Tutorial Link https://codequotient.com/tutorials/Bubble Sort/5a12e7f046765b2b63e34748

TUTORIAL

Bubble Sort

Chapter

Bubble Sort is the simplest comparison based sorting algorithm. It is in-place and needs no extra memory.

Idea:

This algorithm repeatedly goes through the array, compares adjacent elements and swaps them if they are in the wrong order. This is not an efficient sorting algorithm, but due to its simplicity, it is often introduced to the students for understanding the foundations of sorting.

Algorithm - Sort in ascending order

- Repeatedly traverse the array, and in each pass compare the adjacent elements. If the order of adjacent elements is wrong (*i.e. arr[j] > arr[j+1]*), then swap them.
- Suppose the length of the array is **n**. Now to sort the array we need to do at max **n-1** passes(traversals) on it.

- After the first pass, the largest element will move to the last index (*i.e. index n-1*).
- Similarly, after the second pass, the second largest element will move to the second last index (*i.e. index n-2*).
- After the **i**th pass, the **i**th largest element will move to the *index n-i*. That means after the **i**th pass, the last **i** elements in the array will be at their correct positions. Therefore in the next pass, we need to check the adjacent elements only till the **n-i-1** index.
- Finally, after doing all the passes the given array will be sorted.

Let's visualise the algorithm with the following example:

```
arr[ ] = {6, 3, 8, 9, 5}
```

First Pass :

{**6**, **3**, 8, 9, 5}

compare arr[0] with arr[1] : a*rr[0] > arr[1]* so we'll swap them, and the array will become {3, 6, 8, 9, 5}

{3, **6**, **8**, 9, 5}

compare arr[1] with arr[2] : a*rr[1] < arr[2]* so no swapping will happen, and the array will remain same {3, 6, 8, 9, 5}

{3, 6, **8**, **9**, 5}

compare arr[2] with arr[3] : a*rr[2] < arr[3]* so no swapping will happen, and the array will remain same {3, 6, 8, 9, 5}

{3, 6, 8, **9**, **5**}

compare arr[3] with arr[4] : a*rr[3] > arr[4]* so we'll swap them, and the array will become {3, 6, 8, 5, **9**}

After the first pass, the largest element comes to the last index.

Second Pass :

{**3**, **6**, 8, 5, 9}

compare arr[0] with arr[1] : a*rr[0] < arr[1]* so no swapping will happen, and the array will remain same {3, 6, 8, 5, 9}

{3, **6**, **8**, 5, 9}

compare arr[1] with arr[2] : a*rr[1] < arr[2]* so no swapping will happen, and the array will remain same {3, 6, 8, 5, 9}

{3, 6, **8**, **5**, 9}

compare arr[2] with arr[3] : a*rr[2] > arr[3]* so we'll swap them, and the array will become {3, 6, 5, **8**, **9**}

After the second pass, the second largest element has come to its correct position.

Third Pass :

{**3**, **6**, 5, 8, 9}

compare arr[0] with arr[1] : a*rr[0] < arr[1]* so no swapping will happen, and the array will remain same {3, 6, 5, 8, 9}

{3, **6**, **5**, 8, 9}

compare arr[1] with arr[2] : a*rr[1] > arr[2]* so we'll swap them, and the array will become {3, 5, **6**, **8**, **9**}

After the third pass, the third largest element has come to the third last position. Also we can notice that the last 3 elements have occupied their correct positions after the third pass.

Fourth Pass :

{**3**, **5**, 6, 8, 9}

compare arr[0] with arr[1] : a*rr[0] < arr[1]* so no swapping will happen, and the array will remain same {3, **5**, **6**, **8**, **9**}

No need to check for the remaining elements on the right, as they are already at their correct positions.

After doing all the passes, we can clearly observe that the given array is sorted in ascending order.

Pseudo Code

```
for i := 0 to n-2 do
   for j := 0 to n-i-2 do
       if arr[j] > A[j+1]
           swap(A[j], A[j+1])
       endif
   end
end
```

Javascript

```javascript

function bubbleSort(arr,n){
  for (let i = 0; i < n-1; i++){
    // last i elements are already at the correct position
    for (let j = 0; j < n-i-1; j++){
      if (arr[j] > arr[j+1]){
        // swap arr[j], arr[j+1]
        let temp = arr[j]
        arr[j] = arr[j+1]
        arr[j+1] = temp
      }
    }
  }
}

function printArray(arr){
  console.log(arr.join(' '))
}

function main(){
  let arr = [6, 3, 8, 9, 5]
  let n = arr.length

  console.log("Given Array: ")
  printArray(arr)

  bubbleSort(arr, n)

  console.log("Sorted Array: ")
  printArray(arr, n)
}

main()
```

