Final Paper

An Investigation into Multi-Pivot Quicksort Algorithms

Authors:
Paymahn Moghadasian [umpaymah@myumanitoba.ca]

Joshua Hernandez [umhern23@myumanitoba.ca]

Instructor: Dr. S. Durocher

COMP 4420

Advanced Design and Analysis of Algorithms

Due : April 9, 2014

Contents

1 Abstract 2

2 Introduction 2

3 Quicksort 3

3.1 Classic Quicksort 3
3.2 Dual Pivot Quicksort 4
3.3 Yaroslavskiy Quicksort 4
3.4 Optimal Dual Pivot Quicksort 4
3.5 Three Pivot Quicksort 4
3.6 M Pivot Quicksort 4
3.6.1 Testing 4
3.7 Summary 4

4 Analysis 6

4.1 Data Processing 7
4.2 Non-Linear Curve Fit 7

5 Discussion 8

5.1 $n \log(n)$ Trend 8
5.2 Comparisons among Quicksorts 8
5.3 Fit Functions 8
5.4 Future Work 8

6 Conclusion 8

7 Fit Coefficients 8

8 Misc Plots 8

References 15

A GitHub Repository 17

References

[1] C. A. R. Hoare, "Quicksort," *The Computer Journal*, vol. 5, no. 1, pp. 10–16, 1962. 2, 3

[2] S. Kushagra, A. López-Ortiz, A. Qiao, and J. I. Munro, "Multi-pivot quicksort: Theory and experiments," 2013. 2, 3

[3] R. Sedgewick and K. Wayne, "Advanced topics in sorting," 2007. 2

[4] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd ed., 2001. 2

[5] D. E. Knuth, *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. 2

[6] M. H. van Emden, "Increasing the efficiency of quicksort," *Communications of the ACM*, vol. 13, no. 9, pp. 563–567, 1970. 2

[7] M. Foley and C. A. R. Hoare, "Proof of a recursive program: Quicksort," *The Computer Journal*, vol. 14, no. 4, pp. 391–395, 1971. 2

[8] R. Sedgewick, "The analysis of quicksort programs," *Acta Informatica*, vol. 7, no. 4, pp. 327–355, 1977. 2

[9] R. Sedgewick, "Implementing quicksort programs," *Communications of the ACM*, vol. 21, no. 10, pp. 847–857, 1978. 3

[10] D. Cederman and P. Tsigas, "Gpu-quicksort: A practical quicksort algorithm for graphics processors," *Journal of Experimental Algorithmics (JEA)*, vol. 14, p. 4, 2009. 3

[11] C. Martínez and S. Roura, "Optimal sampling strategies in quicksort and quickselect," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 683–705, 2001. 3

[12] J. R. Edmondson, "M pivot sort-replacing quick sort.," in *AMCS*, pp. 47–53, 2005. 3

[13] V. Yaroslavskiy, "Dual-pivot quicksort," *Research Disclosure RD539015 (Sept., 2009), http://iaroslavski. narod. ru/quicksort/DualPivotQuicksort. pdf*. 3

[14] M. Aumüller and M. Dietzfelbinger, "Optimal partitioning for dual pivot quicksort," in *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'13, (Berlin, Heidelberg), pp. 33–44, Springer-Verlag, 2013. 3

[15] C. R. Cook and D. J. Kim, "Best sorting algorithm for nearly sorted lists," *Communications of the ACM*, vol. 23, no. 11, pp. 620–624, 1980. 4

[16] P. Bevington, *Data reduction and error analysis for the physical sciences.* McGraw-Hill, 1969. 8

