COMPARISION OF BUBBLE AND INSERTION SORT

Submitted by

Naga Sindhu [RA2111003011836]
Anisha Kumari [RA2111003011837]

Under the Guidance of
DR. S VIDHYA
Assistant professor,

Department of Computing Technologies

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

APRIL 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

APRIL 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

BONAFIDE CERTIFICATE

Certified that this course project report titled " comparision of bubble and insertion sort " is the bonafide workdone by **Naga Sindhu [RA2111003011836], Anisha Kumari [RA2111003011837]** who carried out under my supervision.certified further,that to the best of my knowledge the work reported here in does not form part of any other work.

SIGNATURE	**SIGNATURE**
Faculty- in-Charge	**Head of the Department**
Dr. S. Vidhya	**Dr. M. Pushpalatha**
Assistant Professor	**Professor and Head**
Department of Computing Technologies	**Department of Computing Technology**
SRMIST – KTR.	**SRMIST – KTR.**
Date :	

1. INTRODUCTION

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats.

ALL SORTING TECHNIQUES ARE–

- Bubble Sort
- Bucket Sort
- Comb Sort
- Counting Sort
- Cycle Sort
- Heap Sort
- Insertion Sort
- Merge Sort
- Pigeonhole Sort
- Quick Sort
- Radix Sort
- Selection Sort
- Shell Sort

2. COMPARISION OF INSERTION SORT AND BUBBLE SORT

Bubble Sort and Insertion Sort are simple sorting algorithms that are commonly used to sort small datasets or as building blocks for more complex sorting algorithms. Here's a comparison of the two algorithms

3.BUBBLE SORT

It is the simplest Sorting technique that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

4.1 ALGORITHM

We know that to sort a list of n elements using bubble sort, we need to perform n – 1 iterations. And for each iteration, we need to:

1. Run a loop over the entire list or array.
2. Compare the element at the index i with the element at i + 1.
3. If the element at i is greater than the element at i + 1, swap both the elements
4. Else, move to the next element.

4.2 PSEUDO CODE

```
procedure bubbleSort( list : array of items )

  loop = list.count;

  for i = 0 to loop-1 do:
    swapped = false

    for j = 0 to loop-1 do:

      /* compare the adjacent elements */
      if list[j] > list[j+1] then
        /* swap them */
        swap( list[j], list[j+1] )
        swapped = true
      end if

    end for

    /*if no number was swapped that means
    array is sorted now, break the loop.*/

    if(not swapped) then
      break
    end if

  end for

end procedure return list
```

4.3 CODE

```
// Bubble sort in C

#include <stdio.h>

// perform the bubble sort
void bubbleSort(int array[], int size) {

  // loop to access each array element
  for (int step = 0; step < size - 1; ++step) {

    // loop to compare array elements
    for (int i = 0; i< size - step - 1; ++i) {

      // compare two adjacent elements
      // change > to < to sort in descending order
      if (array[i] > array[i + 1]) {

        // swapping occurs if elements
        // are not in the intended order
        int temp = array[i];
        array[i] = array[i + 1];
        array[i + 1] = temp;
      }
    }
  }
}

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

int main() {
  int data[] = {-2, 45, 0, 11, -9};

  // find the array's length
  int size = sizeof(data) / sizeof(data[0]);

bubbleSort(data, size);

printf("Sorted Array in Ascending Order:\n");
printArray(data, size);
}
```

4.4 SAMPLE INPUT :

{-2, 45 , 0 , 11 , - 9}

SAMPLE OUTPUT :

{-9 , -2 , 0 , 11 , 45}

4.5 PROBLEM SOLVING

Initial List : 5 3 7 1 6
N = 5

PASS 1 :

5 3 7 1 6

5 >3 ,so swap

3 5 7 1 6

5< 7 , so no swap

3 5 7 1 6

7 > 1 ,so swap

3 5 1 7 6

7 > 6, so swap

3 5 1 6 7

PASS 2 :

