


| | | |
|--|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |

Programming Language: - C++

Code :-


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

void Print_Array(vector<int> Array) {
    for (int i = 0; i < Array.size(); i++) {
        cout << Array[i] << " ";
    }
    cout << endl;
}

void Insertion_Sort(vector<int>& Array) {
    for (int i = 1; i < Array.size(); i++) {
        int key = Array[i];
        int j = i - 1;
        while (j >= 0 && Array[j] > key) {
            Array[j + 1] = Array[j];
            j--;
        }
        Array[j + 1] = key;
    }
}

int main() {
    vector<int> Array = {12 ,45, 57, 78, 89, 62, 7, 49, 21, 23};
    cout << "Array Before Sorting: - " << endl;
    Print_Array(Array);
    Insertion_Sort(Array);
    cout << "Array After Sorting :- " << endl;
    Print_Array(Array);

    return 0;
}
```

| | | |
|---|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |

Output :-

```
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1> cd "
d:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1\" ; if (
$?) { g++ Insertion_Sort.cpp -o Insertion_Sort } ; if ($?) { .\Insertion_Sort }
Array Before Sorting :-
12 45 57 78 89 62 7 49 21 23
Array After Sorting :-
7 12 21 23 45 49 57 62 78 89
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1>
```

Space Complexity:- _____

Justification: -

Time Complexity:

Best Case Time Complexity: _____

Justification: -

Worst Case Time Complexity:- _____

Justification: -

Bubble Sort


Theory: -

- Bubble sort is a sorting algorithm that compares two adjacent elements and swaps them until they are in the intended order.
- Just like the movement of air bubbles in the water that rise up to the surface, each array element moves to the end in each iteration. Therefore, it is called a bubble sort.

Working of Bubble Sort

1) First Iteration (Compare and Swap)

- 1) Starting from the first index, compare the first and the second elements.
- 2) If the first element is greater than the second element, they are swapped.
- 3) Now, compare the second and the third elements. Swap them if they are not in order.
- 4) The above process goes on until the last element


| | | |
|--|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |

```
void Swap(int& x, int& y) {
    int temp = x;
    x = y;
    y = temp;
}
```

```
void Bubble_Sort(vector<int> &Array) {
    int size = Array.size();
    for (int i = 0; i < size - 1; i++) {
        bool Swapped = false;
        for (int j = 0; j < size - i - 1; j++) {
            if (Array[j] > Array[j + 1]) {
                Swap(Array[j], Array[j + 1]);
                Swapped = true;
            }
        }
        if (Swapped == false) {
            break;
        }
    }
    return;
}
```

```
int main() {
    vector<int> Array = { 12 ,45 , 57 , 78 , 89 , 62 , 7 , 49 , 21 , 23 };
    cout << "Array Before Sorting :- " << endl;
    Print_Array(Array);
    Bubble_Sort(Array);
    cout << "Array After Sorting :- " << endl;
    Print_Array(Array);

    return 0;
}
```

| | | |
|---|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |

Output :-

```
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1> cd "
d:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1\" ; if (
$?) { g++ Bubble_Sort.cpp -o Bubble_Sort } ; if ($?) { .\Bubble_Sort }
Array Before Sorting :-
12 45 57 78 89 62 7 49 21 23
Array After Sorting :-
7 12 21 23 45 49 57 62 78 89
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1>
```

Space Complexity:- _____

Justification: -

Time Complexity:

Best Case Time Complexity: _____

Justification: -

Worst Case Time Complexity:- _____

Justification: -


Counting Sort

Theory: -

- Counting sort is a sorting algorithm that sorts the elements of an array by counting the number of occurrences of each unique element in the array.
- The count is stored in an auxiliary array and the sorting is done by mapping the count as an index of the auxiliary array.

Working of Counting Sort

- Find out the maximum element (let it be max) from the given array.
- Initialize an array of length max+1 with all elements 0. This array is used for storing the count of the elements in the array.
- Store the count of each element at their respective index in count array
- Store cumulative sum of the elements of the count array. It helps in placing the elements into the correct index of the sorted array.

| | | |
|--|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |


