Bibliography

[1] OCaml. 99 Problems [Internet]. Rocquencourt, France: OCaml; 2019 [cited 2020 January 11]. Available from: https://ocaml.org/learn/tutorials/99problems.html

[2] Agarwal A, Cruanes S, Monnier F, Scherer G, Grinberg R, Friendly E. OCaml Batteries Included, 2.9.0 [programming language library]. San Francisco (CA), USA: Github; 2018 [cited 2020 May 10]. Available from: https://github.com/ocaml-batteries-team/batteries-included

[3] Appel AW. Efficient Verified Red-Black Trees [Internet]. Princeton (NJ), USA: Princeton University; 2011 [cited 2019 February 25]. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.298.7387&rep=rep1&type=pdf.

[4] Bagwell P. Fast Functional Lists, Hash-Lists, Deques and Variable Length Arrays [Internet]. Lausanne, Switzerland: Infoscience: EPF; 2002 [cited 2019 February 26]. Available from: https://infoscience.epfl.ch/record/52465.

[5] Basios M, Li L, Wu F, Kanthan L, Barr ET. Darwinian Data Structure Selection [Internet]. In: Leavens GT, Garcia A, Păsăreanu CS, editors. Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC FSE'18. 2018 November 4-9; Lake Buena Vista (FL), USA. New York (NY), USA: ACM; 2018 [cited 2021 03 13]. pages 118-129. Available from: https://dl.acm.org/citation.cfm?id=3236043.

[6] Bastani FB, Iyengar SS. The Effect of Data Structures on the Logical Complexity of Programs [Internet]. CACM. 1987 [cited 2019 March 1];30(3):pages 250-259. Available from: https://dl.acm.org/citation.cfm?id=214760.

[7] Biermann F, Sestoft P. Quad Ropes: Immutable, Declarative Arrays with Parallelizable Operations [Internet]. In: Elsman M, Grelck C, Kloeckner A, Padua D, Solomonik E, editors. Proceedings of the 4th ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming. ARRAY'17. 2017 June 18; Barcelona, Spain. New York (NY), USA: ACM; 2017 [cited 2021 March 13]. pages 1-8. Available from: http://dx.doi.org/10.1145/3091966.3091971.

[8] Bik AJC, Wijshoff HAG. Automatic Data Structure Selection and Transformation for Sparse Matrix Computations [Internet]. IEEE Transactions on Parallel and Distributed Systems. 1996 [cited 2019 March 1];7(2):pages 109-126. Available from: https://ieeexplore.ieee.org/abstract/document/485501.

[9] Booth TL, Wiecek CA. Performance Abstract Data Types as a Tool in Software Performance Analysis and Design [Internet]. IEEE Transactions on Software Engineering. 1980 [cited 2019 February 26];6(2):pages 138-151. Available from: https://ieeexplore.ieee.org/abstract/document/1702711.

[10] Brain R. Dynamic Code Analysis vs. Static Analysis Source Code Testing [Internet]. London, UK: Computer Weekly; 2010 [cited 2019 March 27]. Available from: https://www.computerweekly.com/answer/Dynamic-code-analysis-vs-static-analysis-source-code-testing.

[11] Brandauer S. Structured Data [dissertation on the Internet]. Uppsala, Sweden: Acta Universitatis Upsaliensis; 2018 [cited 2019 February 15]. Available from: http://uu.diva-portal.org/smash/get/diva2:1266019/FULLTEXT01.pdf.

[12] Brandauer S, Castegren E, Wrigstad T. Cb: A New Modular Approach to Implementing Efficient and Tunable Collections [Internet]. In: Gonzalez Boix E, Gabriel RP, editors. Proceedings of the 2018 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. Onward!'18. 2018 November 7-8; Boston (MA), USA. New York (NY), USA: ACM; [Preprint] 2018 [cited 2018 January 28]. pages 57-71. Available from: https://2018.onward-conference.org/event/

onward-2018-papers-cflat-a-new-modular-approach-to-implementing-efficient-and-tunable-collections

[13] Brodal GS, Makris C, Tsichlas K. Purely Functional Worst Case Constant Time Catenable Sorted Lists [Internet]. In: Azar Y, ErlebachT, editors. Lecture Notes in Computer Science, Algorithms – ESA 2006. ESA'2006. 2006 September 11-13; Zurich, Switzerland. Berlin, Germany: Springer Berlin Heidelberg; 2006 [cited 2021 March 13]. pages: 172-183. Available from: https://link.springer.com/chapter/10.1007/11841036_18.