3 5 1 6 7

3 < 5 , no swap

3 5 1 6 7

5 > 1 , so swap

3 1 5 6 7
5 < 6 , so no swap

3 1 5 6 7

6 < 7 , no swap

3 1 5 6 7

PASS 3 :

3 1 5 6 7

3 > 1 , so swap

1 3 5 6 7

PASS 4 :

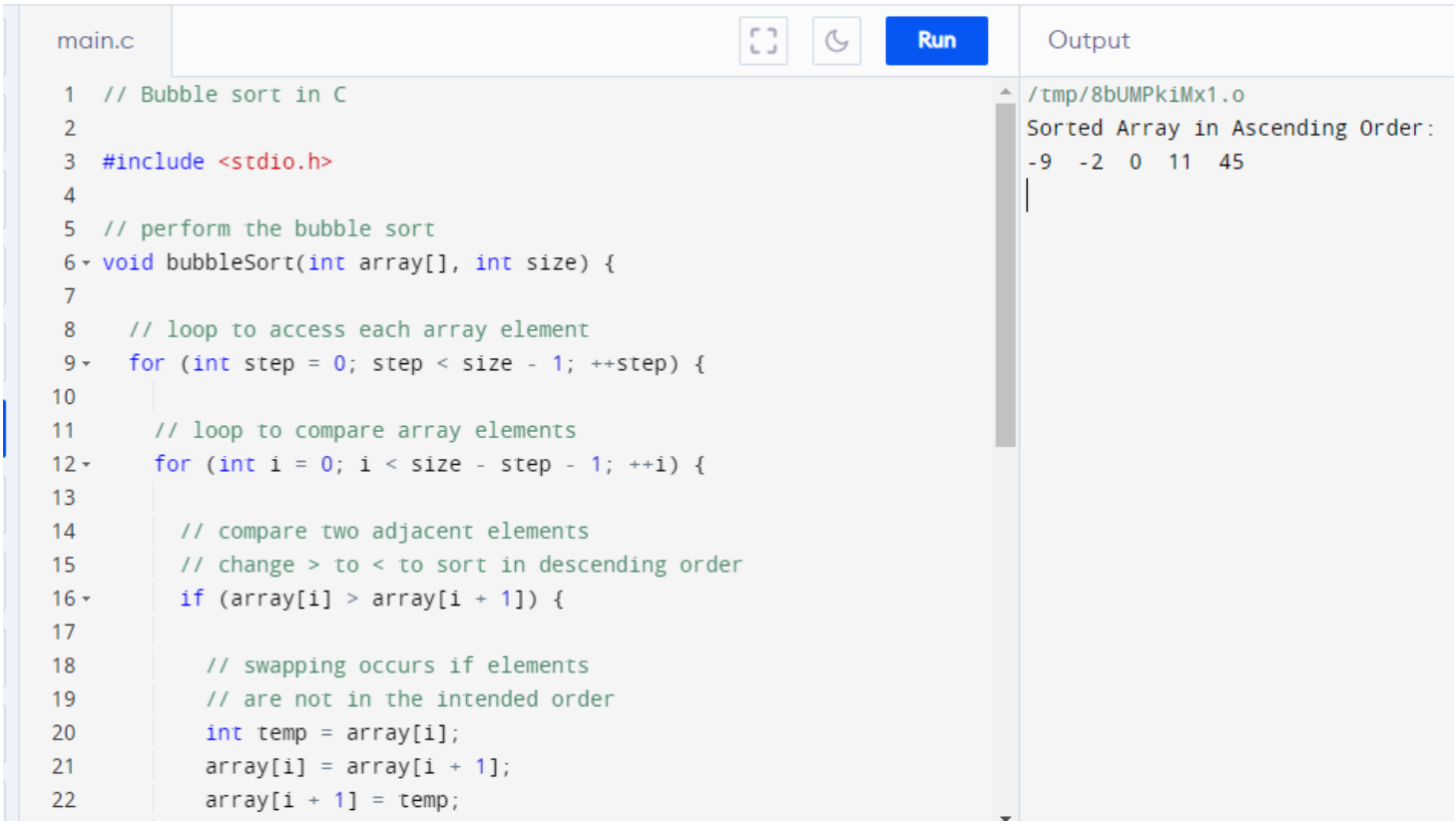
1 3 5 6 7

1 < 3 , no swap

1 3 5 6 7

Final output : 1 3 5 6 7

4.6 IMPLEMENTATION




```c
// Bubble sort in C

#include <stdio.h>

// perform the bubble sort
void bubbleSort(int array[], int size) {

  // loop to access each array element
  for (int step = 0; step < size - 1; ++step) {

    // loop to compare array elements
    for (int i = 0; i < size - step - 1; ++i) {

      // compare two adjacent elements
      // change > to < to sort in descending order
      if (array[i] > array[i + 1]) {

        // swapping occurs if elements
        // are not in the intended order
        int temp = array[i];
        array[i] = array[i + 1];
        array[i + 1] = temp;
```

4.7 TIME COMPLEXITY

Worst Case

- In the worst-case scenario, the outer loop runs O(n) times.
- As a result, the worst-case time complexity of bubble sort is O(n x n) = O(n x n) (n2).

Best Case

- In the best-case scenario, the array is already sorted, but just in case, bubble sort performs O(n) comparisons.
- As a result, the time complexity of bubble sort in the best-case scenario is O(n).

Average Case

- Bubble sort may require (n/2) passes and O(n) comparisons for each pass in the average case.
- As a result, the average case time complexity of bubble sort is O(n/2 x n) = O(n/2 x n) = O(n/2 x n) = O(n/2 x n) = O (n2).

4.INSERTION SORT

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration. Insertion sort works similarly as we sort cards in our hand in a card game. We assume that the first card is already sorted then, we select an unsorted card.

