Karnatak Law Society’s

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

Karnataka, India.



A Seminar Project Report on

**BUCKET SORT ALGORITHM**

Submitted for the requirements of 4th semester B.E. in CSE

for **“DESIGN AND ANALYSIS OF ALGORITHMS (18CSL44)”**

**Submitted by**

**NAME USN**

**1) Ms. Shivani V Banke 2GI20CS140**

**2) Ms. Shradha Mallikarjun Patil 2GI20CS144**

**3) Ms. Srushti B Mudennavar 2GI20CS158**

**Under the guidance of**

**Prof. Pankaja Patil**

**Asst. Prof., Dept. of CSE**

**Academic Year 2021-2022 (Even semester)**

Karnatak Law Society’s

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

Karnataka, India.

**Department of Computer Science and Engineering**



**Certificate**

This is to certify that the Course Seminar report titled **“Bucket Sort Algorithm”** carried out by **Ms. Shivani V Banke, Ms. Shradha Mallikarjun Patil, Ms. Srushti B Mudennavar** bearing **USNs: 2GI20CS140, 2GI20CS144, 2GI20CS158** for **DESIGN AND ANALYSIS OF ALGORITHMS (18CSL44)** course is submittedin partial fulfilment of the requirements for 4th semester B.E. in **Computer Science and Engineering,** Visvesvaraya Technological University, Belagavi. It is certified that all corrections/ suggestions indicated have been incorporated in the report. The course project report has been approved as it satisfies the academic requirements prescribed for the said degree.

Date: Signature of Guide

Place: Belagavi Prof. Pankaja Patil

Asst., Prof., Dept. of CSE

KLS Gogte Institute of Technology, Belagavi

Karnatak Law Society’s

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

**Academic Year 2021-22 (Even Semester)**

**Semester: IV**

**Course: DESIGN AND ANALYSIS OF ALGORITHMS (18CSL44)**

**Rubrics for evaluation of Course Project**

**Student Name: Ms. Shivani Banke**

**Student USN: 2GI20CS140**

**Student Branch: Computer Science and Engineering**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Project Component** | **Max. Marks** | **Marks Earned** |
| 1 | Relevance of the project and its objectives | 02 |  |
| 2 | Tools/Framework used | 01 |  |
| 3 | Methodology / Design | 02 |  |
| 4 | Implementation and Results | 03 |  |
| 5 | Project Report | 02 |  |
|  | **Total** | **10** |  |

Karnatak Law Society’s

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

**Academic Year 2021-22 (Even Semester)**

**Semester: IV**

**Course: DESIGN AND ANALYSIS OF ALGORITHMS (18CSL44)**

**Rubrics for evaluation of Course Project**

**Student Name: Ms. Shradha Mallikarjun Patil**

**Student USN: 2GI20CS144**

**Student Branch: Computer Science and Engineering**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Project Component** | **Max. Marks** | **Marks Earned** |
| 1 | Relevance of the project and its objectives | 02 |  |
| 2 | Tools/Framework used | 01 |  |
| 3 | Methodology / Design | 02 |  |
| 4 | Implementation and Results | 03 |  |
| 5 | Project Report | 02 |  |
|  | **Total** | **10** |  |

Karnatak Law Society’s

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008

**Academic Year 2021-22 (Even Semester)**

**Semester: IV**

**Course: DESIGN AND ANALYSIS OF ALGORITHMS (18CSL44)**

**Rubrics for evaluation of Course Project**

**Student Name: Ms. Srushti B Mudennavar**

**Student USN: 2GI20CS158**

**Student Branch: Computer Science and Engineering**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Project Component** | **Max. Marks** | **Marks Earned** |
| 1 | Relevance of the project and its objectives | 02 |  |
| 2 | Tools/Framework used | 01 |  |
| 3 | Methodology / Design | 02 |  |
| 4 | Implementation and Results | 03 |  |
| 5 | Project Report | 02 |  |
|  | **Total** | **10** |  |

**ABSTRACT**

**Bucket sort**, or **bin sort**, is a sorting algorithm that works by distributing the elements of an array into a number of buckets.  It is a distribution sort, a generalization of pigeonhole sort that allows multiple keys per bucket, and is a cousin of 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 algorithm.

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Content** | **Page No.** |
| 1 | Bucket Sort Algorithm | **1** |
| 2 | Tools/Framework used | **1** |
| 3 | Methodology / Design | **2** |
| 4 | Algorithm / Pseudocode | **5** |
| 5 | Analysis and Performance | **5** |
| 7 | Applications of the Bucket Sort Algorithm | **6** |
| 8 | Conclusion | **6** |
| 9 | References | **6** |

**Bucket Sort Algorithm:**

Bucket Sort is a sorting technique that works on an algorithm that takes an unsorted array as input and returns an array of sorted elements. The working principle of bucket sort is to divide the given series of numbers into several buckets, sort the data in each bucket according to the requirement, and then merge the data again that outputs an array of sorted elements.

