Analysis of Restoring and Non-Restoring Division Algorithm

By Tim Kellermann, Sudeep Potluri

Due: 29 November 2022

**Abstract**

Throughout this article we will verify the complexity of both the Restoring and Non-Restoring division algorithms. Within this analysis we will include useful metrics such as number of additions/subtractions that will quantify the execution time of both algorithms.

# Motivation

Our goal is to create a software routine which performs a sequence of subtractions. An algorithm is a well-defined procedure that takes an input and produces some output. This definition is rather broad, many things in everyday life can be written as an algorithm. Brushing your teeth, making your bed, tying your shoes, etc. However, there are many ways to achieve the same solution. For example, I could brush my teeth with a toothpick. This procedure would lead to great inefficiencies. Recognizing these inefficiencies, we have the option to choose what algorithm best fits our needs and minimizes the execution time which then leads to a faster machine.

# Introduction

This article we will test both Restoring and Non-Restoring algorithms to verify that the non-Restoring method is faster by using less additions and subtractions.

# Procedure

How did you do this wonderful thing anyway? This is where you put your pseu- docode (see Algorithm 1), pre-conditions, post conditions, and Invariants. De- scribe the idea in a general way. How you implement your procedure goes in the Experimental Analysis.

**Algorithm 1** restoring(R,Q)

1: **procedure** BUILD-MAX-HEAP(A)

2: assert(A is not empty)

3: *heapsize* = *A.length*

4: **for** *i* = *lheapsize/*2*J* **downto** 2 **do**

5: Max-Heapify(A,*i*)

6: assert(Heap-Test(*A, i, heapsize*)

7: **end for**

8: assert(Heap-Test(*A,* 1*, heapsize*)

9: **end procedure**

**Algorithm 2** Non-Restoring(R,Q)

1: **procedure** BUILD-MAX-HEAP(A)

2: assert(A is not empty)

3: *heapsize* = *A.length*

4: **for** *i* = *lheapsize/*2*J* **downto** 2 **do**

5: Max-Heapify(A,*i*)

6: assert(Heap-Test(*A, i, heapsize*)

7: **end for**

8: assert(Heap-Test(*A,* 1*, heapsize*)

9: **end procedure**

# Testing

## Testing Plan and Results

This can be done as a table. Something like:

|  |  |  |  |
| --- | --- | --- | --- |
| Test input | Expected Results | Actual Results | Corrective Actions |
| Empty array | nothing returned | nothing was returned | no action taken |
| All elements identical | same array | Program faulted | off–by–one error corrected |

## Problems Encountered

Normally, you would not see these in a paper, but our projects are a special case. You might see this in a paper that developed a simulator for running virtual experiments. If you had no problems, state “No significant problems were encountered.” Syntax errors and typo’s are not problems. You can give a brief description of how you fixed the problem, for example “Updating to the latest version of java corrected the issues with program faults.” Use your best judgement on what is a problem and what is not. If you got help

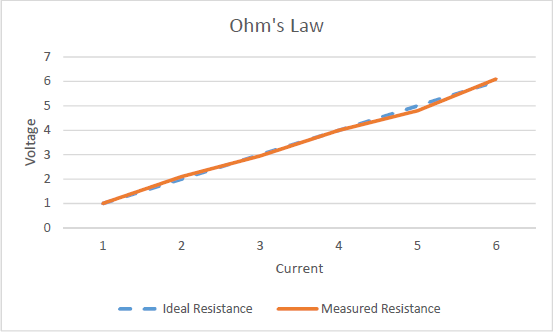


Figure 1: A Sample Graph

with your problem, acknowledge it here or cite the help as a conversation1.

# Experimental Analysis

IF you wish, you can include a paragraph about the specifications of the platform you used to run your experiments. Be very brief: “These timings were carried out on a 427 cubic inch supercharged Dell MT708 with dual overhead BlueRay drives. The power of the CPU was wasted by running Windows 8.1 Pro with all the latest updates as of January 12, 1995. Oracle Java 1.0 was used as the programming language/compiler.”

Here you need to graph your results and explain them. This is the main part of your paper. Graphs should be inserted as figures and referenced by “...agreed well with the expected results, see Figure 1”. Be careful of color-coding parts of graphs as most printing is still black and white. Notice how I used both color and dashes to differentate the two lines on the graph. You should be able to get away with one graph with two lines for each case. Mark *n*0 on the graph if you can. **Label the axes and title your graph.** If you don’t label an axis, you will lose points. There is one allowed exception: Theoretical curves of time complexity can have an axis labeled “time” without units.

Current style is to place tables *before* they are referenced with the caption above the table. Figures are place *after* they are referenced and the caption is below the figure. If

1Look up how to cite conversations. Also avoid footnotes.

you have time, read the IEEE Style Sheet in the same folder with this sample.

Explain clearly what your results mean. If you have a result that makes no sense, clearly state why it makes no sense. If you can, give a plausible explanation as to what may have cause the strange result. Do not try to hide a bad result.

# Conclusions

Here you answer any questions raised in the previous sections. You should end with a wrap–up and firm statement of what a great accomplishment this is.

