide and conquer, and branch and reduce?

A1: Divide and conquer algorithms divide the input.

Branch and reduce algorithms divide the solution space.

https://stackoverflow.com/questions/41140614/what-is-the-difference-between-divide-and-conquer-and-branch-and-reduce

FAQ 02

Q2: What does O(log n) mean exactly?

A2: For example, looking up people in a phone book is O(log n). You don't need to check every person in the phone book to find the right one;

instead, you can simply divide-and-conquer by looking based on where their name is alphabetically, and in every section you only need to explore a subset of each section before you eventually find someone's phone number.

https://stackoverflow.com/questions/2307283/what-does-olog-n-mean-exactly?noredirect=1&lq=1

FAQ 03 - Sorting

- Given a list of data points, sort those data points into **ascending** / descending order by some quantity.

FAQ 04 - Merge Sort Coding

```
void mergeSort(Vector<int>& vec) {
   /* A list with 0 or 1 elements is already sorted by definition. */
   if (vec.size() <= 1) return;

   /* Split the list into two, equally sized halves */
   Vector<int> left, right;
   split(vec, left, right);

   /* Recursively sort the two halves. */
   mergeSort(left);
   mergeSort(right);

   /*
    * Empty out the original vector and re-fill it with merged result
    * of the two sorted halves.
    */
   vec = {};
   merge(vec, left, right);
}
```

```
void mergeSort(Vector<int>& vec) {
   /* A list with 0 or 1 elements is already sorted by definition. */
   if (vec.size() <= 1) return;

   /* Split the list into two, equally sized halves */
   Vector<int> left, right;
   split(vec, left, right);

   /* Recursively sort the two halves. */
   mergeSort(left);
   mergeSort(right);

   /*
    * Empty out the original vector and re-fill it with merged result
    * of the two sorted halves.
    */
   vec = {};
   merge(vec, left, right);
}
```

