Finding Maxima and Minima Algorithms Time Complexity Analysis

Tarun Kumar
UE143103
Computer Science Engineering
University Institute of Engineering and Technology
Panjab University
Chandigarh, 160030
Email: tarunk.1996@gmail.com

***Abstract*—This practical analyses the search time complexity of finding minimum and maximum valued element (or minima and maxima) from a list of numbers using two different approaches and comparing the result obtained. The two approaches used here are Brute force and Divide & conquer algorithm for finding the minimum and maximum valued element. Both of the algorithm has a linear time complexity (i.e. $O(n)$). This practical analyses the search time complexities of two algorithms i.e brute force and divide and conquer search.**

I. Introduction

Brute Force is a technique for finding an element in a given list by accessing all the elements in the given list individually. This is one of the easiest algorithm which serves the purpose. In this paper, Brute force algorithm is used for finding the minimum and maximum valued elements in a given list of numbers. Based on Brute force algorithm, all the elements must be accessed and out of that minimum and maximum valued element are evaluated. Initially, the first element is assigned to minimum and maximum variable. Then each element in the list is compared with minimum variable if the number is smaller than minimum, assign that number to minimum, similarly for maximum if the number is greater than maximum then assign maximum with that number, do this until the list is exhausted completely. At the end, Maximum and minimum of out of that list is present in the variables return maximum and minimum to the main function. This algorithm has a $O(n)$ time complexity.

The other algorithm which is used for finding the minimum and maximum valued element in given set of numbers(list) is Divide and Conquer algorithm. In Divide and Conquer algorithm, The list is divided into several smaller list until a single element is left. Once the list is indivisible, we select the minimum and maximum valued element out of it and as the function is recursively called by passing array, minimum, maximum and size of the array as arguments to the function. Once minimum and maximum is returned to the parent function it evaluates the minimum and maximum obtained from the two function called above and evaluates the minimum and maximum out of it. This process is carried out until the maximum and minimum is not returned to the main function. This algorithm also have a $O(n)$ time complexity.

II. Problem

To analyze the time required to find maxima and minimua from a list of numbers using both the approaches (i.e Brute Force and Divide and Conquer Algorithm).

III. Aim

To write a program which finds the average time required to find minimum and maximum valued elements out of a list filled with random numbers using Brute force and Divide and Conquer algorithm. Also, plot a graph demonstrating the total average time required to find the maximum and minimum valued element where the size of list increases gradually. Discuss the factors involved in the nature of the curve observed in the graph.

IV. Approach

Define an array of integers and some variables which will be used to compute the maxima and minima in the list. Also initialize a 2D array for storing the time taken to find the minimum and maximum valued element in the array using both algorithms and for different size values of the array. Populate the array of integers with random numbers using random function within a certain range. Now note the clock time or assign the current system time to some variable. Call the MinmaxDC user defined function while passing the necessary variables as arguments. Once, the function is called the control is given to the function which evaluates the Maximum and Minimum elements from the list passed as argument based on the algorithm specified below in the paper. After the execution of the function, the control of the program returns back to the main function. Note the current clock time in another variable and evaluate the total time required by the function to find the minimum and maximum valued elements in the list. Store the evaluated time in the 2D array along with the size of the array. Now, again note down the system clock time into some variable. Similarly for Brute force algorithm call the MinmaxBF function passing the necessary variables as

arguments. Once the function returns the program counter back to main function after finding the minimum and maximum valued elements in the list based on the algorithm specified in the below section. Again note down the system clock time and evaluate the time and store the resultant time in the 2D array. Repeat the following above steps multiple number of times to get a precise and accurate result. Also run these steps to for different sizes of the array and plot the graph accordingly.

V. Experiment

To achieve the goal of the experiment,the program must follow some particular criteria which is discussed in this section in detail. The experimental results obtained in this practical are based on the following approach and implementation. Brute force and Divide & Conquer algorithm problem can be solved by executing following steps :

