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

TUTORIAL

Bubble Sort

Chapter

Introduction to Bubble Sort

This algorithm is based on the idea of repeatedly comparing pairs of adjacent elements and then switching their positions if they exist in the wrong order. It will repeat this procedure multiple times to ensure that every element is at its proper position. For example, the following array has 5 elements as below: -

```
15   11   14   12   18
```

To sort this array in ascending order, bubble sort will perform following steps called passes: -

First Pass:

```
11    15    14    12    18, compares the first two elements, and
swaps since 15 > 11.
11    14    15    12    18, Next two elements and Swap since 15 > 14
11    14    12    15    18, Next two elements and Swap since 15 > 12
```

```
11    14    12    15    18, Next two elements are already in order
(18 > 15), algorithm does not swap them.
```

After first pass in ascending order, the largest element of the array will be in the last position. So that next time we have to repeat this comparison till the last element only. In each pass at least one element will be placed at its proper position in the array.

Second Pass:

```
11    14    12    15    18,          No swap as 11 and 14 are in
order
11    12    14    15    18,          Swap since 14 > 12
11    12    14    15    18,          No swap as 14 and 15 are in
order
11    12    14    15    18,          No swap as 15 and 18 are in
order.
```

Now, after the second pass, the array is already sorted, but the algorithm will run for n-1 passes to check all possibilities. We can implement the algorithm to stop at this position. But in general, Bubble sort needs one whole pass without any swap to know it is sorted. Following algorithm will explain the bubble sort procedure: -

```
for i := 1 to n do
   for j := 1 to n do
       if A[j].key > A[j+1].key
           swap(A[j], A[j+1])
       endif
   end
end
```

```c
#include<stdio.h>

```

```
3  void printArray(int array[], int size)
4  {
5    int i;
6    for (i=0; i < size; i++)
7      printf("%d ", array[i]);
8    printf("\n");
9  }
10
11 void bubbleSort(int array[], int n)
12 {
13   int i, j, temp;
14   for (i = 0; i < n-1; i++)
15   {
16     for (j = 0; j < n-1; j++)
17       if (array[j] > array[j+1])
18       {
19         temp = array[j];
20         array[j] = array[j+1];
21         array[j+1] = temp;
22         printf("In Pass - %d\n",i+1);
23         printf("%d & %d are
   swapped\n",array[j+1],array[j]);
24         printArray(array, n);   // During Sorting
25       }
26     printf("Array after Pass - %d\n",i+1);
27     printArray(array, n);   // During Sorting
28   }
29 }
30
31 int main()
32 {
33   int array[] = {15, 11, 14, 12, 18};
34   int n = 5;
35   /* we can also calculate the number of elements in an
   array by using sizeof(array)/sizeof(array[0]).        */
36   printf("Un-Sorted array: \n");
37   printArray(array, n);     // Unsorted array
38   bubbleSort(array, n);     // Call the sorting routine
     printf("\nSorted array: \n");
```

```

  printArray(array, n);   // Sorted array
  return 0;
}

```

Java

```
import java.util.Scanner;
// Other imports go here
// Do NOT change the class name
class Main{
    static void printArray(int array[], int size){
        int i;
        for (i=0; i < size; i++)
            System.out.printf("%d ", array[i]);
        System.out.printf("\n");
    }

    static void bubbleSort(int array[], int n){
        int i, j, temp;
        for (i = 0; i < n-1; i++){
            for (j = 0; j < n-1; j++)
                if (array[j] > array[j+1]){
                    temp = array[j];
                    array[j] = array[j+1];
                    array[j+1] = temp;
                    System.out.printf("In Pass - %d\n",i+1);
                    System.out.printf("%d & %d are swapped\n",array[j+1],array[j]);
                    printArray(array, n);   // During Sorting
                }
            System.out.printf("Array after Pass - %d\n",i+1);
            printArray(array, n);   // During Sorting
        }
    }

    public static void main(String[] args){
        int array[] = {15, 11, 14, 12, 18};
```

```
        int n = 5;
        /* we can also calculate the number of elements in an array by using sizeof(array)/sizeof(array[0]).*/
        System.out.printf("Un-Sorted array: \n");
        printArray(array, n);     // Unsorted array
        bubbleSort(array, n);     // Call the sorting routine
        System.out.printf("\nSorted array: \n");
        printArray(array, n);   // Sorted array
    }
}
```

Python 3

```
def printArray(A,size):
    for i in range(size):
        print(A[i],end=' ');
    print()

def bubbleSort(arr,n):
    for i in range(n-1):
        for j in range(0, n-1):
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]
                print('In Pass-',str(i+1));
                print(str(arr[j+1]),' & ',str(arr[j]),'are swapped');
                printArray(arr,n);
        print('Array after Pass-',i+1);
        printArray(arr,n);
    print();


if __name__=="__main__":
    A = [15,11,14,12,18]
    print('Unsorted Array:')
    printArray(A,len(A));
    print()


    bubbleSort(A,len(A));
```

```

    print('Sorted Array');
    printArray(A,len(A));
```

C++

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;

void printArray(int array[], int size){
    int i;
    for (i=0; i < size; i++)
        cout<<array[i]<<' ';
    cout<<endl;
}

void bubbleSort(int array[], int n){
    int i, j, temp;
    for (i = 0; i < n-1; i++){
        for (j = 0; j < n-1; j++)
            if (array[j] > array[j+1]){
                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
                cout<<"In Pass - "<<i+1<<endl;
                cout<<array[j+1] << " & " << array[j] <<
" are swapped "<<endl;
                printArray(array, n);   // During Sorting
            }
        cout<<"Array after Pass - "<<i+1<<endl;
        printArray(array, n);   // During Sorting
    }
}

int main(){
    int array[] = {15, 11, 14, 12, 18};
    int n = 5;
    /* we can also calculate the number of elements in an
array by using sizeof(array)/sizeof(array[0]).*/
```

```
34      cout<<"Un-Sorted array:"<<endl;
35      printArray(array, n);     // Unsorted array
36      bubbleSort(array, n);     // Call the sorting routine
37      cout<<"\nSorted array:"<<endl;
38      printArray(array, n);   // Sorted array
39      return 0;
40  }
```

```
Un-Sorted array:
15 11 14 12 18
In Pass - 1
15 & 11 are swapped
11 15 14 12 18
In Pass - 1
15 & 14 are swapped
11 14 15 12 18
In Pass - 1
15 & 12 are swapped
11 14 12 15 18
Array after Pass - 1
11 14 12 15 18
In Pass - 2
14 & 12 are swapped
11 12 14 15 18
Array after Pass - 2
11 12 14 15 18
Array after Pass - 3
11 12 14 15 18
Array after Pass - 4
11 12 14 15 18

Sorted array:
11 12 14 15 18
```

Complexity Analysis

The output above shows the swapping done in each pass and final array after each pass. The algorithm always perform n-1 passes, where n is the number of elements in the array. **The time complexity of this algorithm is O(n^2) as both loops in bubbleSort() function will run n times.**

Properties of Bubble Sort

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

Best Case Time Complexity: O(n). Best case occurs when array is already sorted.

Auxiliary Space: O(1)

Sorting In Place: Yes

Stable: Yes

It is popular for its capability to detect a very small error (like swap of just two elements) in almost-sorted arrays and fix it with just linear complexity (2n).

codequotient.com

Tutorial by codequotient.com | All rights reserved, CodeQuotient 2020