1 Abstract

The quicksort is a thoroughly studied sorting algorithm and is commonly among the first efficient sorts learned by students of computer science. Many variants of the quicksort have been proposed, from the classic quicksort introduced by Tony Hoare in 1961 [1] to Yaroslavskiy's dual pivot quicksort introduced in 2009 and used by the Java 7 Standard Library [2]. Since Hoare's first proposal, much research has gone into attempting to minimize the total number of swaps done by the sort, the total number of comparisons done by the sort and minimizing the worst case runtime. We aim to experimentally validate the swap and comparison count of several variants of the quicksort and compare the runtimes and various optimizations. Our results SUMMARIZE THE RESULTS

2 Introduction

Sorting is a fundamental concept of computer science wherein a totally ordered multiset is modified such that the elements of the multiset are rearranged (permuted) in either non-decreasing or non-increasing order. A broad range of applications benefit from sorting such as organizing an MP3 library by song title to quickly identifying duplicates in a list to more advanced applications such as load balancing, data compression and computer graphics [3]. It is well known that all comparison based sorting algorithms are lower bound by $\Omega(nlogn)$ comparisons [4] and quicksort is no exception to this rule. Interestingly, there are non-comparison based sorts such as the counting sort and the radix sort which take advantage of certain properties of the data set and get around the lower bound of comparison base sorts. A summary of space and time complexities can be found in Table 1.

The quicksort was first introduced by Tony Hoare in [1] and was quickly adopted by the field as a standard sorting algorithm alongside the mergesort [5] and several other sorting algorithms. Quicksorts simplicity, high average case performance, poor worst case performance and low use of additional memory has made it one of the most highly studied "efficient" sorting algorithms. van Emden [6] proposed an optimization to the quicksort in 1970 which resulted in a 15% improvement on the efficiency of the quicksort. In 1971 Hoare and Foley [7] provided a formal proof of other correctness of the quicksort. In 1977, Sedgewick did a thorough analysis of quicksort [8] and then in 1978 Sedgewick again published a paper on the quicksort exploring various

Sort Method	Space	Average Case Time	Worst Case Time
Selection	$O(1)$	$O(n^2)$	$O(n^2)$
Insertion	$O(1)$	$O(n^2)$	$O(n^2)$
Merge	$O(n)$	$O(n\log(n))$	$O(n\log(n))$
Quicksort	$O(1)$	$O(n\log(n))$	$O(n^2)$
Radix	$O(n)$	$O(n*k)$	$O(n*k)$
Counting	$O(m)$	$O(n+m)$	$O(n+m)$

Table 1: Summary of space and time complexities of various sorts where n represents the number of elements, k represents the number of digits in the largest value and m represents the maximum value to be sorted.

implementations and proposing several optimizations [9]. The academic interest in the quicksort has not slowed since the late 1970s; papers regarding the quicksort continue to be published for adapting the quicksort to GPUs [10], providing optimal pivot selection [11] and using more than one pivot [12].

In this paper we explore, compare and contrast the classic implementation of the quicksort against various multipivot quicksort implementations and their respective optimizations. We are particularly interested in comparing the classic quicksort against the standard dual pivot quicksort, Yaroslavskiy's dual pivot quicksort[13], Aumüller and Dietzfelbinger's optimized dual-pivot quicksort [14], Edmonson's M-Pivot quicksort [12] and the three pivot quicksort introduced by Kushagra et al [2]. We will first introduce the various quicksorts by providing a brief overview of their behaviour, implementation and potential optimizations. Next we will discuss the experiments and provide their results. Following this we provide an analysis, discuss future work and conclude the paper.

3 Quicksort

The following descriptions will assume that the algorithms are being used to sort a list A in non-decreasing order where n denotes the number of elements in A. Additionally, all algorithms are assumed to perform the sort in place.

3.1 Classic Quicksort

The classic quicksort, first introduced by Hoare [1] in 1962 is a recursive sorting algorithm which falls into the category of divide and conquer algorithms. All of following variations of the quicksort have these qualities as well. Quicksort can be summarized as follows:

1. Select a pivot in the current range of elements to be sorted.

2. Partition the list such that all elements less than the pivot appear to the left of the pivot while all elements greater than or equal to the pivot appear to the right of the pivot.

