CHAPTER 4
ANALYSIS AND DESIGN

4.1 Analysis

Since first use in 1970, algorithms visualization are highly evolved to use as additional learning material at computer science lecturer. In case to learning algorithms, visualization is one of communication method which can build a representation about an algorithm and exploring how it works. That was made more developed by needs about interactive algorithms visualization in case of system and programming environment.

Among various algorithms, sorting is one of basic algorithm that often learning in computer science education. Sorting algorithms could be divided by existences of comparing process, which many people often call it as comparisonal and non-comparisonal. Comparisonal sorting algorithm was often found in basic computer science learning, but in other side non-comparisonal sorting algorithm is rarely encountered. Although in fact, non-comparisonal sorting algorithm is one of linear algorithms. Linear algorithm has a good time complexity while executing data, but this algorithm needs more memory. This explanation also underlies two non-comparisonal sorting algorithms, which are Counting and Radix Sort stand as main material on this project.

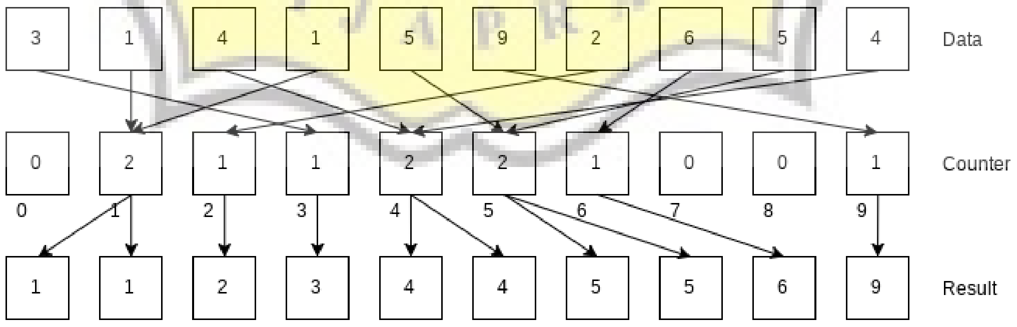


Illustration 4.1: Counting Sort Illustration

Counting sort algorithm is a sorting technique based on keys between a specific range. Spesific range can obtained by finding minimum until maximum value of initial array. It works by counting every element of initial array which has same value with prepared array index. Then doing counting back technique to calculate position of each object as sequential.