5.1 ALGORITHM

Step 1 - If the element is the first element, assume that it is already sorted. Return 1.

Step2 - Pick the next element, and store it separately in a key

Step3 - Now, compare the key with all elements in the sorted array.

Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.

Step 5 - Insert the value.

Step 6 - Repeat until the array is sorted.

5.2 PSEUDO CODE

```
procedure insertionSort(A: list of sortable items)
   n = length(A)
   for i = 1 to n - 1 do
       j = i
```

```
        while j > 0 and A[j-1] > A[j] do
            swap(A[j], A[j-1])
            j = j - 1
        end while
    end for
end procedure
```

5.3 CODE

```
// Insertion sort in C

#include <stdio.h>

// Function to print an array
void printArray(int array[], int size) {
  for (int i = 0; i< size; i++) {
printf("%d ", array[i]);
  }
printf("\n");
}

void insertionSort(int array[], int size) {
  for (int step = 1; step < size; step++) {
    int key = array[step];
    int j = step - 1;

    // Compare key with each element on the left of it until an element smaller than
    // it is found.
    // For descending order, change key<array[j] to key>array[j].
    while (key < array[j] && j >= 0) {
array[j + 1] = array[j];
      --j;
    }
array[j + 1] = key;
  }
}

// Driver code
int main() {
  int data[] = {9, 5, 1, 4, 3};
  int size = sizeof(data) / sizeof(data[0]);
insertionSort(data, size);
printf("Sorted array in ascending order:\n");
printArray(data, size);
}
```

5.4 SAMPLE INPUT :

{9 , 5 , 1 , 4 , 3}

SAMPLE OUTPUT :

{1 , 3 , 4 , 5 , 9}

5.5 PROBLEM SOLVING

Initial List : 5 3 7 1 6
N = 5

PASS 1 :

