☼Hi, welcome to Algorithm design: sorting.

As we know, **Computational thinking is the skill of 21st century. what is Computational thinking? it is a problem solving process, which involves formulating a problem in such a way that its solution can be effectively executed using a computer.**
You also learnt that Algorithm is a step-by-step description of a means to solve a problem.

How to design an algorithm? We will choose some important algorithms to illustrate the concept of algorithm design.

For someone who is learning to write a program, what are the most important algorithms? That rather depends on how the person defines the term "important". Since there are many ways an algorithm can be important, we will be dealing with the most common algorithms applied in programming. Sorting and searching. We will be using sorting as an example in this lesson, and the searching in the next lesson. None of these methods requires a knowledge of advanced mathematics or data structures.

First, let's look at SORTING:

☼We all know that Google maps can sort results by categories and rating.

☼Likewise, in Windows/ File Explores, you can also do sorting. You can quickly sort files and folders by size, name, or Date modified

☼Most people know what sorting is and one can sort a small sequence of numbers in a few seconds. Recall our scenario of Finding the Distance to N Locations, add one more feature, sort them according to the distance. In a few seconds, it is done.

Each may have a distinct strategy for doing it, but few can explain to someone how to sort an arbitrary set of numbers. They themselves may not know how they do it; they can simply tell when something is sorted and have some process for sorting in mind. In short, the process of sorting is one of the simplest things that is hard to describe.

☼At the end of this lesson, you should be able to.

- Describe the process of sorting
- Explain the importance of different types of sorting algorithms
- Perform sorting algorithms particularly sorting alphabetically or numerically
- Sort an array using bubble sort and merge sort
- Apply your knowledge and understanding of sorting algorithms to your problem solving
- Recognize that "no single" best sorting applies to all scenarios

☼We will go through the following topics in this lesson.

☼In computer science, a sorting algorithm is an algorithm that puts elements of a list in a certain order. The most frequently used orders are numerical order and alphabetical order. Efficient sorting is important for optimizing the efficiency of other algorithms, for example, search algorithms which require input data to be in sorted lists. Sorting is also often useful for producing human-readable output.

Sorting involves placing things in an order defined by a function that ranks them somehow. For numbers, ranking means using the numerical value. So: the sequence 1, 3, 2 is not in proper order, but 1, 2, 3 is in ascending (getting larger) order and 3, 2, 1 is in descending (getting smaller) order.

So how can a sequence be placed in a sorted order? By using a sorting algorithm,

of course. For all the discussions in this lesson, assume that the problems involve sorting in ascending order.

☼Because sorting is so important in computer science, it has been studied at great length. Different sorting algorithms are available. There is no single best sorting algorithm. The efficiency of a sorting algorithm is based on the number of comparisons it takes during sorting. In this lesson, we will only introduce the bubble sort and merge sort.

☼Bubble sort, which is sometimes referred to as sinking sort, is one of the simplest sorting algorithms that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list.

It is visualized in this video. (Click the video icon)

☼ Our purpose is to sort the list in ascending order. That is, let larger elements ‘bubble” to the top of the list.

The **bubble sort** makes multiple passes through a list.

For each pass, the bubble sort is operated as follows:

Compare the first two numbers, and if the second number is smaller than the first, then the numbers are swapped. In this example, 26<54, so swap. (click video icon)

Then move down one number, compare the number and the number that follows it, and swap the two numbers, if necessary. In this example, third number is larger than second number, no swapping required.

Repeat this process, compare the third number and the fourth number, fourth number is smaller than third number, so swap. (click video icon)

☼ The process keeps repeating until last two numbers of the array have been compared. (click video icon)

Largest number has now moved to the bottom of the list.

Each sequence of comparison is called a pass.

☼ At the start of the second pass, the largest value is now in place. However, the list is yet completely in the ascending order. We still need to perform the second pass through the list. There are n-1 items left to sort. meaning that there will be n−2 pairs.

At the end of the second pass, the list of numbers now looks like the following.

☼ Since each pass places the next largest value in place, the total number of passes necessary will be n−1. After completing the n−1 passes, the smallest item must be in the correct position with no further processing required. That is all numbers are in the correct order.

☼ If there were a “best” sorting algorithm then this would be the place to describe it. As there is not, perhaps the best thing to do would be to look at an algorithm that is quite different from the bubble sort.