C

```c
#include <stdio.h>

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

void bubbleSort(int arr[], int n)
{
   for (int i = 0; i < n-1; i++)
   {
        // last i elements are already at the correct
position
       for (int j = 0; j < n-i-1; j++)
       {
           if (arr[j] > arr[j+1])
               swap(&arr[j], &arr[j+1]);
       }
   }
}

void printArray(int arr[], int n)
{
   for (int i = 0; i < n; i++)
```

```
        printf("%d ", arr[i]);
    printf("\n");
}

int main()
{
    int arr[] = {6, 3, 8, 9, 5};
    int n = sizeof(arr)/sizeof(arr[0]);

    printf("Given Array: ");
    printArray(arr, n);

    bubbleSort(arr, n);

    printf("Sorted Array: ");
    printArray(arr, n);

    return 0;
}

```

Java

```
public class Main
{
    static void bubbleSort(int arr[], int n)
    {
        for (int i = 0; i < n-1; i++)
        {
            // last i elements are already at the correct position
            for (int j = 0; j < n-i-1; j++)
            {
                if (arr[j] > arr[j+1])
                {
                    // swap arr[j], arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
                }
```

```java
            }
        }
    }

    static void printArray(int arr[], int n)
    {
        for (int i = 0; i < n; i++)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    public static void main(String args[])
    {
        int arr[] = {6, 3, 8, 9, 5};
        int n = arr.length;

        System.out.print("Given Array: ");
        printArray(arr, n);

        bubbleSort(arr, n);

        System.out.print("Sorted Array: ");
        printArray(arr, n);
    }
}

```

Python 3

```python

def bubbleSort(arr,n):
    for i in range(n-1):
        # last i elements are already at the correct position
        for j in range(n-i-1):
            if(arr[j] > arr[j+1]):
                # Swapping elements
                arr[j],arr[j+1] = arr[j+1],arr[j]


```

```python
def printArray(arr):
    print(' '.join(str(x) for x in arr))

if __name__ == '__main__':
    arr = [6, 3, 8, 9, 5]
    n = len(arr)
    print('Given Array:')
    printArray(arr)

    bubbleSort(arr,n)

    print('Sorted Array:')
    printArray(arr)
```

C++

```cpp
#include <iostream>
using namespace std;

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

void bubbleSort(int arr[], int n)
{
   for (int i = 0; i < n-1; i++)
   {
        // last i elements are already at the correct position
       for (int j = 0; j < n-i-1; j++)
       {
           if (arr[j] > arr[j+1])
               swap(&arr[j], &arr[j+1]);
       }
   }
}

```

```
void printArray(int arr[], int n)
{
   for (int i = 0; i < n; i++)
       cout << arr[i] << " ";
   cout << "\n";
}

int main()
{
   int arr[] = {6, 3, 8, 9, 5};
   int n = sizeof(arr)/sizeof(arr[0]);
   cout<<"Given Array: ";
   printArray(arr, n);

   bubbleSort(arr, n);

   cout<<"Sorted Array: ";
   printArray(arr, n);

   return 0;
}

```

Optimising Above Implementation

In the above implementation, we have to do n-1 passes on the array even if the array was already sorted in some **i**th pass. It can be optimised by stopping the algorithm if no swapping happened at all in the current pass.

Pseudo Code

```
for i := 0 to n-2 do
    swapped = false;
    for j := 0 to n-i-2 do
        if arr[j] > A[j+1]
            swap(A[j], A[j+1])
            swapped = true;
        endif
    end
    if swapped == false
        break;
end
```

Javascript

```javascript

function bubbleSort(arr,n){
  for (let i = 0; i < n-1; i++){
    let swapped = false
    // last i elements are already at the correct
position
    for (let j = 0; j < n-i-1; j++){
      if (arr[j] > arr[j+1]){
        // swap arr[j], arr[j+1]
        let temp = arr[j]
        arr[j] = arr[j+1]
        arr[j+1] = temp
        swapped = true
      }
    }
    // If no swapping happened in the current pass, then
break
    if(!swapped){
        break;
    }
  }
}

function printArray(arr){
  console.log(arr.join(' '))
}

function main(){
  let arr = [6, 3, 8, 9, 5]
  let n = arr.length

  console.log("Given Array: ")
  printArray(arr)

  bubbleSort(arr, n)

  console.log("Sorted Array: ")
  printArray(arr, n)
}

main()
```

