**LAB REPORT**

**Submitted by**

**RUDRA SINHA [RA2011003010220]**

**CHINMAYI ANAND [RA2011003010232]**

**GOKUL T [RA201100301239]**

**Under the Guidance of**

Kishore Anthuvan Sahayaraj K

Computing **technologies**

**In partial satisfaction of the requirements for the degree of**

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE ENGINEERING**



**SCHOOL OF COMPUTING**

**COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR - 603203**

**JUNE 2022**



SRM INSTITUTION OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

**BONAFIDE CERTIFICATE**

Certified that this lab report titled **“Bucket Sort Problem”** is the bonafide work done by RUDRA SINHA(RA2011003010220),CHINMAYI ANAND(RA2011003010232), GOKUL T

(RA2011003010239) who carried out the lab exercises under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

**SIGNATURE**

**Content**

|  |  |
| --- | --- |
| **S. No** | **Topic** |
| 1. | Contribution Table |
| 2. | Problem Definition |
| 3. | Problem Explanation, Diagram and Example |
| 4. | Design Techniques Used |
| 5. | Algorithm for the problem |
| 6. | Explanation of Algorithm |
| 7. | Time Complexity Analysis |
| 8. | Conclusion |
| 9. | References |

# Contribution Table: -

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.no | Name | Reg. No. | Contribution |
| 1 | Rudra Sinha | RA2011003010220 | Problem Definition  , Problem Explanation with diagram and example,  Design Techniques used (Divide and Conquer), Algorithm for the problem, Explanation of algorithm with example, Complexity analysis, Conclusion, References. |
| 2 | Chinmayi Anand | RA2011003010232 | Cover Page, Content  ,Contribution table, Design Techniques used(Divide and Conquer), Explanation of algorithm with dry run testing, Team Management. |
| 3 | Gokul T | RA2011003010239 | Page formatting, Explanation of  algorithm with example, Complexity analysis, designing  the report. |

1. **Problem definition: -**

**Bucket sort**, or **bin sort**, is a [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm) that works by distributing the elements of an [array](https://en.wikipedia.org/wiki/Array_data_structure) into a number

of [buckets](https://en.wikipedia.org/wiki/Bucket_(computing)). Each bucket is then sorted individually, either

using a different sorting algorithm, or by recursively applying the bucket sorting algorithm. It is a [distribution](https://en.wikipedia.org/wiki/Distribution_sort) [sort](https://en.wikipedia.org/wiki/Distribution_sort), a generalization of [pigeonhole sort](https://en.wikipedia.org/wiki/Pigeonhole_sort) that allows multiple keys per bucket, and is a cousin of [radix sort](https://en.wikipedia.org/wiki/Radix_sort) in the most-to- least significant digit flavor. Bucket sort can be implemented with comparisons and therefore can also be considered

a [comparison sort](https://en.wikipedia.org/wiki/Comparison_sort) algorithm. The [computational](https://en.wikipedia.org/wiki/Analysis_of_algorithms) [complexity](https://en.wikipedia.org/wiki/Analysis_of_algorithms) depends on the algorithm used to sort each bucket, the number of buckets to use, and whether the input is uniformly distributed.

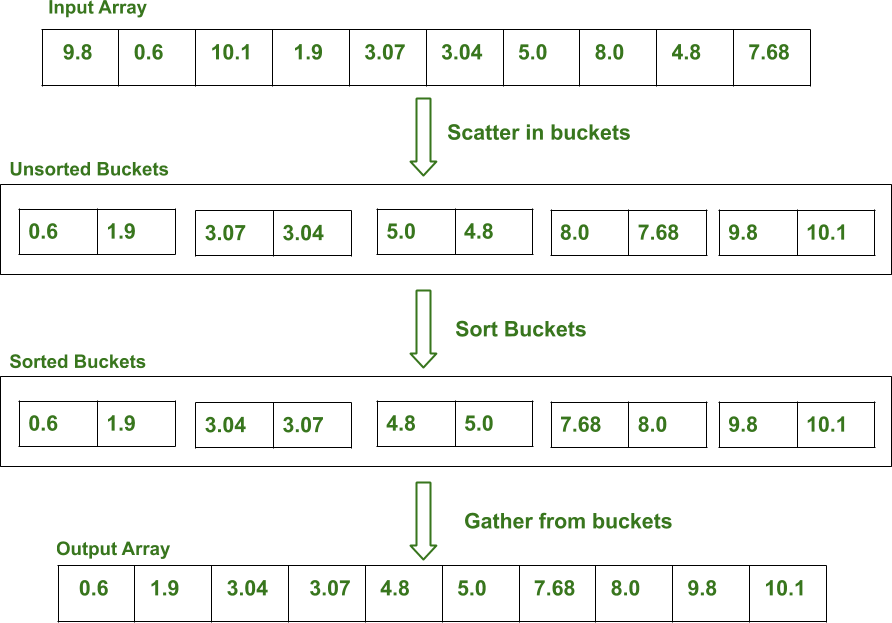
Given an array of decimal numbers, these numbers are either integers or floating-point numbers in both positive and negative. Negative numbers are converted into positive numbers and then sorted through bucket sort algorithm separately.