3. Recursively call quicksort on the left and right halves of the currently partitioned range.

This naïve implementation does not handle it's worst case gracefully. The classic quicksort degrades to a time complexity of $O\left(n^2\right)$ when the data being processed is in nearly sorted or already sorted order. Due to the simplicity of this implementation there are some minor changes which can be made to speed up the average case and aid the worst case runtimes. Two simple improvements which can be done are: perform an insertion sort when the range to be sorted meets some threshold and select the pivot more intelligently. It was shown by Cook and Kim [15] that insertion sorts are the fastest sort for small lists and nearly sorted lists. Cook and Kim also showed that a hybrid of insertion sort and quicksort outperforms either of the sorts used individually. A perfectly selected pivot (the median of the range) would ensure that the worst case space complexity is bound by $O\left(n\log(n)\right)$. Perfect pivot selection is not possible; however, selecting pivot to be the median of any three elements in the range greatly reduces the probability of a worst-case scenario arising.

3.2 Dual Pivot Quicksort

3.3 Yaroslavskiy Quicksort

3.4 Optimal Dual Pivot Quicksort

3.5 Three Pivot Quicksort

3.6 M Pivot Quicksort

RAWR

3.6.1 Testing

Just to test the subsection code. Test text : Recall from section 3.1 on page 3

3.7 Summary

References

[1] C. A. R. Hoare, "Quicksort," *The Computer Journal*, vol. 5, no. 1, pp. 10–16, 1962. 2, 3

Sort Method	Comparisons
Classic	$2n \log n - 1.51n + O(\log(n))$
Dual Pivot	$2.13n \log n - 2.57n + O(\log(n))$
Optimal Dual Pivot	$1.8n \log n + O(n)$
Three Pivot	$1.846n \log n + O(n)$
Yaroslavskiy	$1.9n \log n - 2.46n + O(\log(n))$
M Pivot	$O(n \log n)$

Table 2: Summary table of theoretical comparisons.

Sort Method	Swaps
Classic	$0.33n \log n - 0.58n + O(\log(n))$
Dual Pivot	$0.8n \log n - 0.3n + O(\log(n))$
Optimal Dual Pivot	$0.33n \log n + O(n)$
Three Pivot	$0.615n \log n + O(n)$
Yaroslavskiy	$0.6n \log n + 0.08n + O(\log(n))$
M Pivot	$O(n \log n)$

Table 3: Summary table of theoretical swaps.

[2] S. Kushagra, A. López-Ortiz, A. Qiao, and J. I. Munro, "Multi-pivot quicksort: Theory and experiments," 2013. 2, 3

[3] R. Sedgewick and K. Wayne, "Advanced topics in sorting," 2007. 2

[4] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd ed., 2001. 2

[5] D. E. Knuth, *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. 2

[6] M. H. van Emden, "Increasing the efficiency of quicksort," *Communications of the ACM*, vol. 13, no. 9, pp. 563–567, 1970. 2

[7] M. Foley and C. A. R. Hoare, "Proof of a recursive program: Quicksort," *The Computer Journal*, vol. 14, no. 4, pp. 391–395, 1971. 2

[8] R. Sedgewick, "The analysis of quicksort programs," *Acta Informatica*, vol. 7, no. 4, pp. 327–355, 1977. 2

[9] R. Sedgewick, "Implementing quicksort programs," *Communications of the ACM*, vol. 21, no. 10, pp. 847–857, 1978. 3

Name of Sort Algorithm	Pivot Selection Methods	Number of Pivots
Classic Quicksort	3	1
Dual Pivot Quicksort	2	2
Heap Optimized M-Pivot Quicksort	1	3,4,5,6
M-Pivot Quicksort	1	3,4,5,6
Optimal Dual Pivot Quicksort	2	2
Three Pivot Quicksort	1	3
Yaroslavskiy Quicksort	1	2

Table 4: List of all the variation of quicksorts executed.