[14] Brodie ML. Data abstraction, databases, and conceptual modelling: an annotated bibliography [Internet]. 1 ed. Baltimore (MD), USA: University of Maryland; 1980 [cited 2020 May 10]. Available from: https://books.google.se/books?id=rlLVmDy9hPkC&printsec=frontcover&source=gbs_book_other_versions_r&redir_esc=y#v=onepage&q&f=false

[15] Broman D. Miking, 2019 June 28th [interpreter and programming language]. Stockholm, Sweden: MCS (KTH Royal Institute of Technology); 2019 [cited 2019 June 28th]. Available from: https://github.com/miking-lang/miking

[16] Chung L. Non-Functional Requirements in Software Engineering [Internet]. 1 ed. Berlin, Germany: Springer Berlin Heidelberg; 2000 [cited 2020 May 10]. Available from: https://kth-primo.hosted.exlibrisgroup.com/permalink/f/qra184/46KTH_ALMA_DS51175259280002456

[17] Coleman D. A Structured Programming Approach to Data [Internet]. Berlin, Germany: Springer Berlin Heidelberg; 1979 [cited 2019 February 4]. Available from: https://www.springer.com/gp/book/9781468479874.

[18] Costa D, Andrzejak A. CollectionsSwitch: A Framework for Efficient and Dynamic Collection Selection [Internet]. In: Knoop J, Schordan M, Johnson T, O'Boyle M, editors. Proceedings of the 2018 International Symposium on Code Generation and Optimization. CGO 2018. 2018 February 24-28; Vienna, Austria. New York (NY), USA: ACM; 2018 [cited 2021 March 13]. pages 16-26. Available from: https://dl.acm.org/citation.cfm?id=3168825.

[19] Cousot P, Cousot R. Modular Static Program Analysis [Internet]. In: Horspool RN, editor. International Conference on Compiler Construction. CC'02. 2002 April 8-12; Grenoble, France. Berlin, Germany: Springer Berlin Heidelberg; 2002 [cited 2019 February 26]. pages: 159-179. Available from: https://link.springer.com/chapter/10.1007/3-540-45937-5_13.

[20] Danielsson NA. Lightweight Semiformal Time Complexity Analysis for Purely Functional Data Structures [Internet]. In: Wadler P, editor. Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. POPL'08. 2008 January 7; San Francisco (CA), USA. New York (NY), USA: ACM; 2008 [cited 2021 March 13]. pages 133-144. Available from: https://kth-primo.hosted.exlibrisgroup.com/permalink/f/1vh8pa3/TN_cdi_pascalfrancis_primary_20233711

[21] Danner N, Paykin J, Royer JS. A Static Cost Analysis for a Higher-order Language [Internet]. In: Brady EC, editor. Proceedings of the 7th workshop on programming languages meets program verification. PLPV'13. 2013 January 22; Rome, Italy. New York (NY), USA: ACM; 2013 [cited 2021 March 13]. pages 25-34. Available from: https://dl.acm.org/citation.cfm?id=2428123.

[22] De Wael M, Marr S, De Koster J, Sartor JB, De Meuter W. Just-in-Time Data Structures [Internet]. In: Murphy GC, Steele GLJ, editors. Proceedings of the 2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. Onward! 2015. 2015 October 25-30; Pittsburgh (PA), USA. New York (NY), USA: ACM; 2015 [cited 2021 March 13]. pages: 61-75. Available from: https://dl.acm.org/citation.cfm?id=2814231.

