Arrays,
2D Arrays,
Sorting

Monika Ketepally

Arrays

What is array?

Array is a Linear data structure which is a collection of data items having similar data types stored in contiguous memory locations. By knowing the address of the first item we can easily access all items/elements of an array. Arrays and its representation is given below.

Why do we need array?

Suppose we have to store 100 numbers. Creating 100 variables is really a arduous job. Creating an array to store those 100 numbers is lot more easier than to create 100 variables.

Advantages of arrays

- Storage of variables in contiguous memory locations.
- Accessing a particular variable from the group is easy.
- An array can have any number of dimensions such as 1,2,3..etc.
- Using arrays, other data structures like linked lists, stacks, queues, trees, graphs etc can be implemented.
- Two-dimensional arrays are used to represent matrices.

Disadvantages of arrays

- In C/C++/python length of an array has to be mentioned during the time of creation and can’t be changed later i.e, array will have fixed size, so size should be known priorly.
- Insertion and deletion are quite difficult in an array as the elements are stored in consecutive memory locations and the shifting operation is costly.
- Allocating more memory than the requirement leads to wastage of memory space and less allocation of memory also leads to a problem.

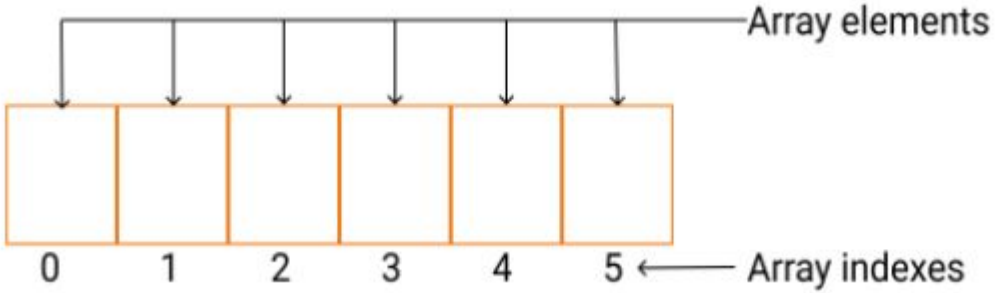
Types of arrays

1. One-Dimensional Array
2. Two-Dimensional Array
3. Multi-Dimensional Array

One-Dimensional array

- In one-dimensional array, all elements are arranged in linear foam as list.
- The elements of an array are accessed by using an index.
- The index of an array of size N can range from 0 to N−1.
- First element of array can be accessed by **a[0]** which is internally decoded by the compiler as ***(a + 0).** Same goes with other array elements.

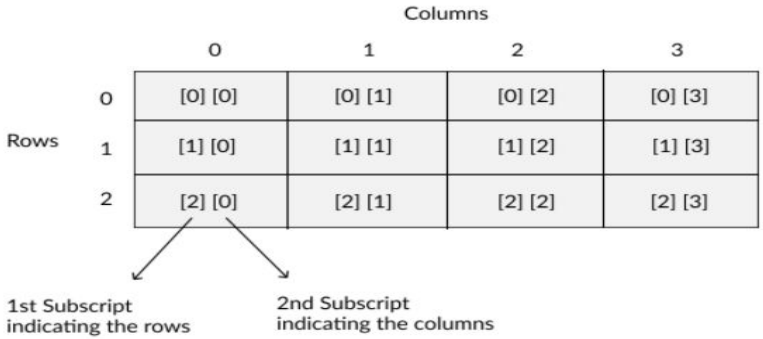
Example:



Two- Dimensional array

- A two-dimensional array is an array of arrays.
- 2-dimensional arrays are the most commonly used. They are used to store data in a tabular manner.
- If an array has dimensions of m x n size, number of elements present in the array are : m*n where indexes start from [0][0] and have till [m-1][n-1].

- Example



How to create 1-d array?

In C/C++ : datatype variableName[lengthOfArray];

In Java : datatype [] variableName = new datatype[lengthOfArray];

How to create 2-d array?

In C/C++ : datatype variableName[lengthOf1dOfArray][lengthOf2dOfArray];

In Java : datatype [][] variableName = new datatype[lengthOf1dOfArray][lengthOf2dOfArray];

Array Operations

- **Traverse** − Print all the elements in the array one by one.
- **Insertion** − Adds an element at the given index.
- **Deletion** − Deletes an element at the given index.
- **Search** − Searches an element in the array using the given index or the value.
- **Update** − Updates an element at the given index.