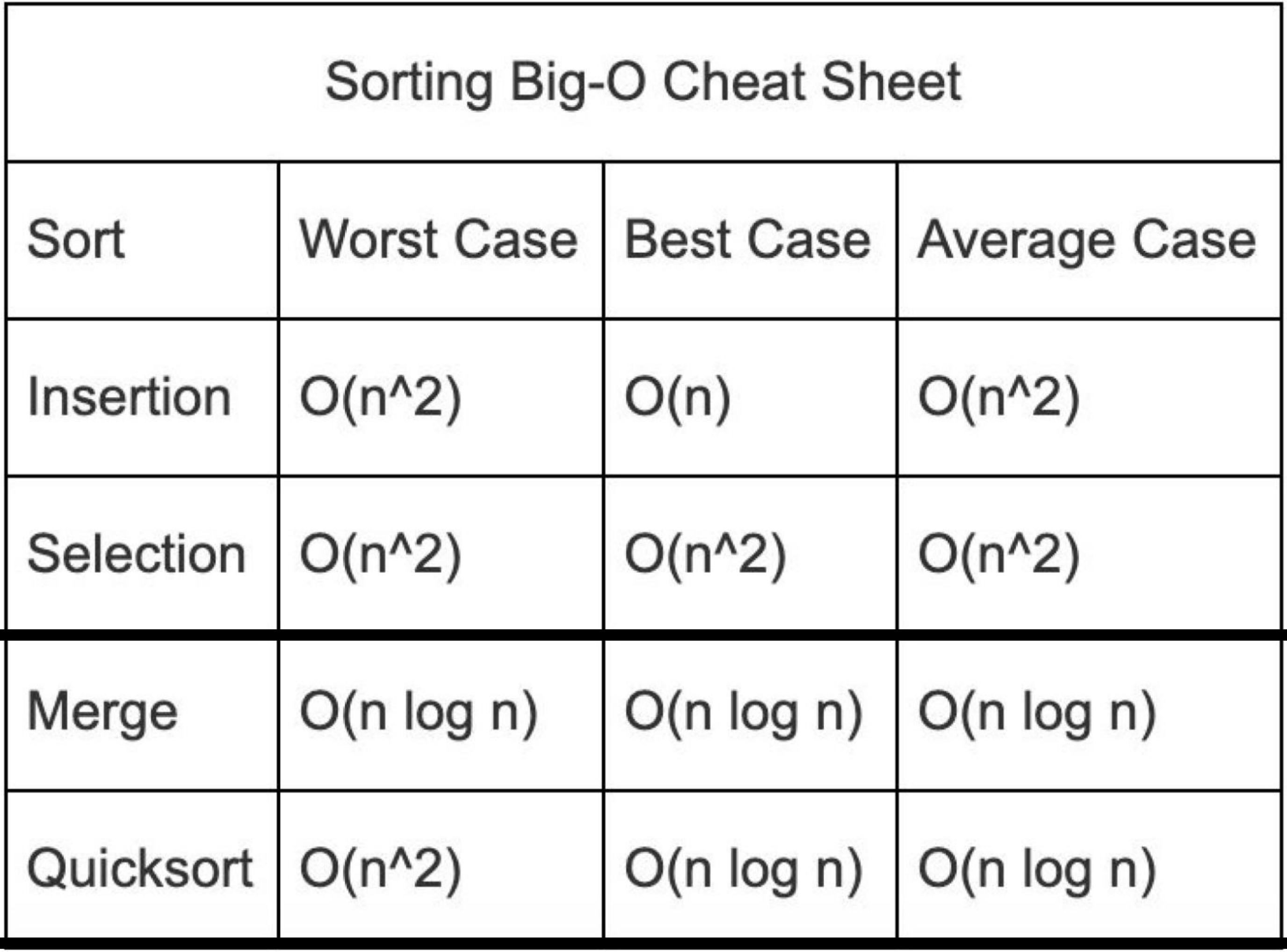
O(n) work

O(n) work

```
void mergeSort(Vector<int>& vec) {
   /* A list with 0 or 1 elements is already sorted by definition. */
   if (vec.size() <= 1) return;

   /* Split the list into two, equally sized halves */
   Vector<int> left, right;
   split(vec, left, right);

   /* Recursively sort the two halves. */
   mergeSort(left);
   mergeSort(right);

   /*
    * Empty out the original vector and re-fill it with merged result
    * of the two sorted halves.
    */
   vec = {};
   merge(vec, left, right);
}
```

O(n log n) work

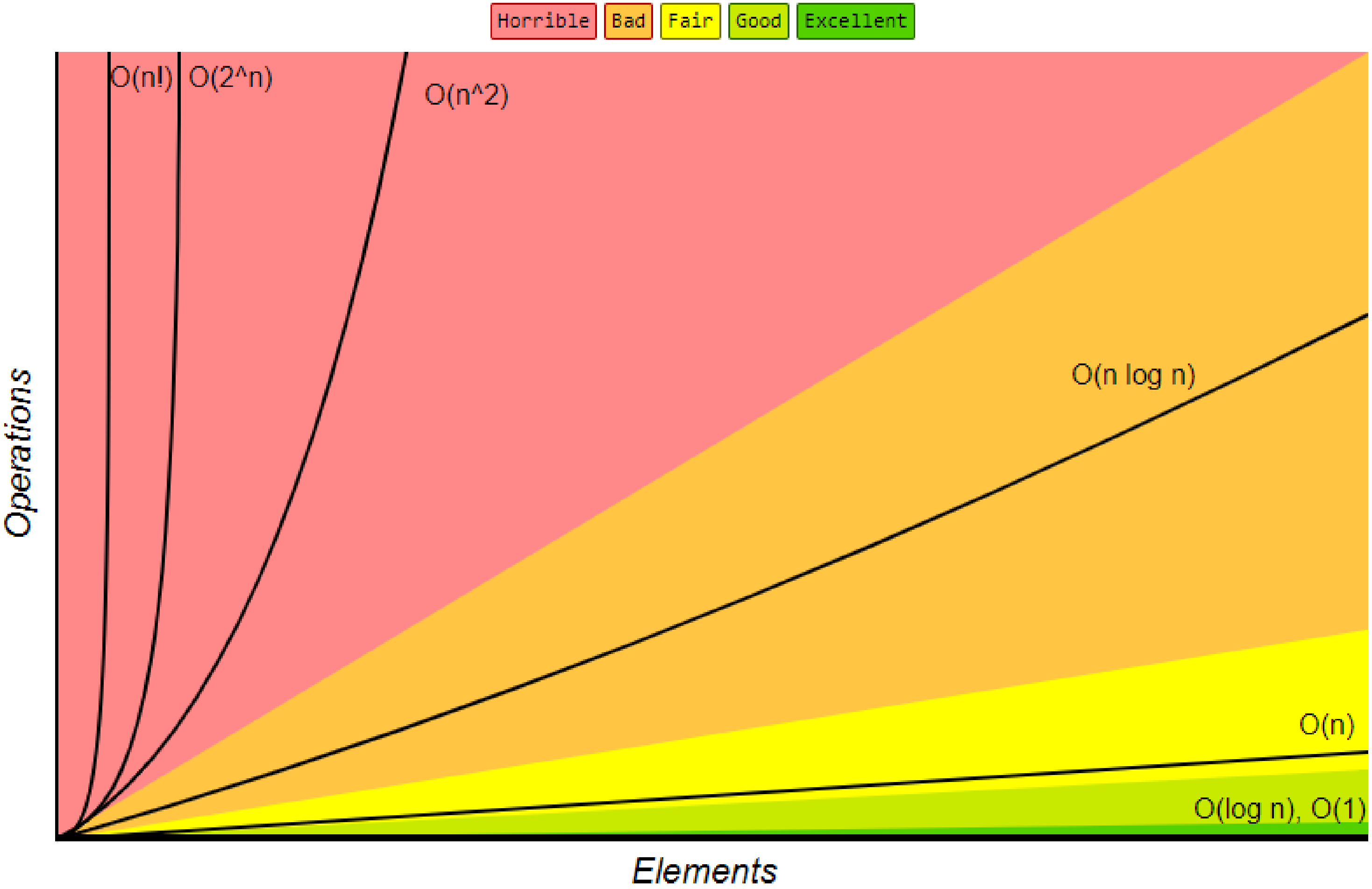
FAQ 05 – Complexity Chart



Sorting Big-O Cheat Sheet

Sort	Worst Case	Best Case	Average Case
Insertion	O(n^2)	O(n)	O(n^2)
Selection	O(n^2)	O(n^2)	O(n^2)
Merge	O(n log n)	O(n log n)	O(n log n)
Quicksort	O(n^2)	O(n log n)	O(n log n)