# References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein., *Introduction to Algorithms, 3rd Edition*. MIT press Cambridge, 2001.

# Appendix

An Appendix starts on a new page after everything else. This is were your code will be.

JAVA Source Code: **fragment.java**

/ / Sample r o u t i n e wi t h a s s e r t s t a t e m e n t s p u b l i c v o i d Build Max Heap(*<*T*>* A [ ] )

*{*

/ / Author : G Howser

/ / Purpose : B u i l d s a Max Heap i n A[ ] wi t h r o o t A[ 1 ]

/ / Heap Test : R e t u r n s ” t r u e ” i f f A[ j ] Max Heap f o r j , j + 1 , j + 2 , . . . n

/ /

/ / Pre c o n d i t i o n : A[ ] i s a non*−*empty a r r a y of comparable o b j e c t s

/ / I n v a r i a n t : A[ j ] i s r o o t of a Max Heap f o r j = i + 1 , i + 2 , . . . n i f ( debug )

*{*

a s s e r t ( ( ( A[ 1 ] *>* 0 ) *| |* (A[ 1 ] *<*= 0 ) ) ) ;

*}*;

i n t h e a p s i z e ;

i n t n = A. l e n g t h *−* 1 ;

h e a p s i z e = A. l e n g t h *−* 1 ; / / NOTE: z e r o r e l a t i v e a r r a y s b u t we a r e i g n o r i n g A[ 0 ] f o r ( i n t i = Math . f l o o r ( ( n / 2 . 0 ) ) ; i *−−*; i *<* 1 )

*{*

Max Heapify ( A, i ) ; i f ( debug )

*{*

a s s e r t ( ( Heap Test (A[ ] , i ) ) ;

*}*;

/ / P o s t c o n d i t i o n : A[ ] i s a Max Heap i f ( debug )

*{*

a s s e r t ( ( Heap Test (A[ ] , 1 ) ) ;

*}*;

*}*

JAVA Source Code: **fragment2.java**

/ / Sample r o u t i n e wi t h a s s e r t s t a t e m e n t s a n d a method t o e v a l u a t e an I n v a r i a n t p u b l i c v o i d Build Max Heap 2 (A [ ] )

*{*

/ / Author : G Howser

/ / Purpose : B u i l d s a Max Heap i n A[ ] wi t h r o o t A[ 1 ]

/ / Heap Test : R e t u r n s ” t r u e ” i f f A[ j ] Max Heap f o r j , j + 1 , j + 2 , . . . n

/ /

/ / Pre c o n d i t i o n : A[ ] i s a non*−*empty a r r a y of comparable o b j e c t s

/ / I n v a r i a n t : A[ j ] i s r o o t of a Max Heap f o r j = i + 1 , i + 2 , . . . n i f ( debug )

*{*

a s s e r t ( ( ( A[ 1 ] *>* 0 ) *| |* (A[ 1 ] *<*= 0 ) ) ) ;

*}*;

i n t h e a p s i z e ;

i n t n = A. l e n g t h *−* 1 ;

h e a p s i z e = A. l e n g t h *−* 1 ; / / NOTE: z e r o r e l a t i v e a r r a y s b u t we a r e i g n o r i n g A[ 0 ] f o r ( i n t i = Math . f l o o r ( ( n / 2 . 0 ) ) ; i *−−*; i *<* 1 )

*{*

Max Heapify ( A, i ) ; i f ( debug )

*{*

a s s e r t ( ( Heap Test (A[ ] , i , h e a p s i z e ) ) ;

*}*;

/ / P o s t c o n d i t i o n : A[ ] i s a Max Heap i f ( debug )

*{*

a s s e r t ( ( Heap Test (A[ ] , 1 , h e a p s i z e ) ) ;

*}*;

*}*

p r i v a t e boo l Heap Test (A[ ] , i n t i , i n t h S i z e )

*{*

/ / Author : G Howser

/ / Purpose : I n v a r i a n t f o r v a r i o u s r o u t i n e s i n Heap S o r t

/ / Heap Test : R e t u r n s ” t r u e ” i f f A[ j ] Max Heap f o r j , j + 1 , j + 2 , . . . n

/ /

/ / Pre c o n d i t i o n : A[ ] i s a non*−*empty a r r a y of i n t e g e r s

/ / P o s t c o n d i t i o n : R e t u r n s t r u e i f and on l y i f A[ 1 . . . h S i z e ] i s a heap i f ( debug )*{*

a s s e r t ( ( ( A[ 1 ] *>* 0 ) *| |* (A[ 1 ] *<* = 0 ) ) ) ; *}* ;

bo o l r e s u l t = t r u e ;

i n t h e a p s i z e = h S i z e ; i n t n = A. l e n g t h *−* 1 ;

/ / Code needs t o be w r i t t e n , so f o r now a lways r e t u r n t r u e i f ( debug )

*{*System . o u t . p r i n t l n ( ”TODO: Code need t o be w r i t t e n *}*; r e t u r n r e s u l t ;

*}*