Applications of arrays

- Used to Perform Matrix Operations.
- Used to implement Search Algorithms.
- Used to implement Data structures like stacks, queues, heaps, hash tables.
- Used to implement CPU Scheduling Algorithms.

Sorting

A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements.

Few popular sorting algorithms are:

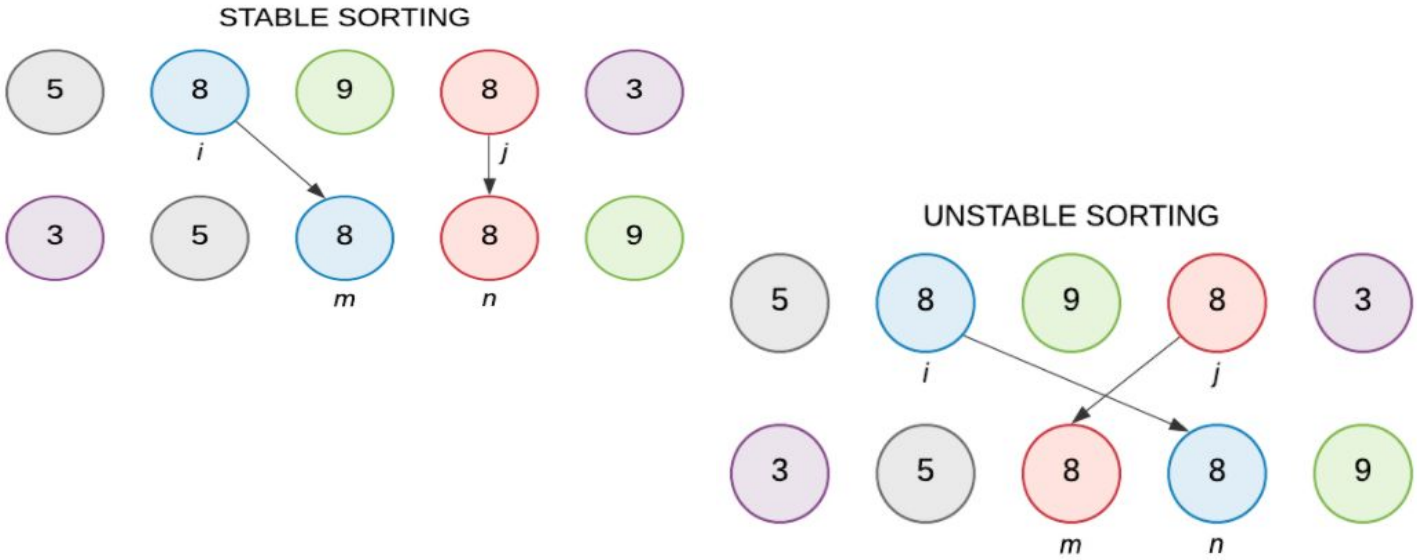
- Bubble sort
- Selection sort
- Insertion sort
- Merge sort
- Quick sort
- Count sort
- Heap sort
- Radix Sort
- Shell sort

Stable sorting algorithms

A sorting algorithm is said to be stable if two objects with equal keys appear in the same order in sorted output as they appear in the input array to be sorted.

- Some Sorting Algorithms are stable by nature, such as Bubble Sort, Insertion Sort, Merge Sort, Count Sort etc.
- Comparison based stable sorts such as Merge Sort and Insertion Sort, maintain stability by ensuring that,
  Element A[j] comes before A[i] if and only if A[j]<A[i], where i<j .
- Other non-comparison based sorts such as Counting Sort maintain stability by ensuring that the Sorted Array is filled in a reverse order so that elements with equivalent keys have the same relative position.
- Some sorts such as Radix Sort depend on another sort, with the only requirement that the other sort should be stable.