Bucket Sort follows the Scatter-order-gather approach since the elements are scattered first into the respective buckets and are then sorted in the bucket and gathered to form a sorted array as the final step.

**The basic procedure of performing the bucket sort is given as follows -**

* First, partition the range into a fixed number of buckets.
* Then, toss every element into its appropriate bucket.
* After that, sort each bucket individually by applying a sorting algorithm.
* And at last, concatenate all the sorted buckets.

**The advantages of bucket sort are -**

* Bucket sort reduces the no. of comparisons.
* It is asymptotically fast because of the uniform distribution of elements.

**The limitations of bucket sort are -**

* It may or may not be a stable sorting algorithm.
* It is not useful if we have a large array because it increases the cost.
* It is not an in-place sorting algorithm, because some extra space is required to sort the buckets.

**Tools/Framework used:**

CodeBlocks IDE and C programming Language

**Methodology and Design:**

For Floating point numbers:

**bucketSort(arr[], n)**

1) Create n empty buckets (Or lists).

2) Do following for every array element arr[i].

.......a) Insert arr[i] into bucket[n\*array[i]]

3) Sort individual buckets using insertion sort.

4) Concatenate all sorted buckets.

Let us take an input array of 7 elements.

The input array is:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.54 | 0.44 | 0.45 | 0.64 | 0.49 | 0.59 | 0.10 |

Create an empty array of size 10, as shown below. Each slot of this list or array is to use as a bucket for storing elements.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

To insert elements in the respective buckets, follow the following three steps:

* Let us take an element from the input array as an example that is 0.54
* Multiply it by the size of the empty array(size = 10) and convert the result to an integer using int(). You will get:

0.54 × 10 = 5.4

On converting it to integer, we get, int (5.4) = 5

Finally, 5.4 is inserted into bucket-5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.54 | 0.44 | 0.45 | 0.64 | 0.49 | 0.59 | 0.10 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0.54  0.59 | 0 | 0 | 0 | 0 |

Similarly, 0.59 is also inserted into the same bucket, bucket-5, because the two elements’ floor value is equal.

The above process continues until all the elements settle in their respective buckets.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.10 | 0 | 0 | 0 | 0.44  0.45  0.49 | 0.54  0.59 | 0.64 | 0 | 0 | 0 |

The elements of each bucket are then sorted using any of the algorithm techniques.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.10 | 0 | 0 | 0 | 0.44  0.45  0.49 | 0.54  0.59 | 0.64 | 0 | 0 | 0 |

Finally, the elements from the buckets have been gathered by passing the elements into the original array by iteration.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0.10 | 0.44 | 0.45 | 0.49 | 0.54 | 0.59 | 0.64 |

For Integers:

Step1-Take input array find the MAX number in the array

Step2-Define 10 queues each representing bucket for each digit from

0to9.

Step3-Consider the least significant digit of each number in the

list which is to be sorted.

Step4-Insert each number into their respective queue based on the

least significant digit.

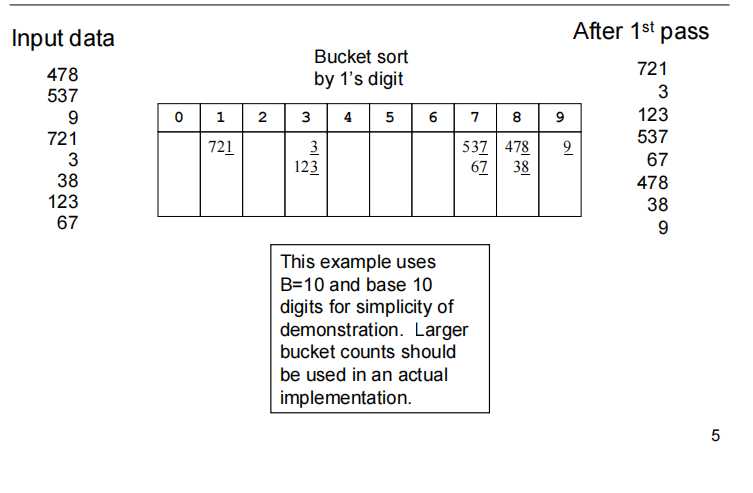
Step5-Group all the numbers from queue0to queue9in the order