[23] De Wael M, Marr S, De Meuter W. Data Interface + Algorithms = Efficient Programs [Internet]. In: Proceedings of the 9th International Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems PLE. ICOOOLPS'14. 2014 July 28; Uppsala, Sweden. New York (NY), USA: ACM; 2014 [cited 2019 February 26]. pages 1-4. Available from: https://dl.acm.org/citation.cfm?id=2633303.

[24] Driscoll JR, Sarnak N, Sleator DD, Tarjan RE. Making Data Structures Persistent. Journal of Computer and System Sciences [Internet].

1989 [cited 2019 February 25];38(1):pages 86-124. Available from: https://www.sciencedirect.com/science/article/pii/0022000089900342.

[25] Eck DJ. Data Structures and Algorithms [Internet]. Geneva (NY), USA: Hobart and William Smith Colleges; 2004 [cited 2020 May 10]. Available from: http://math.hws.edu/eck/cs327_s04/

[26] Foo NY. Algebraic Specifications as Solutions of Implementation Equations [Internet]. IEEE Transactions on Software Engineering. 1987 [cited 2019 February 25];13(12):pages 1364-1369. Available from: https://ieeexplore.ieee.org/abstract/document/1702185.

[27] Germane K, Might M. Functional Pearl - Deletion: The Curse of the Red-Black Tree [Internet]. Journal of Functional Programming. 2014 [cited 2019 February 26];24(4):pages 423-433. Available from: https://www.cambridge.org/core/services/aop-cambridge-core/content/view/S0956796814000227.

[28] Gosain A, Sharma G. A Survey of Dynamic Program Analysis Techniques and Tools [Internet]. Advances in Intelligent Systems and Computing. 2014 [cited 2019 March 27];327:pages 113-122. Available from: https://www.researchgate.net/publication/286217492_A_Survey_of_Dynamic_Program_Analysis_Techniques_and_Tools.

[29] Hinze R, Paterson R. Finger Trees: A Simple General-Purpose Data Structure [Internet]. Journal of Functional Programming. 2006 [cited 2019 February 26];16(2):pages 297-217. Available from: https://www.cambridge.org/core/journals/journal-of-functional-programming/article/finger-trees-a-simple-generalpurpose-data-structure/BF419BCA07292DCAAF2A946E6BDF573B.

[30] INRIA. OCaml, 4.07.1 [programming language]. Rocquencourt, France: INRIA; 2020 [cited 2020 August 29th]. Available from: https://ocaml.org/

[31] Jordan W, Bejo A, Persada AG. The Development of Lexer and Parser as Parts of Compiler for GAMA32 Processor's Instruction-set using Python [Internet]. In: Wibowo FW, editor. 2019 International

Seminar on Research of Information Technology and Intelligent Systems. ISRITI 2019. 2019 December 5-6; Yogyakarta, Indonesia. Piscataway (NJ), USA: IEEE; 2019 [cited 2021 March 13]. pages: 450-455. Available from: https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=9034617

[32] Jung C, Rus S, Railing BP, Clark N, Pande S. Brainy: Effective Selection of Data Structures [Internet]. ACM SIGPLAN Notices. 2011 [cited 2019 February 26];46(6):pages 86-97. Available from: https://dl.acm.org/citation.cfm?id=1993509.

[33] Kahrs S. Red-Black Trees with Types [Internet]. Journal of Functional Programming. 2001 [cited 2019 February 26];11(4):pages 425-432. Available from: https://www.cambridge.org/core/journals/journal-of-functional-programming/article/redblack-trees-with-types/632BDF7BE8CD1F6EAEDEA37E6046E4A1.

[34] Kaplan H, Tarjan RE. Persistent Lists with Catenation via Recursive Slow-Down [Internet]. In: Leighton FT, Borodin A, editors. Proceedings of the twenty-seventh annual ACM symposium on Theory of computing. STOC'95. 1995 May 29-June 1; Las Vegas (NV), USA. New York (NY), USA: ACM; 1995 [cited 2019 February 26]. pages 93-102. Available from: https://dl.acm.org/citation.cfm?id=225090.

[35] Kaplan H, Tarjan RE. Purely Functional, Real-Time Deques with Catenation [Internet]. JACM. 1999 [cited 2019 February 25];46(5):pages 577-603. Available from: https://dl.acm.org/citation.cfm?id=324139.