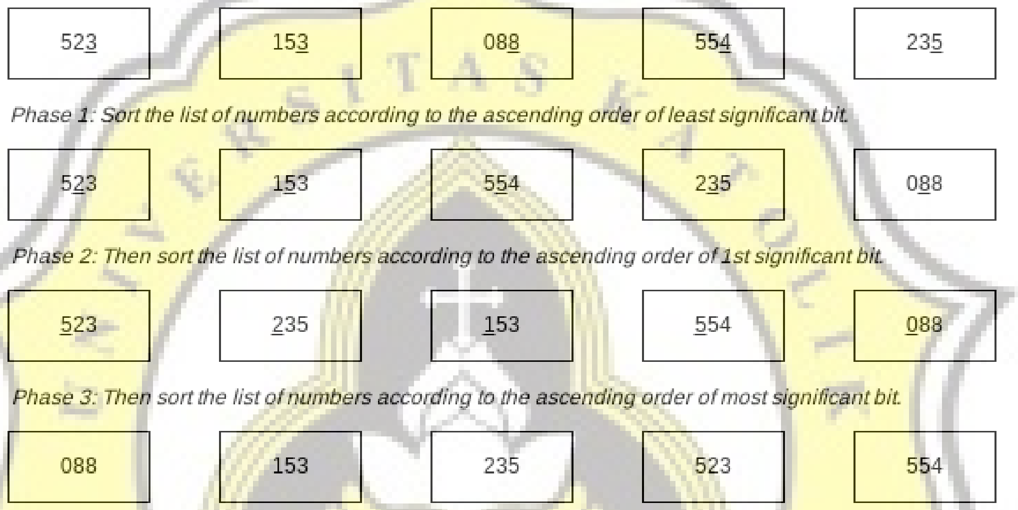
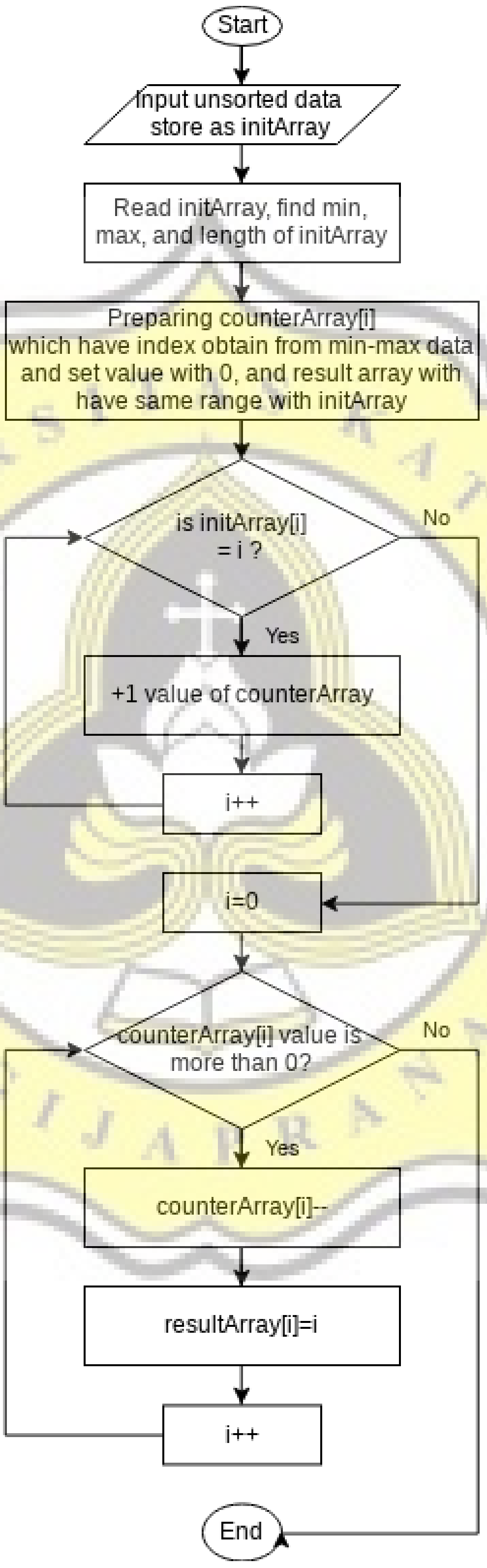

523
153
088
554
235
Phase 1: Sort the list of numbers according to the ascending order of least significant bit.
523
153
554
235
088
Phase 2: Then sort the list of numbers according to the ascending order of 1st significant bit.
523
235
153
554
088
Phase 3: Then sort the list of numbers according to the ascending order of most significant bit.
088
153
235
523
554

Illustration 4.2: Radix Sort Illustration

Main idea of *Radix* sort algorithm was exists because *counting* sort needs wasteful memory at sorting big data. Essense of radix sort is doing digit by digit putting sub value into bucket incrementaly, starting from least significant digit or vice versa. Same as *counting* sort, this algorithm also grouping sub value which was limited by key-range spesific into basic number bucket. How much iteration are same amount with how many digit of maximum value from data.

4.2 Design

Design of application is needed as a basic for guidance in generating source code. Shade of design will be obtained by creating a rough idea of an application and write it into a form of flow chart diagram.