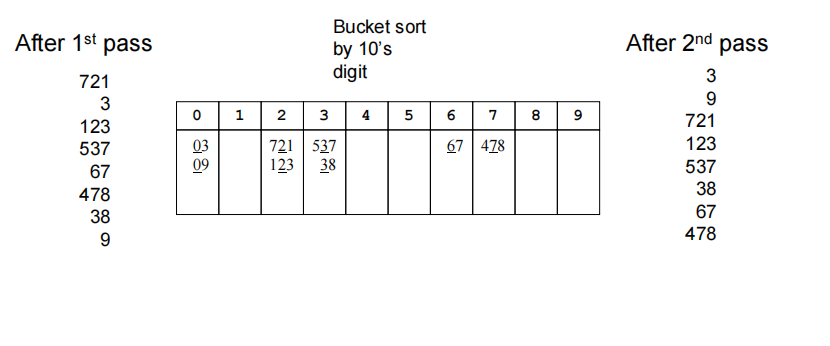
they have inserted into their respective queues.

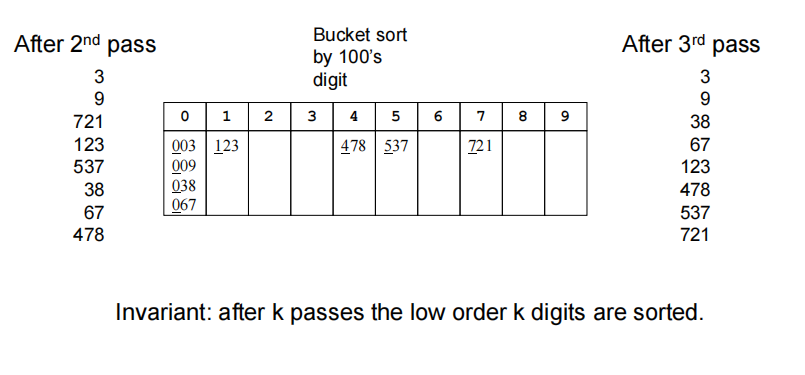
Step 6- Repeat from step3based on the next least significant digit.

Step7-Repeat from step2until all the numbers are grouped

based on the most significant digit.







**Algorithm and Pseudocode:**

1. Bucket Sort(A[])
2. Let B[0...n-1] be a new array
3. n=length[A]
4. for i=0 to n-1
5. make B[i] an empty list
6. for i=1 to n   do
7. insert A[i] into list B[n a[i]]
8. for i=0 to n-1 do
9. sort list B[i] with insertion-sort
10. Concatenate lists B[0], B[1],........, B[n-1] together in order
11. End

**Analysis and Performance:**

* Best Case Time Complexity: O(n+k)
* Average Case Time Complexity: O(n)
* Worst Case Time Complexity: O(n^2^)
* **Best case complexity:**best case complexity occurs when there are an equal number of elements in the buckets and, it will be even better if the elements inside the buckets are already sorted. Suppose insertion sort is used to sort the elements in the buckets. In that case, the overall best-case complexity will be O(n+k), where O(n) is the complexity for making the buckets and O(k) is the complexity for sorting the elements in the buckets.
* **Average case complexity:** Average case complexity occurs when the elements are distributed randomly in the array of buckets. In general, Bucket sort runs in linear time in all cases until the sum of the squares of the bucket sizes is linear in the total number of elements. Finally, this makes O(n) to be the average-case complexity of the bucket sort algorithm.
* **Worst-case complexity:** In General, one bucket contains many elements when there are elements of the normal range, but when there are elements of the very close range, they are more likely to fall in the same bucket. So, in this case, more elements get placed in the same bucket resulting in more number elements in some buckets than others. The algorithm complexity depends on the number of elements in the bucket and, it becomes even worse when the elements in the bucket are in reverse order. When insertion sort is used to sort the elements then the time complexity will be O(n2)

**Applications of the Bucket Sort Algorithm:**

1. The contact list in your phone is sorted, which means you can easily access your desired contact from your phone since the data is arranged in that manner for you. In other words, “it is sorted”.
2. While shopping on flip kart or amazon, you sort items based on your choice, that is, price low to high or high to low.

**Conclusion:**

* The bucket Sort algorithm sorts the elements of the array by first segregating the array into a number of buckets, sorting each bucket, and then gathering the elements back to form the sorted array.
* Bucket Sort is used to sort an array where elements are uniformly distributed, or where the elements of the array range between 0 and 1.
* Bucket sort can exhibit the best case time complexity of O(n+k), where n is the number of buckets and k is the bucket size.
* The way buckets are provided ranges differ in cases when the elements of the array are floats and integers. This is discussed in detail above.

**References:**

Bucket sort Wikipedia

https://en.wikipedia.org/wiki/Bucket\_sort