ISSN (Online) = 2522-6754 ISSN (Print) = 2522-6746
WORLD JOURNAL OF BIOLOGY AND BIOTECHNOLOGY
Research Manuscript www.sciplatform.com

Complexity analysis of algorithms: A case study on bioinformatics tools

Victória Cardoso dos Santos[a], Gislenne da Silva Moia[a], Mônica Silva de Oliveira[e], Jorianne Thyeska Castro Alves[c], Pablo Henrique Caracciolo Gomes de Sá[d], *Adonney Allan de Oliveira Veras[b]

[a] Faculty of Computer Engineering, Federal University of Pará campus Tucuruí (CAMTUC-UFPA), Pará, Brazil
[b]Faculty of Computing, Federal University of Pará Castanhal (FACOMP-UFPA), Pará, Brazil
[c]Pará State University (UEPA), Campus Marabá, Pará, Brazil
[d]Federal Rural University of Amazonia (UFRA), Campus Tomé-Açu, Pará, Brazil
[e]Amazon Engineering Development Nucleus – (NDAE-UFPA)

Authors' Contribution: **Santos, V. C:** write de manuscript and analysis data; **G.S. Moia** and M. S. Oliveira write the manuscript; **J.T.C. Alves** and **P.H.C.G. De Sá** analysys validate and manuscript revision; A.A.O Veras principal investigator, project managment, desing project.

*Corresponding Author's Email Address: allanverasce@gmail.com

Review Proccess: Double-blind peer review
Received: 21 August 2021 **Revised:** 17 November 2021 **Accepted:** 24 December 2021 **Published Online:** 26 November 2021
Digital Object Identifier (DOI) Number: https://dx.doi.org/10.33865/wjb.006.03.0445

ABSTRACT

The data volume produced by the omic sciences nowadays was driven by the adoption of new generation sequencing platforms, popularly called NGS (Next Generation Sequencing). Among the analysis performed with this data, we can mention: mapping, genome assembly, genome annotation, pangenomic analysis, quality control, redundancy removal, among others. When it comes to redundancy removal analysis, it is worth noting the existence of several tools that perform this task, with proven accuracy through their scientific publications, but they lack criteria related to algorithmic complexity. Thus, this work aims to perform an algorithmic complexity analysis in computational tools for removing redundancy of raw reads from the DNA sequencing process, through empirical analysis. The analysis was performed with sixteen raw reads datasets. The datasets were processed with the following tools: MarDRe, NGSReadsTreatment, ParDRe, FastUniq, and BioSeqZip, and analyzed using the R statistical platform, through the GuessCompx package. The results demonstrate that the BioSeqZip and ParDRe tools present less complexity in this analysis.

Keywords: Time complexity; Computational tools; Empirical analysis.

INTRODUCTION: The adoption of NGS (Next Generation Sequencing) sequencing technologies has stimulated the deposit of biological information in public databases such as the National Center for Biotechnology Information – NCBI (https://www.ncbi.nlm.nih. gov/). Due to the volume of data produced by these technologies, new algorithms capable of performing the most varied analyzes had to be developed (Chain *et al.*, 2009; Kremer *et al.*, 2017). The best practices for the development of algorithms advise that the efficiency of the proposed solution must be observed, and also the solution of the problem in question, in addition to items such as the execution time, amount of threads to perform tasks, and the memory cost. The analysis that makes inferences about the efficiency of algorithms is called algorithmic complexity analysis, through which it is possible to determine the computational effort required to execute a given computational solution (Cormin *et al.*, 1992). The Big-O notation asymptotically analyzes the behavior of a given function, where f(n) is O(g(n)) with n -> ∞ if there are two positive constants c > 0 and n0 > 1, such that f(n) ≤ c * g(n), for n ≥ n0. This notation is used to determine the speed with which a function tends to infinity (Goodrich *et al.*, 2014). It is possible to define the asymptotic behavior of complexity with the observation of the execution time of an algorithm according to the data input (Cormin *et al.*, 1992). In the literature, it is possible to observe that accuracy and memory consumption are used as parameters to determine the efficiency of an algorithm. However, the time required to perform the processing is also deterministic in the analysis of the complexity of algorithms, because an efficient algorithm, as the input grows towards infinity, presents the smallest variation in time for execution. (Levitin, 2012).

OBJECTIVES: This work aims at an empirical algorithmic complexity analysis, performed in computational tools developed to remove redundancy in raw reads from the DNA sequencing process, through the GuessCompx package (Agenis-Nevers *et al.*, 2021) using as input the processing time of each tool.