[10] D. Cederman and P. Tsigas, "Gpu-quicksort: A practical quicksort algorithm for graphics processors," *Journal of Experimental Algorithmics (JEA)*, vol. 14, p. 4, 2009. 3

[11] C. Martínez and S. Roura, "Optimal sampling strategies in quicksort and quickselect," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 683–705, 2001. 3

[12] J. R. Edmondson, "M pivot sort-replacing quick sort.," in *AMCS*, pp. 47–53, 2005. 3

[13] V. Yaroslavskiy, "Dual-pivot quicksort," *Research Disclosure RD539015 (Sept., 2009), http://iaroslavski. narod. ru/quicksort/DualPivotQuicksort. pdf.* 3

[14] M. Aumüller and M. Dietzfelbinger, "Optimal partitioning for dual pivot quicksort," in *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'13, (Berlin, Heidelberg), pp. 33–44, Springer-Verlag, 2013. 3

[15] C. R. Cook and D. J. Kim, "Best sorting algorithm for nearly sorted lists," *Communications of the ACM*, vol. 23, no. 11, pp. 620–624, 1980. 4

[16] P. Bevington, *Data reduction and error analysis for the physical sciences.* McGraw-Hill, 1969. 8

4 Analysis

So we have established the details of the differences between the different variations of quicksorts. To investigate and compare the algorithms we have implemented and ran experiments. We used uniformly random numbers from 0 to 10^{12}. Then created arrays with randomly generated integers for sizes 4 to 50 million Table 4 shows the variations of quicksort to the arrays generated.

Figure 4 is the legend of all the plots generated. You can see the color and shape combination for each version of quick sort algorithm. These color and shape choices will be consistent throughout the following plots. There are 17 distinct derivatives of quicksort that have been run. Note that for the plots we overlay the function :

$$A \cdot n\log(n) + B \cdot n + C\log(n) \tag{1}$$

The nomenclature for the legend is as follows:

(Algorithm Name, Pivot Selection Method Index, Number of Pivots, is Insertion Sort used).

`../../Code/Analysis/PlotLegend.png`

Figure 1: A visual reference for all the quicksort variations and plots.

4.1 Data Processing

In processing the data, we take averages of any common data entries. We plot them and have the following results. Which is summarized in Figures 2, 4, 5, and 6.

4.2 Non-Linear Curve Fit

With the data that has collected we wish to be able to look at a whole data set and quantitatively compare to other data sets. From our introduction of the different quicksort algorithms, the asymptotic runtime of all of the had one of the following forms:

$$A \cdot n \log(n) + B \cdot n + O\left(\log(n)\right) \tag{2}$$

$$A \cdot n \log(n) + O\left(n\right) \tag{3}$$

$$O\left(n \log(n)\right) \tag{4}$$

In correspondence of this we will fit the data to the following equation:

$$A \cdot n \log(n) + B \cdot n + C \log(n) \tag{5}$$

With the function parameters A, B and C where found using a non-linear curve fitting tool in the SciPy module in python. Qualitatively, the fit function that was picked seems to result in good fits visually. Though the fit has less meaning do to the fact we don't have error bars on the data. A closer look at the reduced chi squared values of the fits leads us to believe that the fits are not good.[16] Thus we will treat the results from the best fit as qualitative.

5 Discussion

5.1 $n \log(n)$ Trend

Preliminary observations on Figure 2 show that the optimal dual pivot quicksort had the lowest number of comparisons. This is as expected since it was designed to minimize the number of comparisons.

5.2 Comparisons among Quicksorts

5.3 Fit Functions

5.4 Future Work

Read the comments in the .tex file

6 Conclusion

We did things and stuff!