C

```c
#include <stdio.h>

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}

void bubbleSort(int arr[], int n)
{
   for (int i = 0; i < n-1; i++)
   {
       int swapped = 0;
       // last i elements are already at the correct
position
       for (int j = 0; j < n-i-1; j++)
       {
           if (arr[j] > arr[j+1])
           {
               swap(&arr[j], &arr[j+1]);
               swapped = 1;
           }
       }
       // If no swapping happened in the current pass,
then break
       if (swapped == 0)
           break;
   }
}
```

```c

void printArray(int arr[], int n)
{
   for (int i = 0; i < n; i++)
       printf("%d ", arr[i]);
   printf("\n");
}

int main()
{
   int arr[] = {6, 3, 8, 9, 5};
   int n = sizeof(arr)/sizeof(arr[0]);

   printf("Given Array: ");
   printArray(arr, n);

   bubbleSort(arr, n);

   printf("Sorted Array: ");
   printArray(arr, n);

   return 0;
}

```

Java

```java
public class Main
{
   static void bubbleSort(int arr[], int n)
   {
       for (int i = 0; i < n-1; i++)
       {
           Boolean swapped = false;
           // last i elements are already at the correct
position
           for (int j = 0; j < n-i-1; j++)
           {
               if (arr[j] > arr[j+1])
               {
                   // swap arr[j], arr[j+1]
                   int temp = arr[j];
                   arr[j] = arr[j+1];
                   arr[j+1] = temp;

                   swapped = true;
               }
           }
           // If no swapping happened in the current pass, then break
           if (swapped == false)
               break;
       }
   }

   static void printArray(int arr[], int n)
   {
       for (int i = 0; i < n; i++)
           System.out.print(arr[i] + " ");
       System.out.println();
   }

   public static void main(String args[])
   {
       int arr[] = {6, 3, 8, 9, 5};
       int n = arr.length;

       System.out.print("Given Array: ");
       printArray(arr, n);

       bubbleSort(arr, n);

       System.out.print("Sorted Array: ");
       printArray(arr, n);
   }
}

```

Python 3

```python
def bubbleSort(arr,n):
    for i in range(n-1):
        swapped = False
        # last i elements are already at the correct position
        for j in range(n-i-1):
            if(arr[j] > arr[j+1]):
                # Swapping elements
                arr[j],arr[j+1] = arr[j+1],arr[j]
                swapped = True
        # If no swapping happened in the current pass, then break
        if not swapped:
            break


def printArray(arr):
    print(' '.join(str(x) for x in arr))

if __name__ == '__main__':
    arr = [6, 3, 8, 9, 5]
    n = len(arr)
    print('Given Array:')
    printArray(arr)

    bubbleSort(arr,n)

    print('Sorted Array:')
    printArray(arr)
```

C++

```cpp
#include <iostream>
using namespace std;

void swap(int *a, int *b)
{
   int temp = *a;
   *a = *b;
   *b = temp;
}
```

```cpp

void bubbleSort(int arr[], int n)
{
   for (int i = 0; i < n-1; i++)
   {
       bool swapped = false;
       // last i elements are already at the correct position
       for (int j = 0; j < n-i-1; j++)
       {
           if (arr[j] > arr[j+1])
           {
               swap(&arr[j], &arr[j+1]);
               swapped = true;
           }
       }
       // If no swapping happened in the current pass, then break
       if (swapped == false)
           break;
   }
}

void printArray(int arr[], int n)
{
   for (int i = 0; i < n; i++)
       cout << arr[i] << " ";
   cout << "\n";
}

int main()
{
   int arr[] = {6, 3, 8, 9, 5};
   int n = sizeof(arr)/sizeof(arr[0]);

   cout<<"Given Array: ";
   printArray(arr, n);

```

```
    bubbleSort(arr, n);

    cout<<"Sorted Array: ";
    printArray(arr, n);

    return 0;
}

```

Properties of Bubble Sort:

Worst and Average Case Time Complexity: O(n^2) ; *Worst case occurs when the array is sorted in opposite direction*

Best Case Time Complexity: O(n) ; *Best case occurs when the given array is already sorted*

Space Complexity: O(1)

In-Place Sorting Algorithm: Yes

Stable Sorting Algorithm: Yes

Video Solution

<iframe width="560" height="315" src="https://www.youtube.com/embed/K2YfGoP9Kw4" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>