MATERIALS AND METHODS

Tools and data source: The tools selected for this analysis were MarDRe (Expósito *et al.*, 2017), ParDRe, FastUniq (Xu *et al.*, 2012), NGS Reads Treatment (Gaia *et al.*, 2019), and BioSeqZip (Urgese *et al.*, 2020). They were chosen because they are tools capable of manipulating platform-independent NGS data and are freely available to the scientific community.

For this analysis were used sixteen genome sequencing datasets obtained from NCBI, listed in Table 1.

The analysis: To measure the total processing time for each tool used in this analysis, an *inhouse-Script* was developed using the Python programming language version 3.8. The open-source GuessCompx package was used to empirically estimate the complexity from the total processing time per tool. To obtain the estimate of the algorithmic complexity of the data generated by the tools, the glm function was used (https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/glm), which is present in the platform of statistic R,

it consists of a generalized linear model, adjusting the complexity functions based on time values and the input size, to return the function that indicates the complexity analyzed in each model. Through the Big-O notation, it is possible to order the functions by the increase in the asymptotic growth rate (Goodrich *et al.*, 2014). In Table 2 are listed, in order, the eight complexity functions used in this analysis.

Organism	SRA number	Amount of reads
Escherichia coli O26:H11 str. 11368	ERR351259	5,891,069
E. coli Eco 889	SRR3465539	4,000,000
E. coli Ecol_545	SRR3999078	1,775,561
E. coli Ecol_AZ146	SRR3999096	1,954,201
E. coli O25b:H4-ST131	SRR5194991	7,300,682
Escherichia coli 042	ERR007646	7,055,348
Escherichia coli Ecol_422	SRR3999095	2,811,096
E. coli str. K-12 substr. MG1655	SRR13921546	4,851,790
E. coli O25b:H4-ST131	SRR933487	1,607,156
E. coli O111:H- str. 11128	ERR351258	4,528,090
E. coli O103:H2 str. 12009	ERR351260	3,791,997
E. coli 105_CN 19_B6_M2_C5_P1	SRR6111817	2,927,916
Salmonella enterica 63517	SRR8735180	1,777,597
Kineococcus rhizosphaerae DSM 19711	SRR6479489	2,820,667
K. xinjiangensis DSM 22857	SRR6479482	2,985,511
Arcobacter halophilus	SRR1144800	3,516,714

Table 1: Datasets used in redundancy removal. Shows the organism, SRA number and the number of reads for each dataset.

Name	Description	Execution time
Constant	Regardless of the size of the input dataset, the algorithm will always run at the same time (Goodrich *et al.*, 2014).	O(1)
Double-logarithmic	The order that divides the problem twice into smaller problems, processing at each interaction, ¼ of the data (Cormin *et al.*, 1992).	O(log log n)
Logarithm	The order that divides the problem into smaller problems, processing half of the data at each interaction (Levitin, 2012).	O(log n)
Linear	The order in which performance increases linearly in direct proportion to the size of the input dataset (Goodrich *et al.*, 2014).	O(n)
Linearithmic time	The problem is divided into smaller problems, which are solved independently and then merged (Goodrich *et al.*, 2014).	O(n log n)
Quadratic	Algorithm performance grows proportionally to the square of the input dataset size (Goodrich *et al.*, 2014).	$O(n^2)$
Cubic	Algorithm performance grows proportionally to the cube of the input dataset size (Goodrich *et al.*, 2014).	$O(n^3)$
Quadruple	Algorithm performance grows proportionally to quadruple the size of the input dataset (Goodrich *et al.*, 2014).	$O(n^4)$

Table 2: List of Big-O notation in the analysis.

Workstation: The workstation used to perform the analysis were PowerEdge T440, Intel Xeon Silver 4214 R de 2.4G, 12C, 64GB RDIMM memory, 2933MT/s.

RESULTS AND DISCUSSION: After processing the datasets with each tool, the number of reads per dataset and the total processing time per tool in seconds were generated, as shown in Table 3. These data were used as inputs to obtain an estimate of the algorithmic complexity of each tool. Figure 1 below shows the graph generated right after an initial ordering in an ascending manner, which shows the behavior of each tool as the size of the datasets will increase. On the vertical axis, it contains the time each tool took to remove duplicate reads in each dataset, and on the horizontal axis, it shows the size of the datasets.