7 Fit Coefficients

8 Misc Plots

Sort Method	$A_{\text{comparisons}}$
Classic Quicksort - 1 - 1	0.02151 ± 0.00019
Classic Quicksort - 2 - 1	0.02135 ± 0.00017
Classic Quicksort - 3 - 1	0.01807 ± 0.00006
DualPivot Quicksort - 1 - 2	0.02014 ± 0.00018
DualPivot Quicksort - 2 - 2	0.01772 ± 0.00007
Heap Optimized M-Pivot Quicksort - 1 - 3	0.02778 ± 0.00014
Heap Optimized M-Pivot Quicksort - 1 - 4	0.02788 ± 0.00015
Heap Optimized M-Pivot Quicksort - 1 - 5	0.02842 ± 0.00025
Heap Optimized M-Pivot Quicksort - 1 - 6	0.02869 ± 0.00011
M-Pivot Quicksort - 1 - 3	0.01970 ± 0.00009
M-Pivot Quicksort - 1 - 4	0.02076 ± 0.00015
M-Pivot Quicksort - 1 - 5	0.02165 ± 0.00010
M-Pivot Quicksort - 1 - 6	0.02386 ± 0.00014
Optimal Dual Pivot Quicksort - 1 - 2	0.01959 ± 0.00019
Optimal Dual Pivot Quicksort - 2 - 2	0.01744 ± 0.00007
Three Pivot Quicksort - 1 - 3	0.02587 ± 0.00009
Yaroslavskiy Quicksort - 1 - 2	0.01796 ± 0.00010

Table 5: Summary table coefficients of the non-linear fit for the parameter A on the comparison data.

Sort Method	$B_{\text{comparisons}}$
Classic Quicksort - 1 - 1	-0.05025 ± 0.00469
Classic Quicksort - 2 - 1	-0.04634 ± 0.00428
Classic Quicksort - 3 - 1	-0.02126 ± 0.00163
DualPivot Quicksort - 1 - 2	-0.03650 ± 0.00465
DualPivot Quicksort - 2 - 2	-0.01045 ± 0.00183
Heap Optimized M-Pivot Quicksort - 1 - 3	-0.05055 ± 0.00355
Heap Optimized M-Pivot Quicksort - 1 - 4	-0.05152 ± 0.00368
Heap Optimized M-Pivot Quicksort - 1 - 5	-0.04639 ± 0.00626
Heap Optimized M-Pivot Quicksort - 1 - 6	-0.03889 ± 0.00280
M-Pivot Quicksort - 1 - 3	-0.01917 ± 0.00220
M-Pivot Quicksort - 1 - 4	-0.01716 ± 0.00391
M-Pivot Quicksort - 1 - 5	-0.00902 ± 0.00244
M-Pivot Quicksort - 1 - 6	-0.03206 ± 0.00366
Optimal Dual Pivot Quicksort - 1 - 2	-0.03580 ± 0.00490
Optimal Dual Pivot Quicksort - 2 - 2	-0.01266 ± 0.00165
Three Pivot Quicksort - 1 - 3	-0.04282 ± 0.00232
Yaroslavskiy Quicksort - 1 - 2	-0.01636 ± 0.00251

Table 6: Summary table coefficients of the non-linear fit for the parameter B on the comparison data.

Sort Method	$C_{\text{comparisons}}$
Classic Quicksort - 1 - 1	121.34341 ± 99.97322
Classic Quicksort - 2 - 1	78.33233 ± 91.31275
Classic Quicksort - 3 - 1	22.52742 ± 34.75034
DualPivot Quicksort - 1 - 2	52.22569 ± 99.17037
DualPivot Quicksort - 2 - 2	-53.25104 ± 39.07336
Heap Optimized M-Pivot Quicksort - 1 - 3	94.09226 ± 75.71998
Heap Optimized M-Pivot Quicksort - 1 - 4	124.92512 ± 78.46872
Heap Optimized M-Pivot Quicksort - 1 - 5	56.97527 ± 133.52044
Heap Optimized M-Pivot Quicksort - 1 - 6	29.84293 ± 59.67899
M-Pivot Quicksort - 1 - 3	51.60730 ± 46.87649
M-Pivot Quicksort - 1 - 4	49.35746 ± 83.31083
M-Pivot Quicksort - 1 - 5	4.76647 ± 52.03322
M-Pivot Quicksort - 1 - 6	136.38815 ± 77.96382
Optimal Dual Pivot Quicksort - 1 - 2	56.95127 ± 104.39452
Optimal Dual Pivot Quicksort - 2 - 2	-26.20004 ± 35.22097
Three Pivot Quicksort - 1 - 3	-17.79746 ± 49.38183
Yaroslavskiy Quicksort - 1 - 2	0.73204 ± 53.59187

