



Symbiosis Institute of Technology

Faculty of Engineering
CSE - Academic Year 2023-24
Data Structures Lab Batch 2022-26

Lab Assignment 2																				
Name	Manan Bhimjiyani																			
PRN no	22070122107																			
Batch	2022-26																			
Class	CS B1																			
Academic year & semester	2023-24																			
Date of submission	31/08/2023																			
Title of Assignment	Implement following sorting techniques and find the time complexity:: i. Bubble ii. Selection iii. Insertion																			
Theory	<div>A table comparing the best case, average case, and worst case time complexities of the given sorting algorithms: Bubble Sort, Insertion Sort, and Selection Sort.</div> <table><tr><th>Algorithm</th><th>Best Case</th><th>Avg Case</th><th>Worst Case</th></tr><tr><td>Bubble sort</td><td>$O(n)$</td><td>$O(n^2)$</td><td>$O(n^2)$</td></tr><tr><td>Insertion sort</td><td>$O(n)$</td><td>$O(n^2)$</td><td>$O(n^2)$</td></tr><tr><td>Selection sort</td><td>$O(n^2)$</td><td>$O(n^2)$</td><td>$O(n^2)$</td></tr></table> <div><p>Best case and Worst case time complexities of bubble sort, Insertion sort and selection sort.</p><p>Bubble Sort: Best Case: $O(n)$ - The array is already sorted, and no swaps are needed. Worst Case: $O(n^2)$ - The array is sorted in reverse order, and each element needs to be swapped with every other element.</p><p>Insertion Sort: Best Case: $O(n)$ - The array is already sorted, and each new element can be inserted in constant time. Worst Case: $O(n^2)$ - The array is sorted in reverse order, and each new element requires shifting all previous elements.</p></div>				Algorithm	Best Case	Avg Case	Worst Case	Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$	Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Algorithm	Best Case	Avg Case	Worst Case																	
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$																	
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$																	
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$																	

Selection Sort:

Best Case: $O(n^2)$ - The algorithm doesn't have any optimization for the initial ordering of the array.

Worst Case: $O(n^2)$ - Regardless of the input ordering, the algorithm requires n iterations for finding the minimum element in each pass.

Source Code:

Bubble Sort:

```
1  #include <stdio.h>
2
3  void bs(int arr[], int n) {
4      for (int i = 0; i < n - 1; i++) {
5          for (int j = 0; j < n - i - 1; j++) {
6              if (arr[j] > arr[j + 1]) {
7                  int temp = arr[j];
8                  arr[j] = arr[j + 1];
9                  arr[j + 1] = temp;
10             }
11         }
12     }
13 }
14
15 int main() {
16     int arr[] = {69,26,41,34,96,40};
17     int n = sizeof(arr) / sizeof(arr[0]);
18
19     printf("Original array: ");
20     for (int i = 0; i < n; i++) {
21         printf("%d ", arr[i]);
22     }
23
24     bs(arr, n);
25
26     printf("\nSorted array: ");
27     for (int i = 0; i < n; i++) {
28         printf("%d ", arr[i]);
29     }
30
31     return 0;
32 }
```

Selection Sort:

```

1  #include <stdio.h>
2  void ss(int arr[], int n) {
3      for (int i = 0; i < n - 1; i++) {
4          int mini = i;
5          for (int j = i + 1; j < n; j++) {
6              if (arr[j] < arr[mini]) {
7                  mini = j;
8              }
9          }
10         int temp = arr[i];
11         arr[i] = arr[mini];
12         arr[mini] = temp;
13     }
14 }
15 int main() {
16     int arr[] = {69,11,96,54,88};
17     int n = sizeof(arr) / sizeof(arr[0]);
18     printf("Original array: ");
19     for (int i = 0; i < n; i++) {
20         printf("%d ", arr[i]);
21     }
22     ss(arr, n);
23     printf("\nSorted array: ");
24     for (int i = 0; i < n; i++) {
25         printf("%d ", arr[i]);
26     }
27     return 0;
28 }
29

```

Insertion Sort:

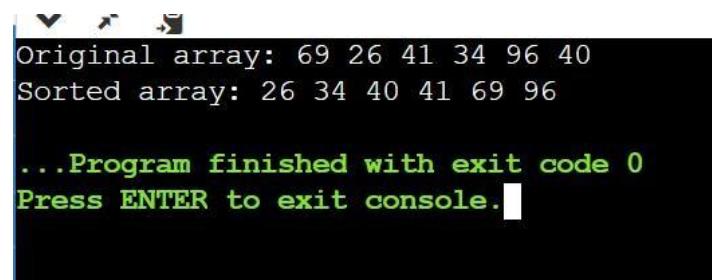
```

1  #include <stdio.h>
2  void is(int arr[], int n) {
3      for (int i = 1; i < n; i++) {
4          int key = arr[i];
5          int j = i - 1;
6          while (j >= 0 && arr[j] > key) {
7              arr[j + 1] = arr[j];
8              j = j - 1;
9          }
10         arr[j + 1] = key;
11     }
12 }
13 int main() {
14     int arr[] = {69,67,54,31,69};
15     int n = sizeof(arr) / sizeof(arr[0]);
16     printf("Original array: ");
17     for (int i = 0; i < n; i++) {
18         printf("%d ", arr[i]);
19     }
20     is(arr, n);
21     printf("\nSorted array: ");
22     for (int i = 0; i < n; i++) {
23         printf("%d ", arr[i]);
24     }
25     return 0;
26 }
27

```

Output :

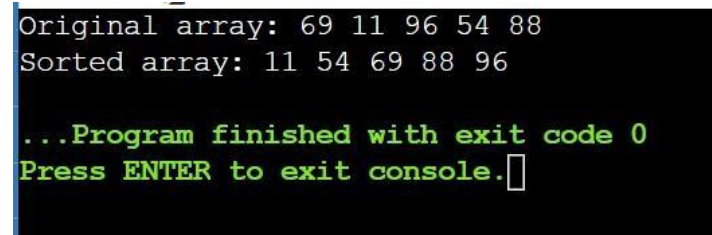
Bubble Sort:



```
Original array: 69 26 41 34 96 40
Sorted array: 26 34 40 41 69 96

...Program finished with exit code 0
Press ENTER to exit console.
```

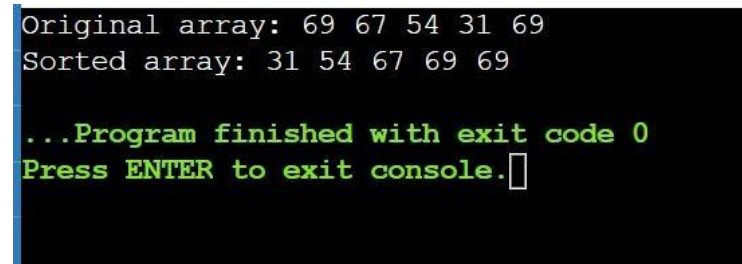
Selection Sort:



```
Original array: 69 11 96 54 88
Sorted array: 11 54 69 88 96

...Program finished with exit code 0
Press ENTER to exit console.
```

Insertion Sort:



```
Original array: 69 67 54 31 69
Sorted array: 31 54 67 69 69

...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion:

Thus, we have studied different sorting algorithms and their time complexities