Merge sort is one of the popular sorting algorithms as it uses the minimum number of comparisons. It is an example of a *divide and conquer* style of algorithm, in which a problem is repeatedly broken up into sub-problems, often using recursion, until they are small enough to solve; the solutions are combined to solve the larger problem. The idea behind merge sort is to break the data into parts that can be sorted trivially, then combine those parts knowing that they are sorted.

Here’s a simple video to help you visualize the merge sort (click video icon)

☼ From the video, we know that there is an important function used in merge sort. That is to merge two sorted lists. Let’s work out the algorithm. List 2 4 9 and 3 5 6 8 are used as an example to illustrate the process step by step.

Step 1. Compare the first elements of both lists one by one. Compare 2 and 3.

Step 2. Move the smaller element out of the list it was found in. Add this value to the list of “sorted items”. 2 is smaller and add to sorted list.

Step 3. Repeat the Process Until Only a Single List is Left Remaining. Compare 4 and 3, 3 is smaller, move to sorted list, then compare 4 and 5, Repeat the process until one list is empty.

Step 4. One list should still contain elements. This list is sorted. Move its contents into the result list.

The final sorted list is 2345689.

☼ Using the a sample list 2 9 4 3 5 8 6 to show how merge sort works. the first step in the merge sort is to split the data into two parts. There are 7 elements in this list, and the middle

element would be at 7//2, or third element, which is number 4, (to view animation)so the two parts are: 2 9 4 and 3 5 8 6.

☼ Splitting again, the first set has 3 elements, and the middle element would be at 3//2=1, (to view animation), so the two parts are: 2 and 9 4.

☼ 2 is individual component and cannot be further split. 9 4 is further split (to view animation) into 9 and 4. The final split breaks the data into individual components:

☼ Now that the individual elements are available, it is easy to sort them, as pairs. On the lower right the pair [9] and [4] is out of order, so they must be swapped with each other. (to view animation) Now they are sorted, the result is 4 9.

☼ Moving up again, the singleton [2] is merged with the pair [4, 9] using merging function by looking at the beginning of each list and copying the smallest element of the pair into a new list. 2 is copied to new list and left list is empty. So all numbers in right list are appended to new list, the merge result is 2,4,9

☼ The same dividing and merging process applies to the right list 3 5 8 6 . Split the data into two parts 3 5 and 8,6. Splitting again, 3, 5 split into two individual components 3 and 5,

Merge 3 and 5, they are in order, so no need to swap. The result is 3 5.

Split 8 6 into two individual components 8 and 6.

Merge 8 and 6, the pair [8] and [6] is out of order, so they must be swapped with each other. Now they are sorted, the result is 6 8.

☼ At each stage, the lists contain more elements and they are sorted internally, with the smallest element at the beginning. As illustrated before , merge two sorted lists is simply a matter of looking at the elements at the beginning of each and copying the smallest one to the result until the lists are empty. In this example, merge 3 5 and 6 8. 3 is less than 6. So copy 3 to the sorted list, then compare 5 and 6 ., 5 is smaller, copy to the sorted list and the left list is empty now. Append rest of the numbers, which are 6 and 8 to the sorted list. The merging result is 3 5 6 8.

☼ The highest level that the list can achieve is by merging 2,4,9 and 3 5 6 8. We have illustrated it before. So, the final sorted list is 2 3 4 5 6 8 9.

Now let's look back the whole process, once the data has been split into individual components, the merge stage creates sorted pairs, the next merge creates sets of 4 sorted numbers, the next 8, and so on, doubling each time until they are all sorted. A logical way to write the program is to use recursion,

☼ Quick summary.

Sorting: an operation that segregates items into groups according to specified criterion, which is so important in computer science that it has been studied at great length.

- **bubble sort**: orders a list of values by repetitively comparing neighboring elements and swapping their positions if necessary
- It is Easier to implement, but slower than any other sort.
- Merge sort algorithm is an important divide-and-conquer sorting algorithm.
- It is a recursive algorithm with the following steps
  - Divide the list into halves,
  - Sort each halve separately, and
  - Then merge the sorted halves into one sorted array.
- Merge sort is an extremely efficient algorithm with respect to time.
- Different algorithms have different asymptotic and constant-factor trade-offs
- No single ‘best’ sort for all scenarios
- Knowing one way to sort, just isn’t enough.

I hope you enjoy this lecture.

And, I will see you next time.