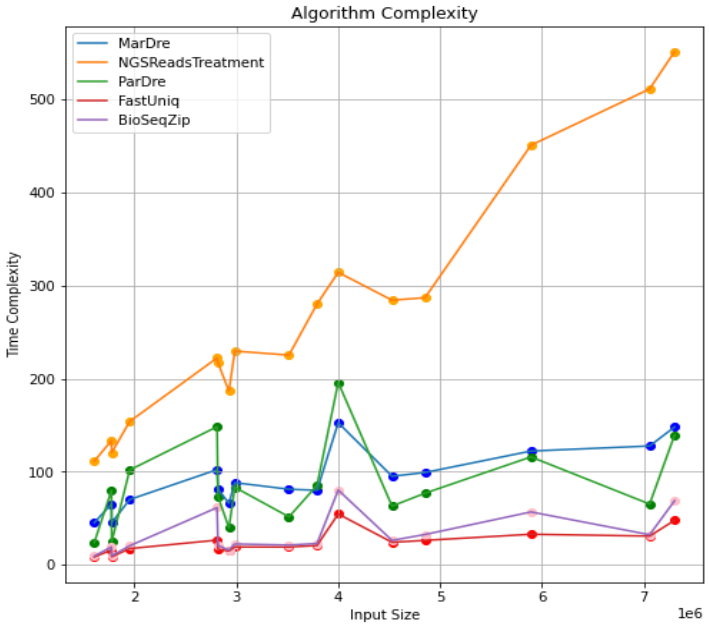

Figure 1: Figure 1: Dataset processing time for each tool.

Figure 2 shows the result generated from the data obtained through the GuessCompx package using the glm function, presenting the best-fitted model referring to the original input and execution time data of the tools, indicating the time complexity of each.

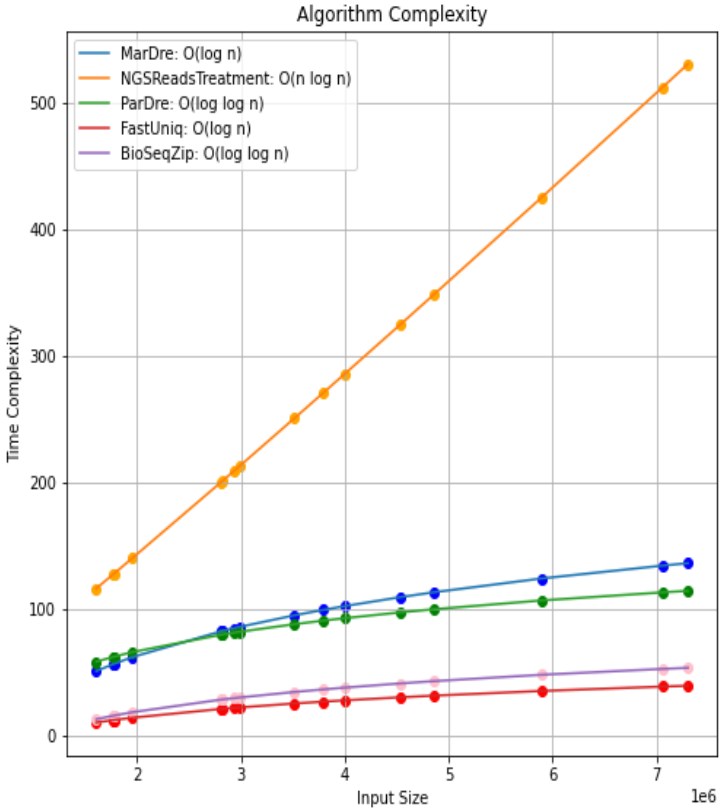

Figure 2: Model adjusted with the classification of the complexity of each tool.

SRA Number	Amount of reads	MarDRe	NGSReadTreatment	ParDRe	FastUniq	BioSeqZip
ERR351259	5,891,069	122.13933	450.61075	115.84975	32.76499	56.74513
SRR3465539	4,000,000	152.54016	314.03648	195.79587	54.55795	80.24278
SRR3999078	1,775,561	65.32743	133.49737	79.97755	15.81383	19.27481
SRR3999096	1,954,201	70.04805	153.90345	101.78853	17.38272	20.40122
SRR5194991	7,300,682	148.03880	551.19522	139.51281	47.75186	68.86436
ERR007646	7,055,348	127.57795	511.02596	65.19286	30.82049	32.26478
SRR3999095	2,811,096	102.13908	222.11067	148.28935	26.45431	61.35715
SRR13921546	4,851,790	99.08816	286.84546	76.88631	26.19284	32.50419
SRR933487	1,607,156	45.21834	111.25275	23.94190	8.44803	9.36266
ERR351258	4,528,090	95.04159	284.15241	63.02875	24.14208	26.16725
ERR351260	3,791,997	80.04319	280.06446	85.27908	20.72433	22.81125
SRR6111817	2,927,916	65.94035	186.44247	39.58490	14.80441	15.88374
SRR8735180	1,777,597	45.13097	120.34384	24.58334	8.53508	9.62314
SRR6479489	2,820,667	81.12143	216.86710	73.42460	16.62816	20.68559
SRR6479482	2,985,511	88.18347	229.49797	83.23705	19.14120	22.32020
SRR1144800	3,516,714	81.11041	225.18378	51.43773	19.05139	21.18828