- Even though few sorting algorithms such as Quick Sort, Heap Sort etc., are unstable by their nature, they can be made stable by also taking the position of the elements into consideration.
- Example



Overview of different sorting algorithms

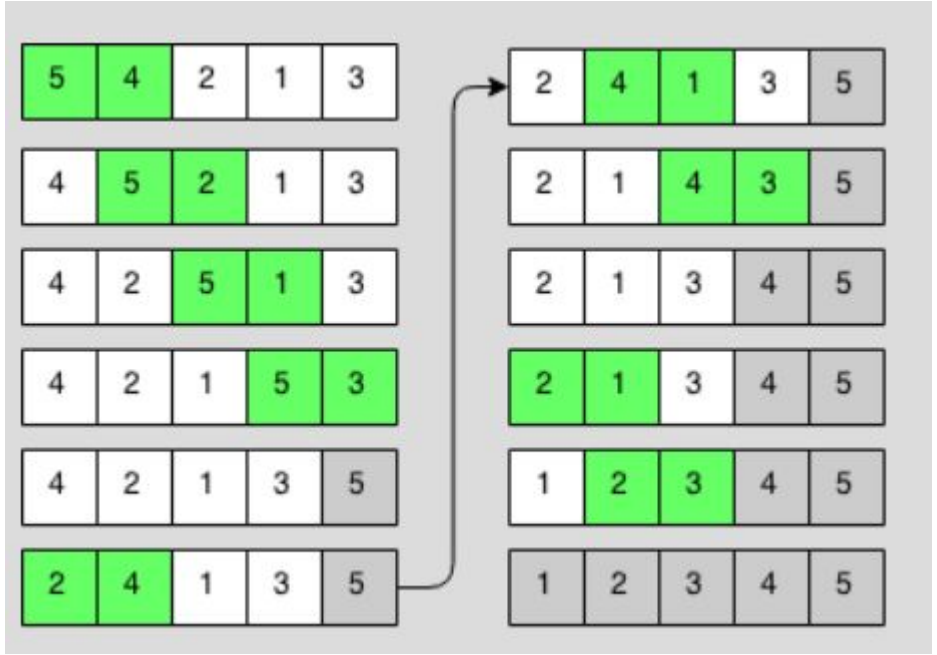
Bubble sort:

- This works by repeatedly swapping the adjacent elements if they are not in the order required.
- Time complexity : O(N^2),

Space complexity : O(1)

- Code snippet:

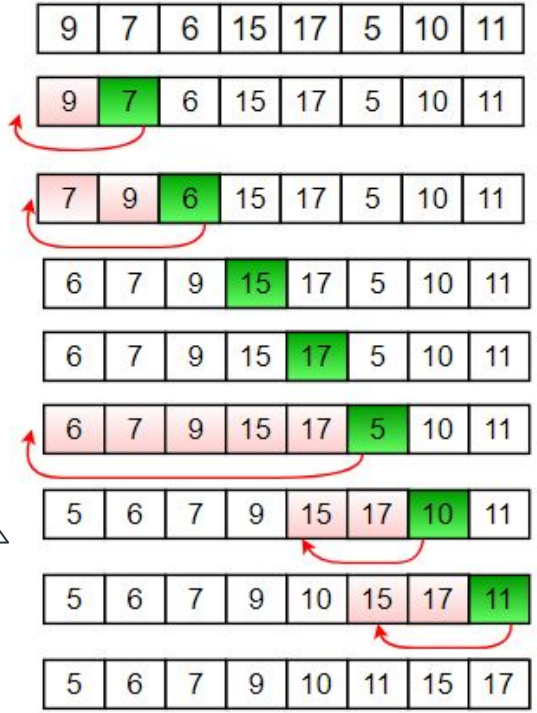
```
for (i = 0; i < n-1; i++) {
    for (j = 0; j < n-i-1; j++){
        if (arr[j] > arr[j+1])
            swap(&arr[j], &arr[j+1]);
}}
```



Insertion sort

- In this sorting algorithm, the array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.
- Time Complexity : O(N^2), Space Complexity : O(1)
- Code Snippet