5 3 7 1 6

5 >3 ,so swap

3 5 7 1 6

PASS : 2

3 5 7 1 6

5< 7 , so no swap

3 < 7 , no swap

3 5 7 1 6

PASS : 3

3 5 7 1 6

7 > 1 ,so swap

5 > 1 , so swap

3 > 1 , swap

1 3 5 7 6

PASS : 4

1 3 5 7 6

7 > 6 , swap

1 3 5 6 7

5 < 6 , no swap

3 < 6,no swap

1 < 6 , no swap

1 3 5 6 7

Final output : 1 3 5 6 7

5.6 IMPLEMENTATION

main.c

Run

Output

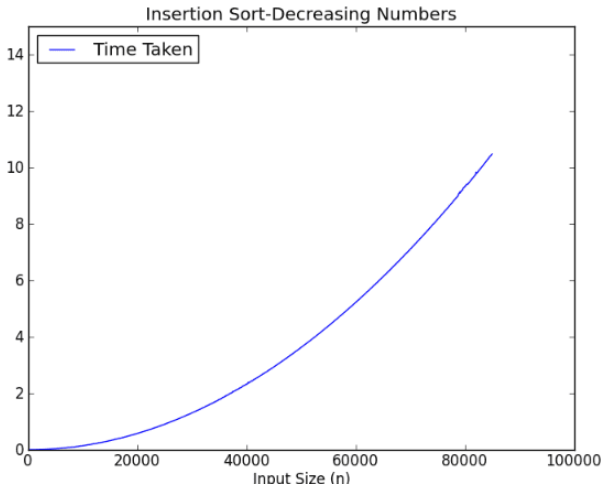
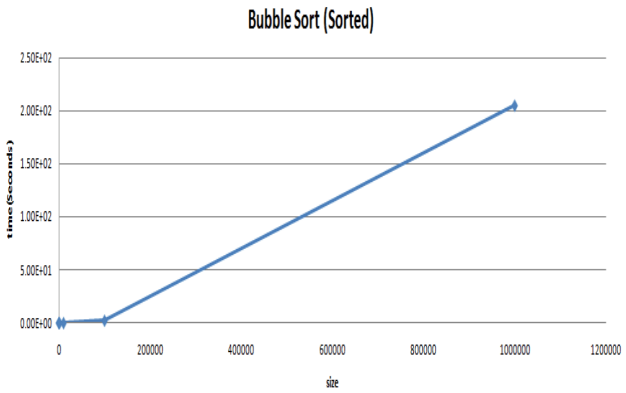
```c
// Insertion sort in C

#include <stdio.h>

// Function to print an array
void printArray(int array[], int size) {
  for (int i = 0; i < size; i++) {
    printf("%d ", array[i]);
  }
  printf("\n");
}

void insertionSort(int array[], int size) {
  for (int step = 1; step < size; step++) {
    int key = array[step];
    int j = step - 1;

    // Compare key with each element on the left of it until an
        element smaller than
    // it is found.
    // For descending order, change key<array[j] to key>array[j].
    while (key < array[j] && j >= 0) {
```

```
/tmp/6eCdI0HFgi.o
Sorted array in ascending order:
1 3 4 5 9
```

