Lecture

NOTE: FOR FURTHER DETAILS AND MORE COMPREHENSIVE STUDY, PLEASE SEE RECOMMENDED BOOKS OR INTERNET.

Sorting

Sorting is nothing but arranging the data in ascending or descending order. The term ***sorting*** came into picture, as humans realized the importance of searching quickly.

There are so many things in our real life that we need to search for, like a particular record in database, roll numbers in merit list, a particular telephone number in telephone directory, a particular page in a book etc. All this would have been a chaos if the data was kept unordered and unsorted, but fortunately the concept of sorting came into existence, making it easier for everyone to arrange data in an order, hence making it easier to search.

Sorting arranges data in a sequence which makes searching easier.

Sorting Techniques

Sorting is ordering a list of objects. We can distinguish two types of sorting. If the number of objects is small enough to fits into the main memory, sorting is called ***internal sorting***. If the number of objects is so large that some of them reside on external storage during the sort, it is called ***external sorting***. In this course we will focus on the internal sorting algorithms.

Sorting Efficiency

Take an example, how will you arrange a deck of shuffled cards in order? you will start by checking every card, and making the deck as you move on. It can take you hours to arrange the deck in order, but that's how a human will do it.

WELL, COMPUTERS DON'T WORK LIKE THIS.

Since the beginning of the programming age, computer scientists have been working on solving the problem of sorting by coming up with various different algorithms to sort data.

The two main criteria to judge which algorithm is better than the other have been:

- Time taken to sort the given data. ***(Time Complexity)***
- Memory Space required to do so. ***(Space Complexity)***

Bubble Sort

The algorithm works by comparing each item in the list with the item next to it, and swapping them if required. In other words, the largest element has bubbled to the top of the array. The algorithm repeats this process until it makes a pass all the way through the list without swapping any items.

It is known as ***bubble sort***, because with every complete iteration the largest element in the given array, bubbles up towards the last place or the highest index, just like a water bubble rises up to the water surface. The worst-case runtime complexity is ***$O(n^2)$***.

Example: Here is one pass of the algorithm. The largest element **‘7’** is bubbled to the top:

7, ***5***, 2, 4, 3, 9
5, ***7***, ***2***, 4, 3, 9
5, 2, ***7***, ***4***, 3, 9
5, 2, 4, ***7***, ***3***, 9
5, 2, 4, 3, ***7***, ***9***

After only 1 pass the first largest value bubbled up to the last index.
5, 2, 4, 3, 7, ***9***

The process will continue pass for every value until all the items have been bubbled to the top.

Code for Bubble Sort:

Below is the code for a method of bubble sort.

```java
public void bubbleSort(int arr[])
{
    int n = arr.length;

    for (int i = 0; i < n; i++)
    {
        for (int j = 1; j < n; j++)
        {
            if (arr[j-1] > arr[j])
            {
                int temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

The code can be modified to make it more efficient however it will not improve the complexity of the algorithm.

```java
public void bubbleSort(int arr[])
{
    int n = arr.length;

    for (int i = n-1; i >= 0; i--)
    {
        for (int j = 1; j <= i; j++)
        {
            if (arr[j-1] > arr[j])
            {
                int temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```