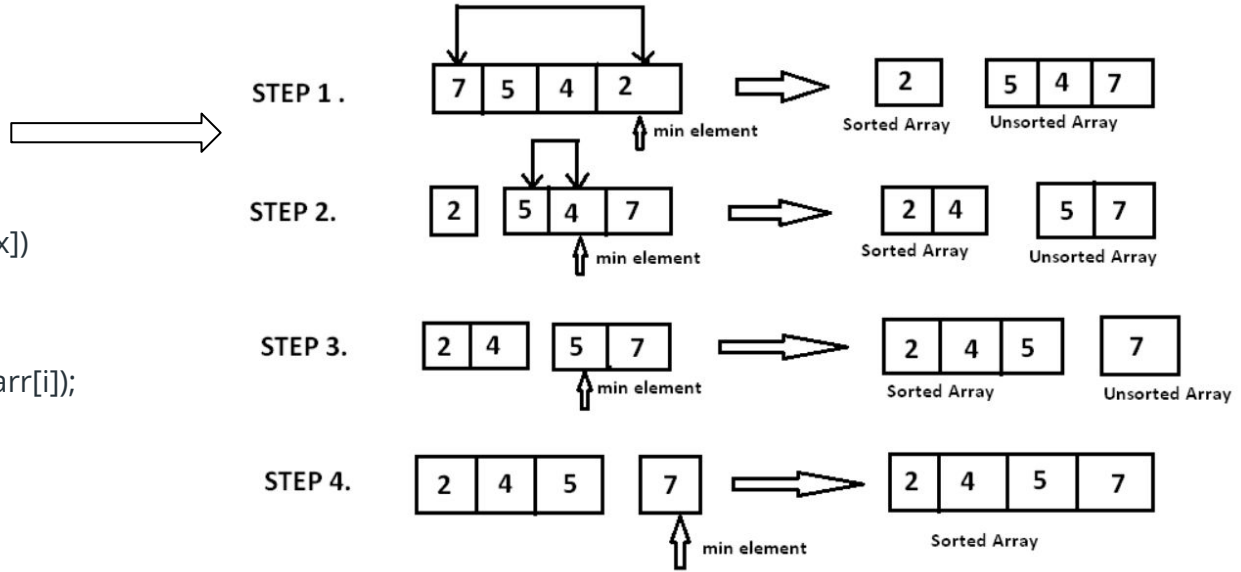
```
for (i = 1; i < n; i++){
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key){
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
```



Selection sort

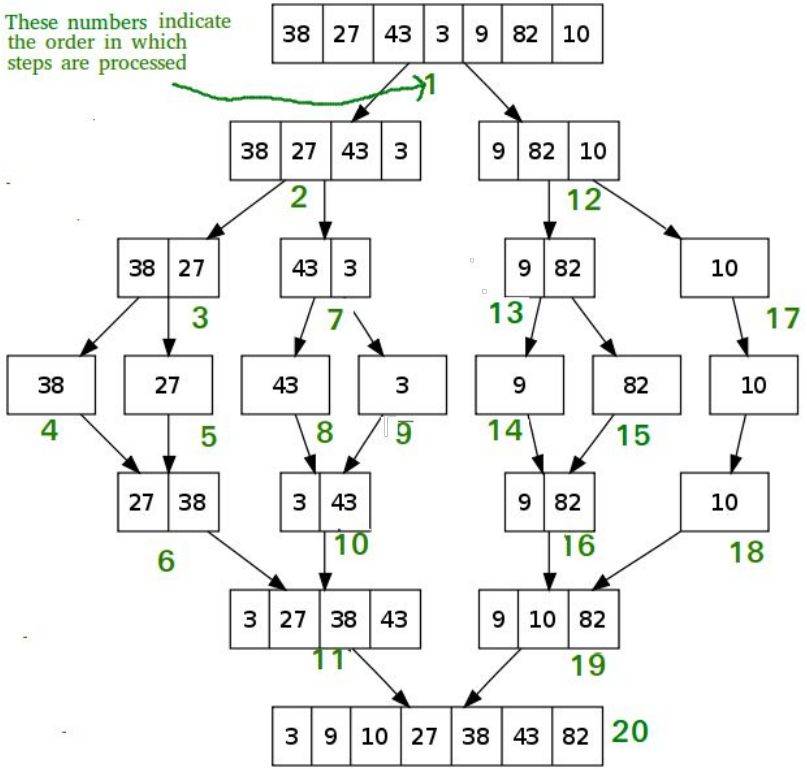
- In this sorting algorithm, array is sorted by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.
- Time Complexity : O(N^2), Space Complexity : O(1)
- Code Snippet

```
for (i = 0; i < n-1; i++){

      min_idx = i;

      for (j = i+1; j < n; j++)

            if (arr[j] < arr[min_idx])

           min_idx = j;

        swap(&arr[min_idx], &arr[i]);

  }}
```