Start
Input unsorted data store as initArray
Read initArray, find min, max, and length of initArray
Preparing counterArray[i] which have index obtain from min-max data and set value with 0, and result array with have same range with initArray
is initArray[i] = i ?
No
Yes
+1 value of counterArray
i++
i=0
counterArray[i] value is more than 0?
No
Yes
counterArray[i]--
resultArray[i]=i
i++
End
Illustration 4.3: Flowchart of Counting Sort Algorithm

Based on flowchart on illustration 4.3 above, counting sort algorithm will processing initial data as *initArray* variable. Afterward, system will searching minimum, maximum, and size of *initArray* values. Then preparing *counterArray* variable with has index obtain from range between minimum until maximum value from *initArray* values. *counterArray* values will add one value every *initArray* value are same with *counterArray* index. Every *counterArray* index which have more than zero value will transfered into *resultArray* variables incrementaly, at the same time value of *counterArray* will minus by one.

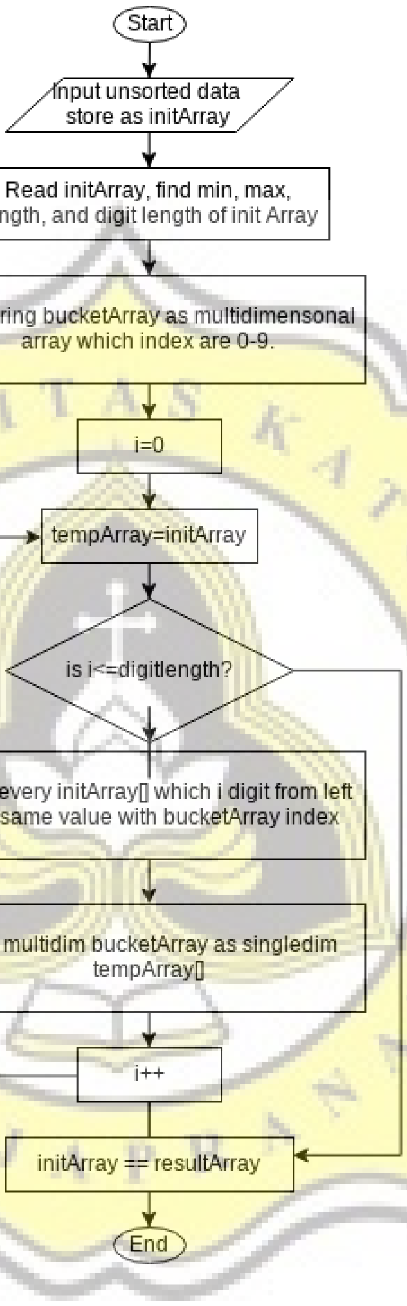

Start
Input unsorted data store as initArray
Read initArray, find min, max, length, and digit length of init Array
Preparing bucketArray as multidimensonal array which index are 0-9.
i=0
tempArray=initArray
is i<=digitlength?
store every initArray[] which i digit from left has same value with bucketArray index
split multidim bucketArray as singledim tempArray[]
i++
initArray == resultArray
End

Illustration 4.4: Flowchart of Radix Sort Algorithms

Radix sort algorithm flowchart are drawn on illustration 4.4 above. This algorithm will preparing *initArray* variables and search maximum value, length of maximum value and size of *initArray*. This algorithm is need place to storing data which was called it as bucket. Bucket has index by 0 until 9 which are obtain from basic number. In *radix* sort, each *initArray* data will be store repeatedly on bucket based from *initArray* sequence, which if each number from least significant digit from each *initArray* value are same with index of bucket.