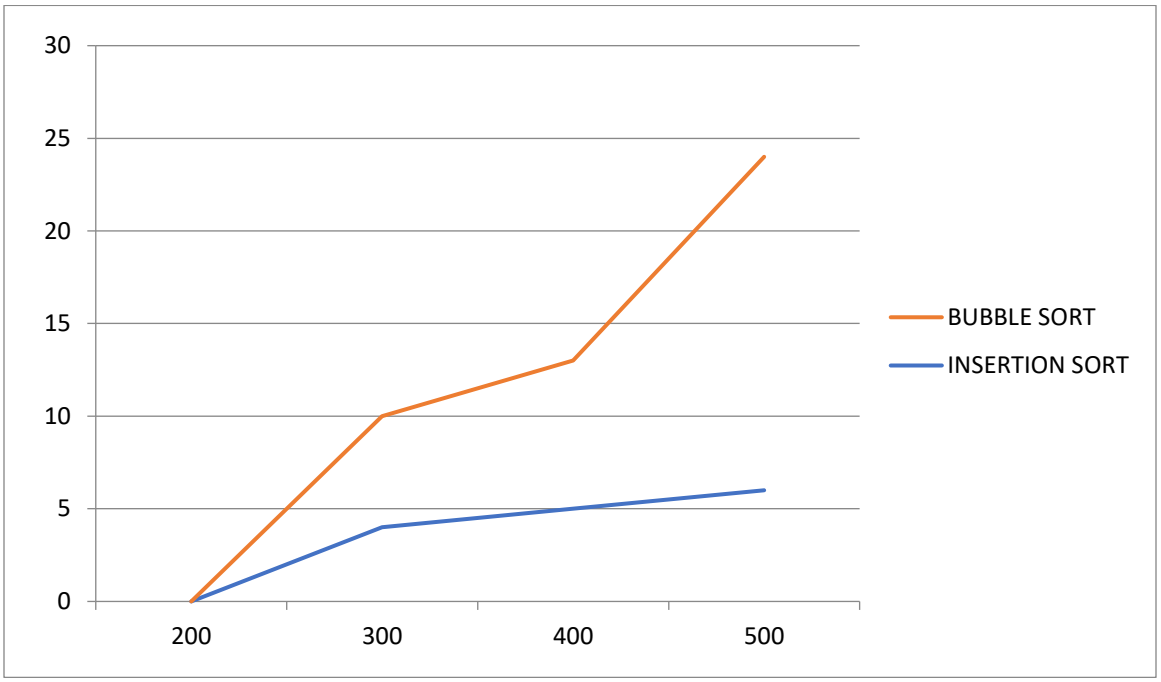
5.7TIME COMPLEXITY

- The worst case time complexity of Insertion sort is **O(N^2)**
- The average case time complexity of Insertion sort is **O(N^2)**
- The time complexity of the best case is **O(N)**.
- The space complexity is **O(1)**

COMPARISON OF INSERTION SORT AND BUBBLE SORT

INSERTION SORT	BUBBLE SORT
insertionSort(array)	**bubbleSort(array)**
for j <- lastSortedIndex down to 0	**for i<- 1 to indexOfLastUnsortedElement-1**
if current element j > X **move sorted element to the right by 1**	**if leftElement > rightElement** **swap leftElement and rightElement**
end insertionSort	**end bubbleSort**
	

7. INSERTION SORT IS BETTER THAN BUBBLE SORT



- **Computer Science comprises various data structures and algorithms that the user may implement for executing to complete a certain task or to solve any problem. Some classical algorithms are most familiar, such as bubble sort, insertion sort and merge sort, and others.**
- **While talking about Insertion sort, it is an easy sorting algorithm that functions in the same way as we sort the playing cards in our hands. Here, the array will be practically split into a sorted as well as an unsorted portion. After that, the values available from the unsorted portion will be picked and then positioned correctly in the sorted portion.**
- **Also, on the other side, the Bubble sort, which is also stated as comparison sort, is the easiest but quite ineffective type of algorithm for sorting, which goes through the list iterating, comparing.**

8. CONCLUSION

It can be concluded that bubble sort is an effortless way of sorting the elements of an array, thus having more time complexity. It is a stable and in-place algorithm which is most used for introducing the concept of sorting algorithms.

Insertion Sort works best with a small number of elements. The worst-case runtime complexity of Insertion Sort is O (n 2) O(n^2) O(n2) similar to that .of Bubble Sort. However, Insertion Sort is considered better than Bubble sort.

9. REFERENCES

1.https://youtu.be/TZRWRjq2CAg

2.https://byjus.com/gate/difference-between-bubble-sort-and-insertion-sort/

3.https://pediaa.com/what-is-the-difference-between-bubble-sort-and-insertion-sort/#:~:text=Bubble%20sort%20is%20a%20simple,one%20element%20at%20a%20time.

***