**CS325 Research Project**

Landon Baer, Tim Sprunger

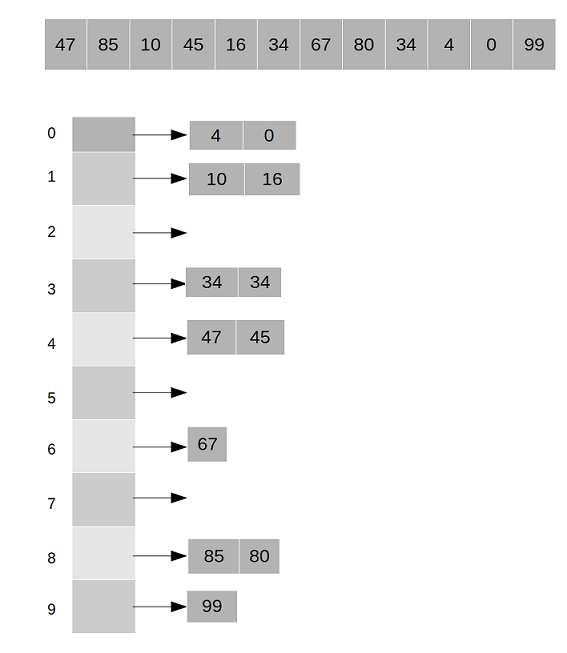
12-6-19

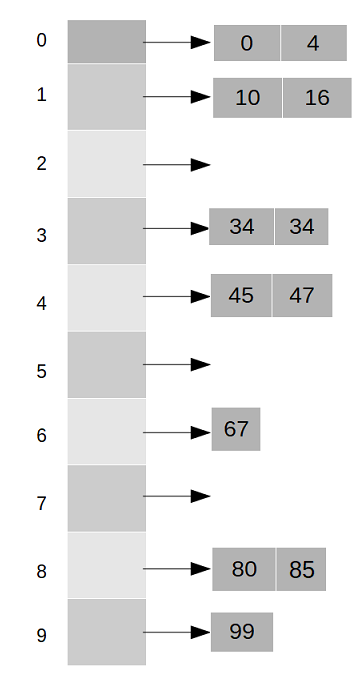
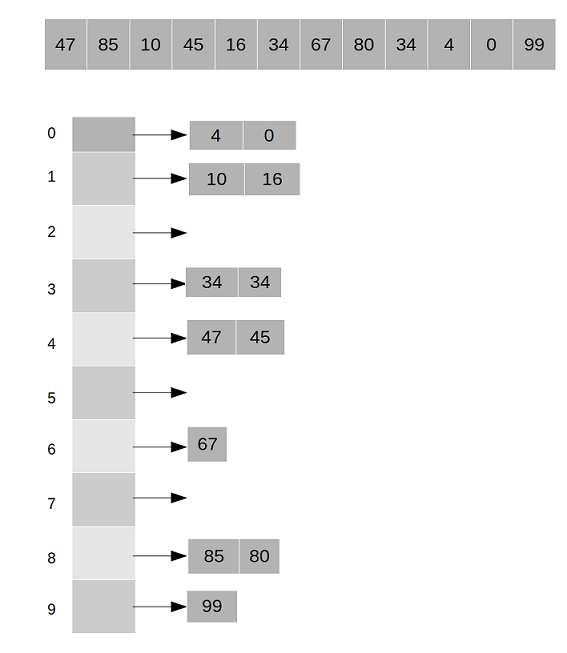
Bucket Sort

**Order of Magnitude:** O(N) but is known to have faster sorting relative to other sorting O(N) algorithms like merge sort or quick sort

**Overall Concept:** Places values in “buckets” depending on its hash value, sorts the buckets using another sorting algorithm, and then pulls the values out of the buckets, resulting in a sorted array.

**Assumptions:** Values in initial array are all positive integers from 0-99 (range could be changed depending on hash function and/or number of buckets)

Example

1. Start with an unsorted integer array
2. Place values in appropriate buckets based on hash function (i.e. divide by 10)
   1. 4/10 returns 0, so it’s placed in bucket 0
   2. 85/10 returns 8, so it’s placed in bucket 8
   3. Buckets are implemented as linked lists, so all values with the same hash value can be stored in the same bucket
3. Sort the buckets using another sort (i.e. insertion)

Unsorted Buckets 🡪 Sorted Buckets

1. Pull values from bucket array in order (i.e. 0, 4, 10, etc…) and place them into the original array
2. Return sorted array

Modifications

1. **Adjusted Hash Function:** Instead of dividing the value by 10, we calculated the square root of the value.

 private static int hash(int number){

        return (int) Math.sqrt(number);

    }

1. **Option to Create Array of Set Size Using Command Line:** java bucketSortModified **15** 150

//Allow for creating an array of set size from CMD

            int[] integerArray = new int[ Integer.parseInt(args[0]) ];

1. **Option to Fill Array w/ Seed Values Using Command Line:** java buketSortModified 15 **150**

 //Allow for array to be filled w/ seed values from CMD

            for (int a=0; a<integerArray.length; a++) {

                integerArray[a] = seed.nextInt(100);

1. **Place Values Back into Array Without Sorting the Buckets, Run Insertion Sort on Array**

// Assign numbers from array to the proper bucket by using hash function

        for(int x=0; x<array.length; x++){

            buckets[ hash(array[x]) ].add( array[x] );

        }

        // Merge buckets to get sorted array

        int z=0;

        for (int v=0; v<buckets.length; v++){

            for (int n=0; n<buckets[v].size(); n++) {

                array[z++] = buckets[v].get(n);

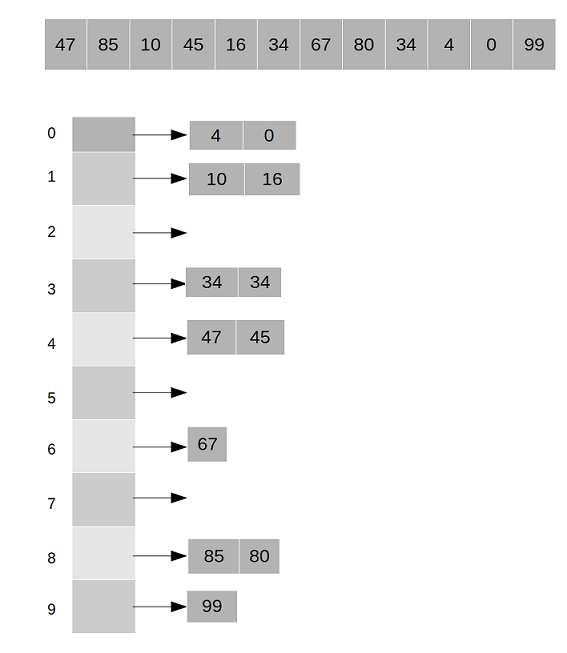
            }

        }

        //INSERTION SORT OF DATA

        insertionSort(array);

New Procedure





1) Unsorted Array

2) Place Values into Appropriate Buckets

3) Place Bucket Values into Original Array

4) Sort Array

5) Sorted Array





**Note:** Because of the square root hash function, it isn’t as clear which bucket it should be in. However, the numbers are more sorted at this point than with a simple /10 hash function.