In bucket sorting, given array's numbers are sent to different buckets These buckets can be filled using linked list.

After segregating them into buckets, each bucket is sorted using a different algorithm, it can be either insertion sort or bubble sort or recursion.

Then all the elements or numbers from these linked lists are gathered and replaced in given array.

# Problem Explanation with diagram and example: -



Let the example be

arr[n]=[9.8,0.6,-10.1,1.9,-3.07,3.04,-5.0,8.0,4.8,7.68]

for i= to n-1:

diff=(arr[i]-min)/range - int(a[i]-min/range) if diff ==0 and arr[i]!=min:

then push arr[i] to b[int((arr[i]-min)/range)-1] else:

push arr[i] to b[int((arr[i]-min)/range)] Then sort each b[i]

Note:

For i from 0 to n-1 b[i] is a linked list. maximum element of arr[n]is max.

* 1. minimum element of arr[n] is min.
  2. range=(max-min)/n
  3. n=number of elements in arr[n].
  4. Each of b[n], i.e, b[i] is a bucket
  5. Number of buckets same as “n”

Since arr[ ] have 10 elements,10 buckets are made.

0

0.6

1

1.9

2

3.0

3.0

3

4

5

4.8

5

6

7

8

7.8

8

9

9.8

10.

This is before sorting each bucket.

After sorting each bucket: in ascending order

0

0.6

1

1.9

3.04

3.07

2

3

4.8

5

4

5

6

7.8

7

8

8

9

9.8

10.1

Then elements in arr[ ] is replaced by sorted elements from each bucket.

# Design Techniques used:-

Divide and conquer is used to divide the given array in to buckets. For individual buckets sorting algorithm is used.

In divide and conquer approach, the problem in hand ,is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing sub problems into even smaller sub problems, we may eventually reach a stage where no more division is possible.

Sorting is a process of ordering or placing a list of elements from a collection in some kind of order. It is nothing but storage of data in sorted order. It arranges the data in a sequence which makes searching easier.

# Algorithm and Explanation of algorithm with example: -

**Divide And Conquer-**

1. Find maximum element and minimum element from the array
2. Calculate the range of each bucket range=(max-min)/n

n is the number of buckets

1. Create n buckets of calculated range
2. Scatter the array elements to these buckets Bucket Index=(arr[i]-min)/range
3. Now sort each bucket individually
4. Gather the sorted elements from buckets to original array

**Pseudo Code:** arr[n], b[n]→buckets for i=0 to n-1:

diff=(a[i]-minimum)/range – int((arr[i]-minimum)/range)

if diff==0 and arr[i]!=minimum:

push arr[i] to b[int((arr[i]-minimum)/range)-1] elses

push arr[i] to b[int(arr[i]-minimum)/range)] for i=0 to n-1:

sort(b[i])// in ascending or descending

# Explanation of algorithm with example:-

Let the array be arr[10]={11.8,6.2,3.7,8.4,7.34,5.56, 6.54,2.98,0.58,1.33}

minimum of arr = min\_ele=0.58 maximum of arr = max\_ele=11.8 range=(11.8-0.58)/10=1.122

b[n] be an array of linked lists.