[36] Kaplan H, Tarjan RE. Purely Functional Representations of Catenable Sorted Lists [Internet]. In: Miller GL, editor. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. STOC'96. 1996 May 22-24; Philadelphia (PA), USA. New York (NY), USA: ACM; 1996 [cited 2021 March 13]. pages 202-211. Available from: https://dl.acm.org/citation.cfm?id=237865.

[37] Kessel M, Atkinson C. Ranking Software Components for Reuse Based on Non-Functional Properties [Internet]. Information Systems Frontiers. 2016 [cited 2019 February 19];18(5):pages 825-853. Available from: https://link.springer.com/article/10.1007/s10796-016-9685-3.

[38] Kostyukov V. A Functional Approach to Standard Binary Heaps [Internet]. Ithaca (NY), USA: Cornell University. [Preprint] 2013 [cited 2019 February 25]. Available from: https://arxiv.org/abs/1312.4666.

[39] Li B, Liu J, Zhao J. Code Generation for Abstract Data Types Based on Program Analysis [Internet]. In. Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security - Companion. QRS-C'15. 2015 August 3-5; Vancouver, Canada. Washington (DC), USA: IEEE Computer Society; 2015 [cited 2019 February 26]. pages 153-160. Available from: https://dl.acm.org/citation.cfm?id=2866697.

[40] Linux. perf, 5.4.g219d54332a09 [computer program]. San Francisco (CA), USA: Linux; 2009 [cited 2020 January 11]. Available from: https://perf.wiki.kernel.org/index.php/Main_Page

[41] Low JR. Automatic Data Structure Selection: An Example and Overview [Internet]. CACM. 1978 [cited 2019 March 2];21(5):pages 376-385. Available from: https://dl.acm.org/citation.cfm?id=359498.

[42] Lynch B, Pirkelbauer P, Dechev D. Building Fast Concurrent Data Structures through Data Structure Families [Internet]. Albuquerque (NM), USA: Sandia National Lab; 2013 [cited 2019 February 27]. Available from: http://pirkelbauer.com/papers/mars13.pdf.

[43] Lämmel R. Software Languages: Syntax, Semantics, and Metaprogramming [Internet]. 1 ed. Berlin, Germany: Springer Berlin Heidelberg; 2018 [cited 2020 May 10]. Available from: https://kth-primo.hosted.exlibrisgroup.com/permalink/f/qra184/46KTH_ALMA_DS51174790680002456

[44] Medcalc. Independent samples t-test [Internet]. Ostend, Belgium: Medcalc; 2020 [cited 2020 January 11]. Available from: https://www.medcalc.org/manual/ttest_computationalnotes.php

[45] Merriam-Webster. Data structure [Internet]. Springfield (MA), USA: Merriam-Webster; 2020 [cited 2021 March 13]. Available from: https://www.merriam-webster.com/dictionary/data%20structure?utm_campaign=sd&utm_medium=serp&utm_source=jsonld

[46] Mogensen TÆ. Introduction to Compiler Design [Internet]. Berlin, Germany: Springer Berlin Heidelberg; 2017 [cited 2020 May 10]. Available from: https://kth-primo.hosted.exlibrisgroup.com/permalink/f/1vh8pa3/TN_cdi_askewsholts_vlebooks_9783319669663

[47] Morris J. Data Structures and Algorithms [Internet]. Auckland, New Zeeland: University of Auckland; 2004 [cited 2021 March 13]. Available from: https://www.cs.auckland.ac.nz/software/AlgAnim/ds_ToC.html

[48] Moss GE, Runciman C. Inductive Benchmarking for Purely Functional Data Structures [Internet]. Journal of Functional Programming. 2001 [cited 2019 February 15];11(5):pages 525-556. Available from: https://dl.acm.org/citation.cfm?id=968439.