Big-O Complexity Chart



O(n)

- For example, the following function is O(n) because the algorithm grows in proportion to its input n:

```
f(int n) {
  int i;
  for (i = 0; i < n; ++i)
    printf("%d", i);
}
```

O(n^2)

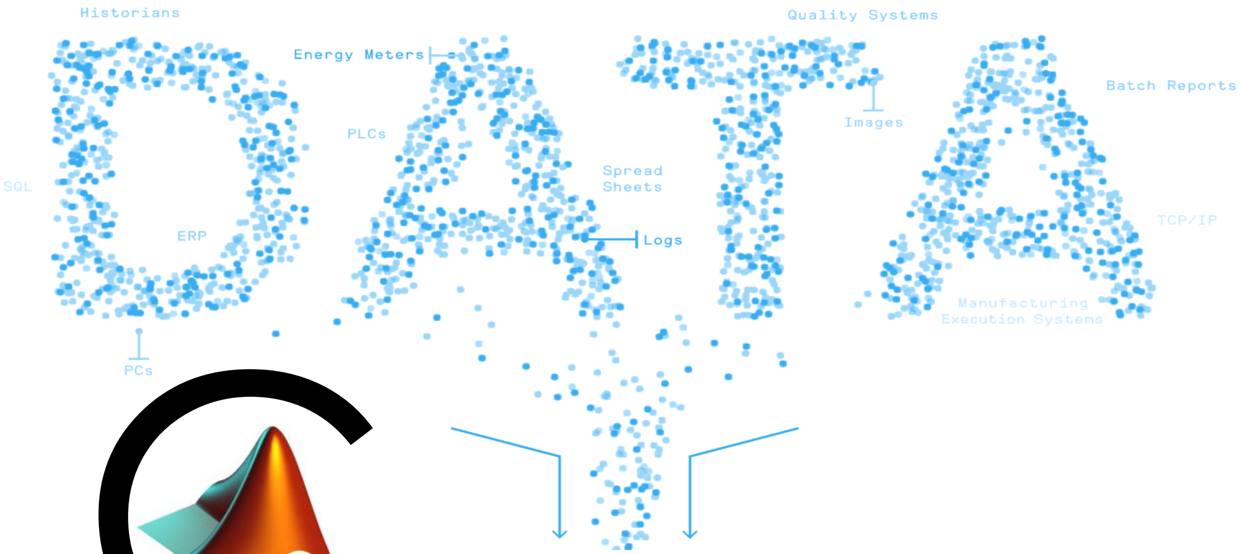
- a nested loop

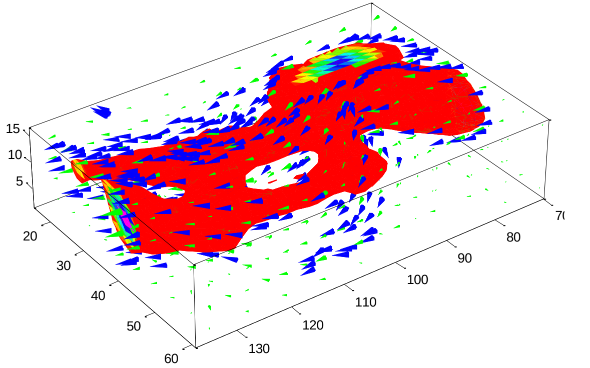
```
f(int n)
{
  int idx, jdx;
  {   for (idx = 0; idx < n; ++idx)
       {     for (jdx = 0; jdx < n; ++jdx)
              printf("%d", jdx);  }
        printf("%d", idx);                  }
}
```

Useful Links

- https://www.freecodecamp.org/news/sorting-algorithms-explained-with-examples-in-python-java-and-c/
- https://www.coursera.org/learn/algorithms-divide-conquer
- https://github.com/TheAlgorithms/Python
- https://www.quora.com/Which-one-is-more-complex-%D0%9E-log-N-or-%D0%9E-N-log-N
- https://www.toptal.com/developers/sorting-algorithms/merge-sort
- https://mp.weixin.qq.com/s/_j9PIG22JNVhxHBNAV9_xQ

Divide and Conquer Sorting Algorithms - An Introduction





Dr Leo Chen
leo.chen@ieee.org
26/Oct/2021

Thanks and Questions