```
int Find_Max(vector<int> Array) {
    int max_num = Array[0];
    for(int i = 1; i < Array.size(); i++) {
        if(Array[i] > max_num) {
            max_num = Array[i];
        }
    }
    return max_num;
}
```

```
void Counting_Sort(vector<int>& Array) {
    int max_num = Find_Max(Array);
    vector<int> count(max_num + 1, 0);

    for (int i = 0; i < Array.size(); i++) {
        count[Array[i]]++;
    }

    int index = 0;
    for (int i = 0; i <= max_num; i++) {
        while (count[i] > 0) {
            Array[index++] = i;
            count[i]--;
        }
    }
    return;
}
```

```
int main() {
    vector<int> Array = { 12 ,45 , 57 , 78 , 89 , 62 , 7 , 49 , 21 , 23 };
    cout << "Array Before Sorting :- " << endl;
    Print_Array(Array);
    Counting_Sort(Array);
    cout << "Array After Sorting :- " << endl;
    Print_Array(Array);
    return 0;
}
```


| | | |
|---|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |

Output :-

```
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1> cd "
d:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1\" ; if (
$?) { g++ Counting_Sort.cpp -o Counting_Sort } ; if ($?) { .\Counting_Sort }
Array Before Sorting :-
12 45 57 78 89 62 7 49 21 23
Array After Sorting :-
7 12 21 23 45 49 57 62 78 89
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1>
```

Space Complexity:- _____

Justification: -

Time Complexity:

Best Case Time Complexity: _____

Justification: -

Worst Case Time Complexity:- _____

Justification: -


Selection Sort

Theory: -

- Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.
- The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.
-

Working of Selection Sort

- 1) Set the first element as a minimum.
- 2) Compare the minimum with the second element. If the second element is smaller than the minimum, assign the second element as the minimum. Compare minimum with the third element. Again, if the third element is smaller, then assign a minimum to the third element otherwise do nothing. The process goes on until the last element.
- 3) After each iteration, the minimum is placed in the front of the unsorted list.

| | | |
|--|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |

```
void Swap(int& x, int& y) {
    int temp = x;
    x = y;
    y = temp;
}
```

```
void Selection_Sort(vector<int>& Array) {

    for(int i = 0; i < Array.size(); i++) {
        int min_index = i;

        for (int j = i + 1; j < Array.size(); j++) {

            if (Array[j] < Array[min_index]) {
                min_index = j;
            }

        }

        Swap(Array[i], Array[min_index]);
    }

    return;
}
```


```
int main() {
    vector<int> Array = { 12 ,45 , 57 , 78 , 89 , 62 , 7 , 49 , 21 , 23 };

    cout << "Array Before Sorting :- " << endl;
    Print_Array(Array);

    Selection_Sort(Array);

    cout << "Array After Sorting :- " << endl;
    Print_Array(Array);

    return 0;
}
```

| | | |
|---|--|-----------------------------------|
|  Marwadi University | Marwari University Faculty of Technology Department of Information and Communication Technology | |
| Subject: Design and Analysis of Algorithms (01CT0512) | Aim: Implementing the Sorting Algorithms | |
| Experiment No: 01 | Date: 02-08-2024 | Enrollment No: 92200133030 |

Output :-

```
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1> cd "
d:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1\" ; if (
$?) { g++ Selection_Sort.cpp -o Selection_Sort } ; if ($?) { .\Selection_Sort }
Array Before Sorting :-
12 45 57 78 89 62 7 49 21 23
Array After Sorting :-
7 12 21 23 45 49 57 62 78 89
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 1>
```

Space Complexity:- _____

Justification: -

Time Complexity:

Best Case Time Complexity: _____

Justification: -

Worst Case Time Complexity:- _____

Justification: -

Conclusion:-