[49] MyTutor. What is the difference between a data structure and a data type? [Internet]. London, UK: MyTutor; 2019 [cited 2020 May 10]. Available from: https://www.mytutor.co.uk/answers/6615/GCSE/Computing/What-is-the-difference-between-a-data-structure-and-a-data-type/

[50] Møller A, Schwartzbach MI. Static Program Analysis [Internet]. Aarhus, Denmark: Department of Computer Science Aarhus University; 2018. Available from: https://cs.au.dk/~amoeller/spa/spa.pdf.

[51] Nielson F, Nielson HR, Hankin C. Principles of Program Analysis [Internet]. 1 ed. Berlin, Germany: Springer Berlin Heidelberg; 2015 [cited 2020 May 10]. Available from: https://kth-primo.hosted.exlibrisgroup.com/permalink/f/1vh8pa3/TN_cdi_askewsholts_vlebooks_9783662038116

[52] Okasaki C. Breadth-first numbering: lessons from a small exercise in algorithm design. In: Wadler P, editor. Proceedings of the 2000 ACM SIGPLAN International Conference on Functional Programming. ICFP '00. 2000 September 18-21; Montreal, Canada. New York (NY), USA: ACM SIGPLAN Not; 2000. 35(9): pages 131–136.

[53] Okasaki C. Purely Functional Data Structures. 1 ed. Cambridge: Cambridge University Press; 1998.

[54] Pierce BC. Types and Programming Languages. Massachusetts (NE), USA: The MIT Press; 2002.

[55] Prokopec A. SnapQueue: Lock-Free Queue with Constant Time Snapshots [Internet]. In: Haller P, Miller H, editors. Proceedings of the 6th ACM SIGPLAN Symposium on Scala. SCALA 2015. 2015 June 13; Portland (ORE), USA. New York (NY), USA: ACM; 2015 [cited 2021 March 13]. pages 1-12. Available from: https://dl.acm.org/citation.cfm?id=2774976.

[56] Reilly ED. Milestones in Computer Science and Information Technology. 1 ed. Westport (CT), USA: Greenwood Publishing Group; 2003.

[57] Rowland T, Weisstein EW. Tensor [Internet]. Hanborough, UK: MathWorld; 2019 [cited 2019 March 27]. Available from: http://mathworld.wolfram.com/Tensor.html.

[58] Schonberg E, Schwartz JT, Sharir M. An Automatic Technique for Selection of Data Representations in SETL Programs [Internet]. ACM TOPLAS. 1981 [cited 2019 March 1];3(2):pages 126-143. Available from: https://www.sciencedirect.com/science/article/pii/B9780934613125500239.

[59] Schwartz JT. Automatic Data Structure Choice in a Language of Very High Level [Internet]. CACM. 1975 [cited 2019 March 2];18(12):pages 722-728. Available from: https://dl.acm.org/citation.cfm?id=361235.

[60] Shacham O, Vechev M, Yahav E. Chameleon: Adaptive Selection of Collections [Internet]. ACM SIGPLAN Notices. 2009 [cited 2019 March 2];44(6):pages 408-418. Available from: https://dl.acm.org/citation.cfm?id=1542522.

[61] Simões HR, Hammond K, Florido M., Vasconcelos P. Using Intersection Types for Cost-Analysis of Higher-Order Polymorphic Functional Programs [Internet]. In: Altenkirch T, McBride C, editors. Lecture Notes in Computer Science, vol 4502: Types for Proofs and Programs. Berlin, Germany: Springer Berlin Heidelberg; 2007 [cited 2021 March 13]. pages 221-236. Available from: https://link.springer.com/chapter/10.1007/978-3-540-74464-1_15.

[62] Skiena SS. The Algorithm Design Manual. 2 ed. London, UK: Springer; 2008.

[63] Smith W. Everyday Data Structures. 1st ed. Birmingham, UK: Packt Publishing; 2017.

[64] Spoonhower D, Blelloch GE, Harper R, Gibbons PB. Space profiling for parallel functional programs [Internet]. Journal of Functional Programming. 2010 [cited 2019 February 4];20(5-6):pages 471-461. Available from: https://www.cs.cmu.edu/~guyb/papers/SBHG11.pdf.