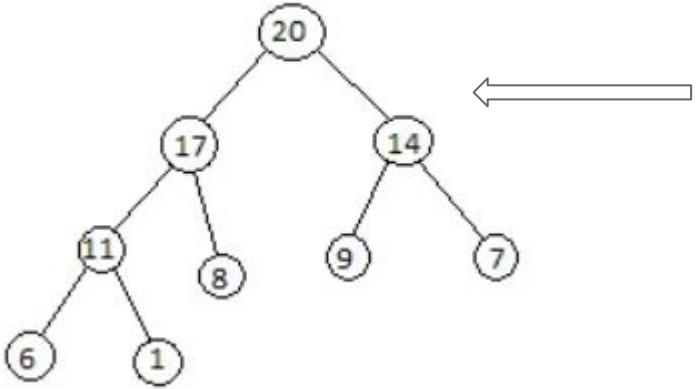


Short note about few other sorting algorithms

Merge Sort: Uses concept of “Divide and Concur” . In this sorting algorithm, The array is divided 2 halves repeatedly till every sub-array has minimal number of elements in it. Every two consecutive sub-arrays are compared based on given criteria and are reordered according to condition and those sub-arrays are further merged to form final resultant array.

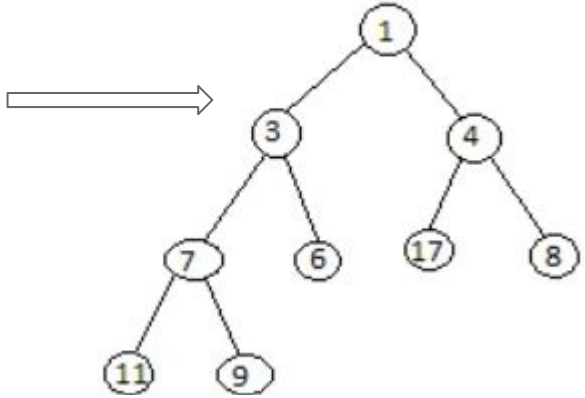


Heap Sort: Here array elements are stored in the form of binary trees where the elements are stored according to the criteria for the sort.



In this example, the elements are arranged in descending order in heap(binary tree).
Where the largest of all (20) will be placed at top of the heap and next largest are placed in next level (17,14) and the smallest are placed at leaves of tree.
This type of tree is called **Max heap**.

In this example, the elements are arranged in ascending order in heap(binary tree).
Where the smallest of all(1) is placed at the root position and next smallest elements(3,4) in the next level of binary tree. Big elements are placed at leaf nodes of binary tree.
This type of binary tree is called **Min heap**.



Resources

Arrays:

1. https://www.geeksforgeeks.org/introduction-to-arrays/
2. https://www.hackerearth.com/practice/data-structures/arrays/1-d/tutorial/

2D Arrays:

1. https://www.hackerearth.com/practice/data-structures/arrays/multi-dimensional/tutorial/
2. https://www.javatpoint.com/data-structure-2d-array
3. https://beginnersbook.com/2014/01/2d-arrays-in-c-example/

Sorting:

1. https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/tutorial/ (You can find all sorting algorithms in side navigation bar)
2. https://www.interviewbit.com/tutorial/sorting-algorithms/ (You can find all sorting algorithms in side navigation bar)
3. https://www.topcoder.com/thrive/articles/Sorting

Assess Yourself

After reading the articles from previous slides, try to work on following:

1. https://www.sanfoundry.com/data-structure-questions-answers-array-array-operations/
2. https://www.careerride.com/view/mcqs-on-sorting-with-answers-19637.aspx
3. https://www.geeksforgeeks.org/algorithms-gq/searching-and-sorting-gq/
4. https://www.geeksforgeeks.org/data-structure-gq/array-gq/
5. https://www.hackerearth.com/practice/data-structures/arrays/1-d/practice-problems/?sort_by=unsolved&p_level=easy
6. https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=arrays
7. https://www.hackerrank.com/domains/algorithms?filters%5Bsubdomains%5D%5B%5D=arrays-and-sorting

Try implementing few sorting algorithms and work on few problems on arrays to have practical knowledge and more understanding.

THANK YOU