|  |  |
| --- | --- |
| CAB301 Assignment 1  Empirical Analysis of an Algorithm | N9845097  Ka Long Lee (Eric)  Due: Friday, 12th April 2019 |

Table of Contents

[1. Description of Algorithm (1 Page) 2](#_Toc5387596)

[2. Implementation of the Algorithm (2 Pages) 2](#_Toc5387597)

[2.1. Functional correctness 2](#_Toc5387598)

[3. Experiment Design (2 Pages) 2](#_Toc5387599)

[3.1. Methodology, tools and techniques 2](#_Toc5387600)

[3.2. Data sizes, Test Data set 2](#_Toc5387601)

[4. Experiential results (4 Pages) 2](#_Toc5387602)

[5. Analysis of Experiential results 3](#_Toc5387603)

[6. Reference (1 Pages) 3](#_Toc5387604)

[7. Appendices (4Pages) 3](#_Toc5387605)

# Description of Algorithm (1 Page)

The main purpose of the Brute Force Median algorithm is to find out the median value in the given array which contains a set of integer value. Considering the different condition that the integer value set can be, we cannot simply take the value in the middle position of the array as the median value of the data set. Because the value set might be sorted randomly. That is the reason why we need to implement Brute Force Median algorithm to retrieve the median value in the array accurately.

The algorithm should return the correct median value in the integer value set despite that the order is sorted randomly, in ascending order or in descending order. It also able to return the correct median value in the condition that the length of array value set is odd or even numbers. Duplicate value in the array is also one of the factors we need to consider for implementing the algorithm.

This algorithm takes only one parameter which is an array contains a set of integer numbers. First of all, the algorithm creates one local variable named as k. It is for storing the median position of the input array. There are two steps for calculating the median position of the array. Firstly, dividing the length of the input array by two. Secondly, ceiling the previous result to the highest integer value it can be. Finally, the result will be assigned to the k variable. If the input array was already sorted before it passed to this function, the median value of the array must be located at the k position.

The algorithm then creates a nested for loop which will repeat according to how many integer numbers exists in the input array. There are total two for loop in this case. Both loops create an indexer variable for selecting elements in the array for different purpose. Indexer variable I is created and updated by the outer loop. In contrast, Indexer j is created and updated by the inner loop. The main purpose of the outer loop is to determine the array element selected by indexer i whether median or not. On the other hand, the purpose of the inner loop is to calculate how many elements are smaller and equals to the value selected by indexer i. At the end of the outer loop,

# Implementation of the Algorithm (2 Pages)

# Functional correctness

# Experiment Design (2 Pages)

# Methodology, tools and techniques

# Data sizes, Test Data set

# Experiential results (4 Pages)

# Basic Operation identified

# Average execution time

# Experience to measure the program execution times

# Analysis of Experiential results

# Experimental results

# Comparing against the theoretical efficiency prediction

# Reference (1 Pages)

# Appendices (4Pages)