[65] Steindorfer MJ. Efficient Immutable Collections [dissertation thesis on the Internet]. Amsterdam, Netherlands: UvA-DARE; 2017 [cited 2019 February 19]. Available from: http://hdl.handle.net/11245.1/9afad883-4bfe-4499-8576-f36d3fbb8d0b.

[66] University of Edinburgh - School of Informatics. Algorithms and Data Structures Note 2 [Internet]. Edinburgh, UK: University of Edinburgh; 2019 [cited 2019 March 27]. Available from: https://www.inf.ed.ac.uk/teaching/courses/inf2b/algnotes/note03.pdf.

[67] University of Wisconsin-Madison. Intro to Abstract Data Types and Sequences [Internet]. Madison (WI), USA: University of Wisconsin-Madison; 2012 [cited 2019 March 27]. Available from: http://pages.cs.wisc.edu/~paton/readings/Old/fall01/SEQUENCES.html.

[68] Van-Roy P, Haridi S. Concepts, Techniques, and Models of Computer Programming. 1 ed. Cambridge (MA), USA: MIT Press; 2004.

[69] Vrajitoru D, Knight W, Practical Analysis of Algorithms [Internet]. 1 ed. Switzerland: Springer International Publishing; 2014 [cited 2020 January 1]. Available from: https://link-springer-com.focus.lib.kth.se/book/10.1007%2F978-3-319-09888-3#about

[70] Wengrow J, A Common-Sense Guide to Data Structures and Algorithms [Internet]. 1 ed. Raleigh (NC), USA: Pragmatic Bookshelf; 2017 [cited 2020 January 1]. Available from: https://learning.oreilly.com/library/view/a-common-sense-guide/9781680502794/.

[71] White JR. On the Multiple Implementation of Abstract Data Types Within a Computation [Internet]. IEEE Transactions on Software

Engineering. 1983 [cited 2019 February 26];9(4):pages 395-411. Available from: https://ieeexplore.ieee.org/abstract/document/1703074.

[72] Xu G. CoCo: Sound and Adaptive Replacement of Java Collections [Internet]. In: Castagna G, editor. Proceedings of the 27th European Conference on Object-Oriented Programming. ECOOP 2013. 2013 July 1-5; Montpellier, France. Berlin, Germany: Springer Berlin Heidelberg; 2013 [cited 2019 March 1]. pages 1-26. Available from: https://www.springer.com/us/book/9783642390371.

[73] Yellin DM. Competitive Algorithms for the Dynamic Selection of Component Implementations [Internet]. IBM Systems Journal. 2003 [cited 2019 March 2];42(1):pages 85-97. Available from: https://ieeexplore.ieee.org/abstract/document/5386847.

[74] York University - Department of Electrical Engineering and Computer Science. Sequence Abstract Data Type [Internet]. Toronto, Canada: York University; 2016 [cited 2019 March 27]. Available from: https://www.eecs.yorku.ca/course_archive/2016-17/F/3311/examples/09-SequenceADT.pdf.

Appendix A

Appendix

A.1 Method-Complexity Matrix

Operation	**Batteries finger tree**	**Okasaki linked list**	**OCaml array**	**OCaml queue**	**OCaml stack**	**Okasaki queue**
All	nlog(n)	n	n	n	n	n
Any	nlog(n)	n	n	n	n	n
Append	log(n)	n	n	n	n	n
Drop	nlog(n)	n	n	n	n	n
Filter	nlog(n)	n	n^2	n	n	n
Find	nlog(n)	n	n	n	n	n
First	const	const	const	const	n	const
Foldl	n	n	n	n	n	n
Foldr	nlog(n)	n	n	n	n	n
Isempty	const	const	const	const	const	const
last	const	n	const	n	n	n
length	const	n	const	const	const	n
Map	n	n	n	n	n	n
nth	log(n)	n	const	n	n	n
Poplast	log(n)	n	n	n	n	n
Pop	log(n)	const	n	n	n	n
Pushlast	log(n)	n	n	n	n	n
Push	log(n)	const	n	n	n	const
Reverse	n	n	n	n	n	n
Take	nlog(n)	n	n	n	n	n

Table A.1: Method-complexity matrix for our implemented data structures.