Table 7: Summary table coefficients of the non-linear fit for the parameter C on the comparison data.

Sort Method	A_{swap}
Classic Quicksort - 1 - 1	0.01026 ± 0.00017
Classic Quicksort - 2 - 1	0.01095 ± 0.00016
Classic Quicksort - 3 - 1	0.00848 ± 0.00012
DualPivot Quicksort - 1 - 2	0.00629 ± 0.00010
DualPivot Quicksort - 2 - 2	0.00606 ± 0.00006
Heap Optimized M-Pivot Quicksort - 1 - 3	0.01004 ± 0.00009
Heap Optimized M-Pivot Quicksort - 1 - 4	0.00898 ± 0.00004
Heap Optimized M-Pivot Quicksort - 1 - 5	0.00809 ± 0.00004
Heap Optimized M-Pivot Quicksort - 1 - 6	0.00759 ± 0.00005
M-Pivot Quicksort - 1 - 3	0.00672 ± 0.00006
M-Pivot Quicksort - 1 - 4	0.00605 ± 0.00003
M-Pivot Quicksort - 1 - 5	0.00535 ± 0.00003
M-Pivot Quicksort - 1 - 6	0.00513 ± 0.00003
Optimal Dual Pivot Quicksort - 1 - 2	0.00629 ± 0.00010
Optimal Dual Pivot Quicksort - 2 - 2	0.00606 ± 0.00006
Three Pivot Quicksort - 1 - 3	0.00635 ± 0.00006
Yaroslavskiy Quicksort - 1 - 2	0.00586 ± 0.00005

Table 8: Summary table coefficients of the non-linear fit for the parameter A on the swap data.

Sort Method	B_{swap}
Classic Quicksort - 1 - 1	-0.00132 ± 0.00442
Classic Quicksort - 2 - 1	-0.01411 ± 0.00417
Classic Quicksort - 3 - 1	0.01903 ± 0.00298
DualPivot Quicksort - 1 - 2	0.00824 ± 0.00264
DualPivot Quicksort - 2 - 2	0.01080 ± 0.00153
Heap Optimized M-Pivot Quicksort - 1 - 3	0.00774 ± 0.00233
Heap Optimized M-Pivot Quicksort - 1 - 4	0.01111 ± 0.00097
Heap Optimized M-Pivot Quicksort - 1 - 5	0.01881 ± 0.00113
Heap Optimized M-Pivot Quicksort - 1 - 6	0.02363 ± 0.00129
M-Pivot Quicksort - 1 - 3	0.01160 ± 0.00156
M-Pivot Quicksort - 1 - 4	0.01890 ± 0.00069
M-Pivot Quicksort - 1 - 5	0.03171 ± 0.00065
M-Pivot Quicksort - 1 - 6	0.03601 ± 0.00077
Optimal Dual Pivot Quicksort - 1 - 2	0.00824 ± 0.00264
Optimal Dual Pivot Quicksort - 2 - 2	0.01080 ± 0.00153
Three Pivot Quicksort - 1 - 3	0.01030 ± 0.00153
Yaroslavskiy Quicksort - 1 - 2	0.01592 ± 0.00127

Table 9: Summary table coefficients of the non-linear fit for the parameter B on the swap data.