Table 3. List of datasets processed by tool.

It can be noted that for the datasets, from the worst to the best complexity time, that is, from the highest to the lowest asymptotic growth rate, there is the NGSReadsTreatment tool, which presents the performance of order O(n log n). Then, the tools MarDRe and FastUniq, which operated in a similar way on the results, obtained a growth of order O(log n). Finally, presenting the best performances in this analysis are the BioSeqZip and ParDRe tools, which obtained O(log log n) complexity.

CONCLUSION: In this work, a complexity analysis was performed among five computational tools, which are used to remove redundancy in raw reads resulting from the DNA sequencing process, using input size and time values as parameters. It is important to emphasize that although the complexity of algorithms is not a new subject, there is a lack of materials within the area of computing and mathematics that address the functions related to the complexity of algorithms. Based on the results obtained, among the five chosen tools, BioSeqZip and ParDRe were shown to be more effective in relation to the datasets used in this analysis, presenting the O (log log n) order complexity. Therefore, the analysis of algorithmic complexity in computational tools applied in the omics sciences is necessary, because, with the constant increase in the volume of data, they become more complex to be processed and, the more predictable the tool in terms of cost of time is, more useful it will be, being able to assist the user, as an evaluation criterion, in choosing the tool that best corresponds the needs of your research.

CONFLICT OF INTERESTS: Authors have no conflict of interest

ACKNOWLEDGMENT: Thanks to the Brazilian Research Council (CNPq) and the Federal University of Pará, this work received help from PROPESP/UFPA. This work is part of the research developed by BIOD (Bioinformatics, Omics, and Development research group - www.biod.ufpa.br). AAOV thanks to Federal University of Pará (UFPA) and PHCGS thanks to PVTA341-2020 from Federal Rural University of Amazonia (UFRA).

REFERENCES: Agenis-Nevers, M., N. D. Bokde, Z. M. Yaseen and M. K. Shende, 2021. An empirical estimation for time and memory algorithm complexities: Newly developed r package. Multimedia tools applications, 80(2): 2997-3015.

Chain, P., D. Grafham, R. Fulton, M. Fitzgerald, J. Hostetler, D. Muzny, J. Ali, B. Birren, D. Bruce and C. Buhay, 2009. Genome project standards in a new era of sequencing. Science, 326(5950): 236-237.

Cormin, T., C. Leiserson and R. Rivest, 1992. Introduction to algorithms mit press: Cambridge. MA.

Expósito, R. R., J. Veiga, J. González-Domínguez and J. Touriño, 2017. Mardre: Efficient mapreduce-based removal of duplicate DNA reads in the cloud. Bioinformatics, 33(17): 2762-2764.

Gaia, A. S. C., P. H. C. G. de Sá, M. S. de Oliveira and A. A. de Oliveira Veras, 2019. Ngsreadstreatment–a cuckoo filter-based tool for removing duplicate reads in ngs data. Scientific reports, 9(1): 1-6.

Goodrich, M. T., R. Tamassia and M. H. Goldwasser, 2014. Data structures and algorithms in java. John Wiley & Sons.

Kremer, F. S., A. J. A. McBride and L. d. S. Pinto, 2017. Approaches for in silico finishing of microbial genome sequences. Genetics molecular biology, 40: 553-576.

Levitin, A., 2012. Introduction to the design & analysis of algorithms 3rd. Villanova university, 1(1): p18.

Urgese, G., E. Parisi, O. Scicolone, S. Di Cataldo and E. Ficarra, 2020. Bioseqzip: A collapser of ngs redundant reads for the optimization of sequence analysis. Bioinformatics, 36(9): 2705-2711.

Xu, H., X. Luo, J. Qian, X. Pang, J. Song, G. Qian, J. Chen and S. Chen, 2012. Fastuniq: A fast de novo duplicates removal tool for paired short reads. PloS one, 7(12): e52249.