1) Declare an array of size 1000 and some more variables which will be used in future of type integer.
2) Declare another 2D array having 50 rows and 3 columns for storing the average time taken in order to find the minimum and maximum number out of the given list using both algorithms oone by one.
3) Start a loop where the size of the array keeps on increasing by a factor of 50. Initially starting with 50 as the size of array upto 700.
4) Populate the first array with random numbers using rand() function(Make sure to specify a domain for it). It can be done using rand()%domain statement.
5) Set the current system clock time to a variable using clock() function.(It will be used to calculate the time taken to find the maximum and minimum in the list).
6) Call the minmaxBF() function by passing the required arguments e.g array itself, size of the array, minimum and maximum variables.
7) As the function returns the required result back to the main function. Again Note down the current system clock time for evaluating the total time required to achieve the goal.
8) Write the size of the array used in the first column and total time taken by the Brute force algorithm to find the minimum and maximum in the list.
9) Again set the current system clock time to a variable using clock() function.(It will be used to calculate the time taken to find the maximum and minimum in the list).
10) Call the minmaxDC() function by passing the required arguments e.g array itself, beginning , ending of array, minimum and maximum variables)
11) Once the function finds the minimum and maximum valued element in the list. It returns the control back to the main function.
12) Note down the current system clock time, evaluate it with the previous time and assign the value to the third column of the 2D array.
13) Repeat steps from 4 to 12 10000 number of times to get a precise value of average time required to find the minimum and maximum valued element in the list using both algorithm separately.

The results obtained in the following experiment is performed on a Ubuntu - Linux(x64 bit) based operating system. The system consist of 4GB of RAM and have a Intel(R) Core(TM) i3-2330M processor with a clock speed of 2.20GHz. To minimize the hindrance, heavy background process were terminated before running the code.

VI. Algorithms for Finding Minima and maxima in list of numbers

Following Algorithm is based on Divide and conquer algorithm for finding minima and maxima from an array.

Algorithm 1 Using Divide and Conquer

minmaxDC(a, high, low, min, max)
if $low = high$ **then**
  $min \leftarrow a[low]$
  $max \leftarrow a[low]$
  $return$
end if
if $high - low = 1$ **then**
  if $a[low] < a[high$ **then**
    $max \leftarrow a[high]$
    $min \leftarrow a[low]$
  else
    $max \leftarrow a[low]$
    $min \leftarrow a[high]$
  end if
  $return$
end if
$mid \leftarrow (low + high)/2$
$maxmin(low, mid)$
$leftmax \leftarrow max$
$leftmin \leftarrow min$
$maxmin(mid + 1, high)$
if $leftmax > max$ **then**
  $max \leftarrow leftmax$
end if
if $leftmin < min$ **then**
  $min \leftarrow leftmin$
end if
$return$

The second approach involves the Brute force method, i.e we recursively call the function minmaxBF until the whole list is not accessed. After every call the min and max values are updated if they fulfill required condition. Min and max variables are called by reference so that the values of the min and max are retained while giving the control back to main function and data are not lost.

Algorithm 2 Brute Force

minmaxBF(a,b,e,*min,*max)
if $b == 0$ **then**
  $*min = *max = a[0]$
else
  if $*min > a[b]$ **then**
    $*min = a[b]$
  end if
  if $*max < a[b]$ **then**
    $*max = a[b]$
  end if
end if
if $b <= e$ **then**
  $minmaxBF(a, b+1, e, min, max)$
end if

VII. Results and Graphs

Following are the graphs obtained after performing the experiment as specified above :

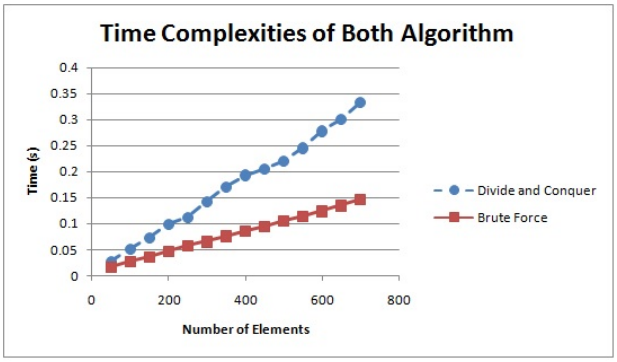

Fig. 1. Time required to find the minimum and maximum valued element in the given list

VIII. Results Obtained

Number of Elements	Time taken in Brute Force (s)	Time taken in Divide and Conquer (s)
50	0.028207	0.01712
100	0.052089	0.027337
150	0.073547	0.03658
200	0.099124	0.046615
250	0.112741	0.057489
300	0.14326	0.065996
350	0.171035	0.076521
400	0.19383	0.085793
450	0.205459	0.095486
500	0.220477	0.105994
550	0.246159	0.115299
600	0.278687	0.124321
650	0.301915	0.134995
700	0.333492	0.146822