Sort Method	C_{swap}
Classic Quicksort - 1 - 1	-36.52075 ± 94.28550
Classic Quicksort - 2 - 1	102.34036 ± 88.94077
Classic Quicksort - 3 - 1	-103.21696 ± 63.57056
DualPivot Quicksort - 1 - 2	-38.02577 ± 56.38166
DualPivot Quicksort - 2 - 2	42.34411 ± 32.52471
Heap Optimized M-Pivot Quicksort - 1 - 3	-53.81114 ± 49.78070
Heap Optimized M-Pivot Quicksort - 1 - 4	-3.24127 ± 20.75454
Heap Optimized M-Pivot Quicksort - 1 - 5	-16.07787 ± 24.00667
Heap Optimized M-Pivot Quicksort - 1 - 6	-20.99394 ± 27.47669
M-Pivot Quicksort - 1 - 3	41.76450 ± 33.26955
M-Pivot Quicksort - 1 - 4	16.90493 ± 14.70501
M-Pivot Quicksort - 1 - 5	-17.36329 ± 13.95680
M-Pivot Quicksort - 1 - 6	-5.54593 ± 16.46902
Optimal Dual Pivot Quicksort - 1 - 2	-38.02577 ± 56.38166
Optimal Dual Pivot Quicksort - 2 - 2	42.34411 ± 32.52471
Three Pivot Quicksort - 1 - 3	14.68107 ± 32.72260
Yaroslavskiy Quicksort - 1 - 2	6.27071 ± 27.13874

Table 10: Summary table coefficients of the non-linear fit for the parameter C on the swap data.

References

[1] C. A. R. Hoare, “Quicksort,” *The Computer Journal*, vol. 5, no. 1, pp. 10–16, 1962. 2, 3

[2] S. Kushagra, A. López-Ortiz, A. Qiao, and J. I. Munro, “Multi-pivot quicksort: Theory and experiments,” 2013. 2, 3

[3] R. Sedgewick and K. Wayne, “Advanced topics in sorting,” 2007. 2

[4] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd ed., 2001. 2

[5] D. E. Knuth, *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching.* Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. 2

[6] M. H. van Emden, “Increasing the efficiency of quicksort,” *Communications of the ACM*, vol. 13, no. 9, pp. 563–567, 1970. 2

[7] M. Foley and C. A. R. Hoare, “Proof of a recursive program: Quicksort,” *The Computer Journal*, vol. 14, no. 4, pp. 391–395, 1971. 2

[8] R. Sedgewick, “The analysis of quicksort programs,” *Acta Informatica*, vol. 7, no. 4, pp. 327–355, 1977. 2

[9] R. Sedgewick, “Implementing quicksort programs,” *Communications of the ACM*, vol. 21, no. 10, pp. 847–857, 1978. 3

[10] D. Cederman and P. Tsigas, “Gpu-quicksort: A practical quicksort algorithm for graphics processors,” *Journal of Experimental Algorithmics (JEA)*, vol. 14, p. 4, 2009. 3

[11] C. Martínez and S. Roura, “Optimal sampling strategies in quicksort and quickselect,” *SIAM Journal on Computing*, vol. 31, no. 3, pp. 683–705, 2001. 3

[12] J. R. Edmondson, “M pivot sort-replacing quick sort.,” in *AMCS*, pp. 47–53, 2005. 3

[13] V. Yaroslavskiy, “Dual-pivot quicksort,” *Research Disclosure RD539015 (Sept., 2009), http://iaroslavski. narod. ru/quicksort/DualPivotQuicksort. pdf.* 3

[14] M. Aumüller and M. Dietzfelbinger, “Optimal partitioning for dual pivot quicksort,” in *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part I*, ICALP’13, (Berlin, Heidelberg), pp. 33–44, Springer-Verlag, 2013. 3

