

Implementation and Analysis of Counting Sort

```
def counting_sort(a):
    k = max(a) + 1
    c = [0] * k
    b = []
    for x in a:
        c[x] = c[x] + 1
    i = 0
    for x in range(k):
        for j in range(c[x]):
            a[i] = x
            i += 1
    return a

def display(a):
    print("Sorted Array: ")
    for i in range(len(a)):
        print(a[i], end=' ')

print("Enter an array")
a = list(map(int, input().split()))
counting_sort(a)
display(a)
```

Input:

10 20 30 50 40 60

Output:

10 20 30 40 50 60

Time Complexity:

- $O(n)$ in best case, worst case, average case

Program:

```
Counting Sort.py x
1 def counting_sort(a):
2     k = max(a) + 1
3     c = [0] * k
4     b = []
5     for x in a:
6         c[x] = c[x] + 1
7     i = 0
8     for x in range(k):
9         for j in range(c[x]):
10            a[i] = x
11            i += 1
12    return a
13
14
15 def display(a):
16     print("Sorted Array: ")
17     for i in range(len(a)):
18         print(a[i], end=' ')
19
20
21 print("Enter an array")
22 a = list(map(int, input().split()))
23 counting_sort(a)
24 display(a)
25
```

Input/Output:

```
Counting Sort
C:\Anaconda\envs\MLProject\python.exe "C:\MLProject\Counting Sort.py"
Enter an array
10 20 30 40 50 60
Sorted Array:
10 20 30 40 50 60
Process finished with exit code 0
```