for i=0 to n-1: i=0:

diff=(arr[0]-min\_ele)1.122 - int(arr[0]-min\_ele/1.122)

=10-10=0

diff==0 and arr[0]!=0.58

b[int((arr[0]-min\_ele)/range)-1]=11.8 b[9]=11.8

i=1:

diff=(arr[1]-min\_ele)1.122 - int(arr[1]-min\_ele/1.122)

=5.0089-5=0.0089

diff!=0

b[int((arr[0]-min\_ele)/range)]=6.2 b[5]=6.2

i=2:

diff=(arr[2]-min\_ele)1.122 - int(arr[2]-min\_ele/1.122)

=2.78-2=0.78

diff!=0

b[int((arr[0]-min\_ele)/range)]=3.7 b[2]=3.7

i=3:

diff=(arr[3]-min\_ele)1.122 - int(arr[3]-min\_ele/1.122)

=6.97-6=0.97

diff!=0

b[int((arr[3]-min\_ele)/range)]=8.4 b[6]=8.4

i=4:

diff=(arr[4]-min\_ele)1.122 - int(arr[4]-min\_ele/1.122)

=6.025-6=0.025

diff!=0

b[int((arr[4]-min\_ele)/range)]=0.025 b[6]=0.025

i=5:

diff=(arr[5]-min\_ele)1.122 - int(arr[5]-min\_ele/1.122)

=4.438-4=0.438

diff!=0

b[int((arr[5]-min\_ele)/range)]=5.56 b[4]=5.56

i=6:

diff=(arr[6]-min\_ele)1.122 - int(arr[6]-min\_ele/1.122)

=5.312-5=0.312

diff!=0

b[int((arr[6]-min\_ele)/range)]=6.5 b[5]=6.54

i=7:

diff=(arr[7]-min\_ele)1.122 - int(arr[7]-min\_ele/1.122)

=2.14-2=0.14

diff!=0

b[int((arr[7]-min\_ele)/range)]=2.98 b[2]=2.98

8

i=8:

diff=(arr[8]-min\_ele)1.122 - int(arr[8]-min\_ele/1.122)

=0-0=0

diff==0 but arr[8]==min\_ele b[int((arr[8]-min\_ele)/range)-1]=0.58

b[0]=0.58

i=9:

diff=(arr[9]-min\_ele)1.122 - int(arr[9]-min\_ele/1.122)

=0.668-0=0.668

diff!=0

b[int((arr[9]-min\_ele)/range)]=1.33 b[0]=1.33

Buckets before sorting-

0

1.33

0.58

NULL NULL

5.56

4

3

2.98

3.7

2

1

5

6.2

6.54

6

NULL

7

7.34

8.4

NULL

9

11.8

Buckets after sorting-

0

1.33

0.58

N N

7

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  | ULL | | | | |
|  |  |  |  |  |  |  |  |  |
| 2 |  |  |  | 2.98 |  |  | 3.7 |  |
|  |  |  |  |  |  |  |  |  |
| 3 |  |  |  | ULL | | | | |
| 4 |  |  |  | 5.56 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| 5 |  |  |  | 6.2 |  |  | 6.54 |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| 6 |  |  |  | 7.3 |  |  | 8.4. |  |

NULL NULL

These elements in sorted buckets are placed in arr[ ] arr[ ]={0.58,1.33,2.98,3.7,5.56,6.2,6.54,7.34,8.4,11.8}

11.8

9

8

arr[ ] is sorted in ascending order

# Time Complexity Analysis: -

Time complexity: if “'n" be number of array elements and buckets

Best Case: O(n) Worst Case: O(n²)

Space Complexity: O(n)

# Conclusion: -

Through bucket sort algorithm, an array of numbers are distributed into buckets, which were implemented by linked list, and each bucket is sorted individually, later sorted elements from these buckets were sent to away.

Hence Bucket sort is analyzed and implemented.

# References: -

* 1. <https://www.programiz.com/dsa/bucket-sort>
  2. https://en.wikipedia.org/wiki/Bucket\_sort
  3. https:/[/www.g](http://www.geeksforgeeks.org/bucket-sort-2/)e[eksforgeeks.org/bucket-sort-2/](http://www.geeksforgeeks.org/bucket-sort-2/)