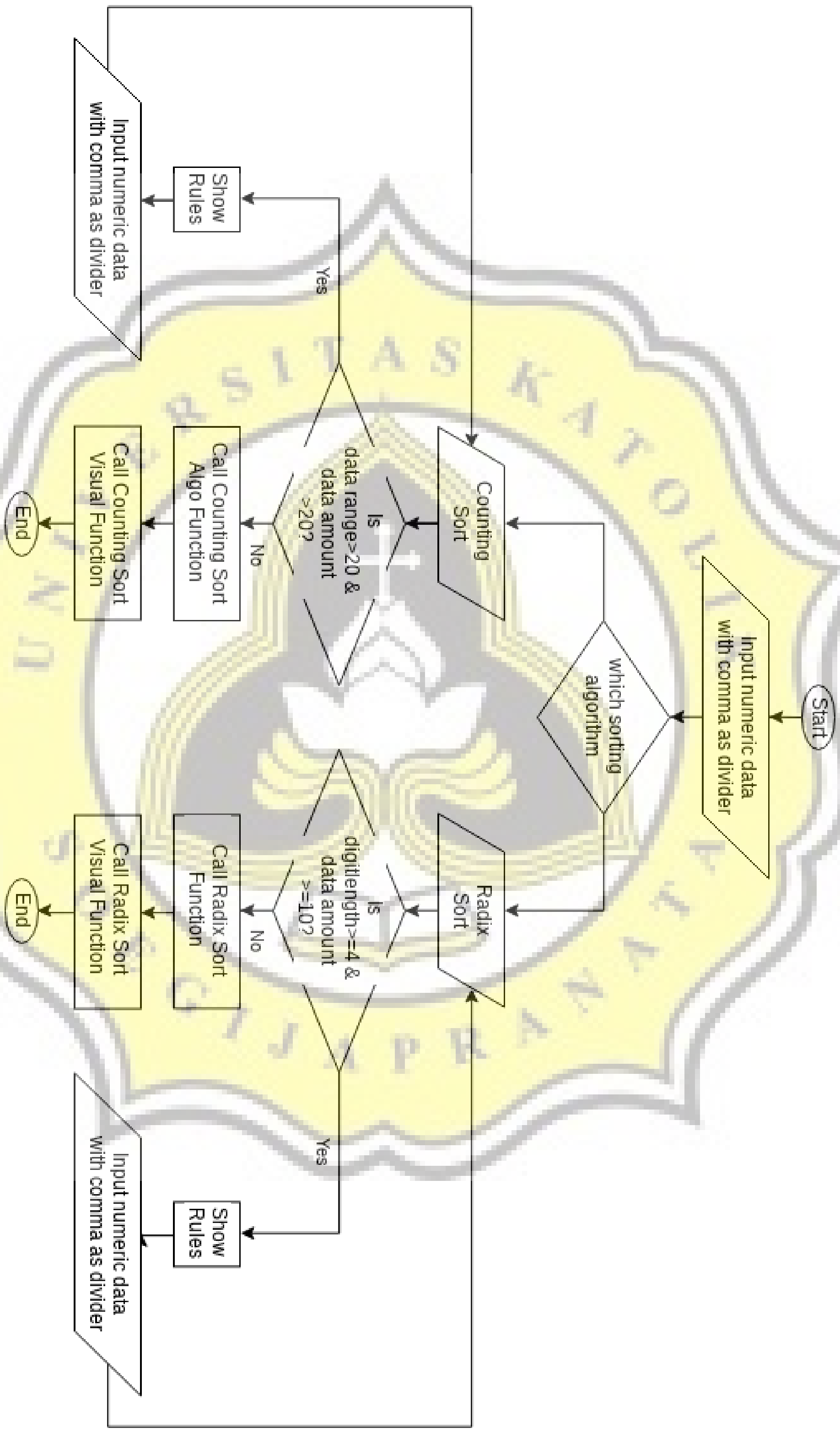

Start
Input numeric data with comma as divider
which sorting algorithm
Counting Sort
Radix Sort
Is data range>20 & data amount >20?
Yes
No
Show Rules
Input numeric data with comma as divider
Call Counting Sort Algo Function
Call Counting Sort Visual Function
End
Is digitlength>=4 & data amount >=10?
Yes
No
Show Rules
Input numeric data with comma as divider
Call Radix Sort Function
Call Radix Sort Visual Function
End

Illustration 4.5: Flowchart of Entire Application

According flowchart diagram on illustration 4.5, user starts application by inputting unsorted data. After that, user choose algorithm to be used to sorting data, and system will call selected sort algorithm function and visualization function. Then user will shown visualization from selected algorithm on each page.