[15] C. R. Cook and D. J. Kim, "Best sorting algorithm for nearly sorted lists," *Communications of the ACM*, vol. 23, no. 11, pp. 620–624, 1980. 4

[16] P. Bevington, *Data reduction and error analysis for the physical sciences.* McGraw-Hill, 1969. 8

References

[1] C. A. R. Hoare, "Quicksort," *The Computer Journal*, vol. 5, no. 1, pp. 10–16, 1962. 2, 3

[2] S. Kushagra, A. López-Ortiz, A. Qiao, and J. I. Munro, "Multi-pivot quicksort: Theory and experiments," 2013. 2, 3

[3] R. Sedgewick and K. Wayne, "Advanced topics in sorting," 2007. 2

[4] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd ed., 2001. 2

[5] D. E. Knuth, *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching.* Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. 2

[6] M. H. van Emden, "Increasing the efficiency of quicksort," *Communications of the ACM*, vol. 13, no. 9, pp. 563–567, 1970. 2

[7] M. Foley and C. A. R. Hoare, "Proof of a recursive program: Quicksort," *The Computer Journal*, vol. 14, no. 4, pp. 391–395, 1971. 2

[8] R. Sedgewick, "The analysis of quicksort programs," *Acta Informatica*, vol. 7, no. 4, pp. 327–355, 1977. 2

[9] R. Sedgewick, "Implementing quicksort programs," *Communications of the ACM*, vol. 21, no. 10, pp. 847–857, 1978. 3

[10] D. Cederman and P. Tsigas, "Gpu-quicksort: A practical quicksort algorithm for graphics processors," *Journal of Experimental Algorithmics (JEA)*, vol. 14, p. 4, 2009. 3

[11] C. Martínez and S. Roura, "Optimal sampling strategies in quicksort and quickselect," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 683–705, 2001. 3

[12] J. R. Edmondson, "M pivot sort-replacing quick sort.," in *AMCS*, pp. 47–53, 2005. 3

[13] V. Yaroslavskiy, “Dual-pivot quicksort,” *Research Disclosure RD539015 (Sept., 2009), http://iaroslavski. narod. ru/quicksort/DualPivotQuicksort. pdf.* 3

[14] M. Aumüller and M. Dietzfelbinger, “Optimal partitioning for dual pivot quicksort,” in *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part I*, ICALP’13, (Berlin, Heidelberg), pp. 33–44, Springer-Verlag, 2013. 3

[15] C. R. Cook and D. J. Kim, “Best sorting algorithm for nearly sorted lists,” *Communications of the ACM*, vol. 23, no. 11, pp. 620–624, 1980. 4

[16] P. Bevington, *Data reduction and error analysis for the physical sciences.* McGraw-Hill, 1969. 8

A GitHub Repository

Project GitHub Repository Link

../../Code/Analysis/AllthePlotsLargeScale_comp.png

../../Code/Analysis/AllthePlotsLargeScale_swap.png

Figure 2: A plot of the data from all the sorting algorithm against the number of comparisons.

../../Code/Analysis/AllPlotsLargeScalelognvsy_OVER_nlogn_comp.png

../../Code/Analysis/AllPlotsLargeScalelognvsy_OVER_nlogn_swap.png

Figure 3: A plot of the data from all the sorting algorithm

../../Code/Analysis/TwoPivotsLargeScale_comp.png

../../Code/Analysis/TwoPivotsLargeScale_swap.png

Figure 4: A plot of the data from all the sorting algorithm with two pivots

../../Code/Analysis/ThreePivotsLargeScale_comp.png

../../Code/Analysis/ThreePivotsLargeScale_swap.png

Figure 5: A plot of the data from all the sorting algorithm with three pivots

../../Code/Analysis/M-PivotQuicksortsSmallScale_comp.png

../../Code/Analysis/M-PivotQuicksortsSmallScale_swap.png

Figure 6: Data from all the version of M-Pivot Sort.