IX. Discussion on the results

In figure 1, It shows the growth of the two curve obtained by the performing the given experiment as specified in the earlier section. The curve with circle and dotted lines denotes the average time required (in seconds) by the Divide and Conquer algorithm to find the minimum and maximum valued elements in the list. The growth in the curve is due to the subsequent growth in the size of the list which is increased gradually.
Also, The curve with square and smooth curve denotes the average time required (in seconds) by the Brute Force algorithm to find minimum and maximum valued elements in the list. The increase in the growth of curve is due to increase in the size of array by the factor of 50.
As it is known, that both of the algorithm requires $O(n)$ time complexity to find the minimum and maximum valued elements in the list. By inference of the graph it justifies the statement and proves that both algorithm increases linearly with increase in the number of elements in the list.
To make the two curves more comparable, Brute force algorithm is applied using recursive function calling instead of iteration. In recursive function, array its starting, ending and values of minimum and maximum are passed as arguments to the function whereas, minimum and maximum are called by reference to easily pass on the values of minimum and maximum valued elements to the parent functions. This technique does the same job done by using iterative function calling.

In the following subsections the properties, nature and anomaly inferred from the graph are listed :

A. Nature of the Curve

Dotted curve i.e curve obtained by applying Divide and conquer algorithm for finding minima and maxima in the list of number is more steeper compared to that of obtained using Brute force algorithm for the same purpose.
Both of the curves have a linear growth rate i.e. having time complexity $O(n)$. Even though this behavior of the curve implies that in Divide and Conquer Strategy more number of constant statements are present inside the recursive function resulting in the increase in the steepness of the curve. As according to the specified algorithm which is used in this experiment a total of 3 statements must be executed if base case is encountered (i.e list has only one element) also 3 statements when another base case is encountered (i.e when list has 2 elements) either of them is maxima or minima of the list. But when a general case is encountered 4 statements are performed with 2 recursive function call. While in case of Brute force algorithm only 2 statements are overall executed and 1 statement if its the base case. So, based on the number of constant statements executed inside both of the algorithm determines the steepness of the curve.

B. Properties

Initially starting the size of the array to be 50, and incrementing it with 50 every time the loop is executed unless it reaches 700. This wide range of numbers are taken to get a good idea of the nature of the curve and to get a precise data each particular size of the array is run 80000 number of times and average out of it is taken for evaluating and designing the graph.
In this procedure there is no best case and worst case exists because every time a list is passed to the function the whole list of numbers must be traced or accessed individually based on the algorithms used to do so. This implies the nature of the curve is independent to the distribution of numbers in the list because no such best and worst case exists.

C. Anomalies in the curve

The curve obtained in the above experiment is quite smooth and expected one. But if noticed carefully, a bump in the dotted curve (i.e. Divide and Conquer Algorithm curve) can be seen in it. From 350 to 450 the bump can be seen in the dotted curve. The reasons behind this anomalous behavior could be due to multiple factors. While execution of the program some interrupt or high priority task had been encountered by the CPU due to which CPU indulges itself to perform and execute that task first which causes increase in the average time to find maxima and minima in the curve. Or it could be due to the background process which were running simultaneously during the execution of the program (Mozilla firefox was running when this program was under execution phase). So, due to any of the above reasons the anomalous behavior is observed in the curve. While the remaining curve is as same as expected i.e. linear in nature having time complexity $O(n)$ and supports the assumptions made before it.

X. Conclusion

- In the experiment performed above it is clearly visible that Brute force algorithm is better compared with Divide and Conquer algorithm even though both have same time complexities i.e. $O(n)$ for finding maxima and minima in list of numbers.
- Brute Force algorithm is easy to use and less complex(in terms of user understanding) than Divide and Conquer algorithm where we recursively divide the list into smaller parts until unique element are not formed. While in Brute force all the elements in the list is uniquely compared with the maxima and minima at that point of time.
- The Steepness of curve obtained by Divide and conquer algorithm is more than Brute force algorithm for finding minima and maxima in the list of numbers.
- A small bump in the graph of the Divide and conquer algorithm is due to some CPU background processes or due to high priorities interrupts encountered while execution of the program
- In this experiment i.e finding minima and maxima in the given list. There exists best or worst case, every case requires same amount of time to access all the elements individually in the list. Hence, no every case is a average case itself.

XI. Attachments

The Program code used for performing the above experiment is uploaded here : http://ideone.com/Gjyiza