JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12 Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3055

Implementasi *Divide and Conquer* pada Algoritma Perkalian Matriks untuk Efisiensi Komputasi

Christian Nicholas Sinaga[1], Jhon Gabriel Simarmata[2*], Juhraini Helfiana Lexa[3*], Adidtya Perdana[4]

[1-4]Fakultas Ilmu Matematika Dan Ilmu Pengetahuan Alam, Program Studi Ilmu Komputer , Universitas Negeri Medan, Medan, Indonesia
Email: [1]christiannicholassinaga@gmail.com, [2*]rielsimarmata06@gmail.com,
[3*]juhrainihelfianalexa@gmail.com,[4]adidtya@unimed.ac.id
(* : coressponding author)

Abstrak−Perkalian matriks merupakan operasi fundamental dalam komputasi yang banyak digunakan dalam berbagai bidang, namun metode konvensional memiliki kompleksitas waktu $O(n^3)$ sehingga kurang efisien untuk ukuran data besar. Penelitian ini bertujuan untuk mengimplementasikan algoritma perkalian matriks menggunakan pendekatan *Divide and Conquer* serta menganalisis efisiensi kinerjanya dibandingkan metode konvensional. Metode penelitian meliputi studi literatur, perancangan algoritma, implementasi menggunakan bahasa Python, serta pengujian dengan berbagai ukuran matriks. Selain itu, digunakan optimasi algoritma Strassen untuk meningkatkan performa. Hasil pengujian menunjukkan bahwa pada matriks berukuran kecil, metode konvensional masih memiliki performa yang kompetitif karena minimnya overhead. Namun, pada matriks berukuran menengah hingga besar, pendekatan *Divide and Conquer* yang dioptimalkan dengan algoritma Strassen terbukti lebih efisien dengan waktu komputasi yang lebih cepat. Dengan demikian, pendekatan ini lebih direkomendasikan untuk pengolahan matriks berukuran besar guna meningkatkan efisiensi komputasi.

Kata Kunci: Perkalian Matriks; *Divide and Conquer*; Efisiensi Komputasi; Algoritma Strassen; Python

Abstract− *Matrix multiplication is a fundamental operation in computing that is widely used in various fields; however, the conventional method has a time complexity of $O(n^3)$, making it less efficient for large-scale data. This study aims to implement a matrix multiplication algorithm using the Divide and Conquer approach and analyze its performance efficiency compared to the conventional method. The research methodology includes literature review, algorithm design, implementation using the Python programming language, and testing with various matrix sizes. In addition, the Strassen algorithm is applied as an optimization technique to improve performance. The results show that for small-sized matrices, the conventional method remains competitive due to its minimal overhead. However, for medium to large-sized matrices, the Divide and Conquer approach optimized with the Strassen algorithm proves to be more efficient, with faster computation time. Therefore, this approach is recommended for processing large-scale matrices to enhance computational efficiency.*

Keywords *Matrix Multiplication; Divide and Conquer; Computational Efficiency,; Strassen Algorithm; Python*

1. PENDAHULUAN

Perkalian matriks merupakan salah satu operasi fundamental dalam bidang komputasi yang memiliki peran penting dalam berbagai aplikasi, seperti kecerdasan buatan, pengolahan citra, komputasi ilmiah, hingga analisis data berskala besar. Seiring dengan meningkatnya ukuran data dan kompleksitas permasalahan, kebutuhan akan algoritma yang efisien menjadi semakin krusial. Metode konvensional dalam perkalian matriks umumnya memiliki kompleksitas waktu $O(n^3)$, sehingga kurang optimal ketika dihadapkan pada ukuran matriks yang besar.

Salah satu pendekatan yang dapat digunakan untuk meningkatkan efisiensi komputasi adalah paradigma divide and conquer. Pendekatan ini bekerja dengan cara membagi suatu permasalahan besar menjadi beberapa sub-masalah yang lebih kecil, menyelesaikannya secara independen, kemudian menggabungkan hasilnya menjadi solusi akhir (Harjono, 2023). Konsep ini dinilai efektif karena mampu menyederhanakan proses penyelesaian masalah kompleks serta meningkatkan efisiensi pemrosesan data.

Algoritma divide and conquer telah banyak diterapkan dalam berbagai bidang komputasi. Misalnya, dalam pengukuran wilayah menggunakan metode grid, pendekatan ini mampu memberikan hasil yang cukup optimal dengan tingkat akurasi yang tinggi (Anggraeni & Wibawa, 2024). Selain itu, pada sistem pengurutan data dan pengolahan informasi, metode ini juga terbukti mampu meningkatkan efisiensi proses melalui pembagian masalah menjadi bagian-bagian yang

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12 Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3055

lebih kecil dan terstruktur (Santi et al., 2024). Dalam konteks komputasi modern, pendekatan divide and conquer juga digunakan untuk meningkatkan efisiensi dalam pemrosesan workflow dan komputasi paralel (Toussi et al., 2022)..

Lebih lanjut, pendekatan divide and conquer tidak hanya relevan dalam algoritma klasik, tetapi juga diterapkan dalam berbagai pengembangan teknologi terbaru, termasuk pemrosesan data berbasis model dan komputasi skala besar, di mana pemecahan masalah ke dalam sub-komponen menjadi kunci dalam meningkatkan performa sistem (Zou et al., 2022). Bahkan dalam komputasi numerik seperti dekomposisi matriks, pendekatan ini mampu meningkatkan efisiensi paralel dan mengurangi konsumsi energi secara signifikan (Toussi et al., 2022).

Berdasarkan uraian tersebut, dapat disimpulkan bahwa penerapan algoritma divide and conquer memiliki potensi besar dalam meningkatkan efisiensi komputasi, khususnya pada operasi yang kompleks seperti perkalian matriks. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan algoritma perkalian matriks berbasis divide and conquer serta menganalisis efisiensi komputasi yang dihasilkan dibandingkan dengan metode konvensional.

2. METODE

2.1 Tahapan Penelitian

Pelaksanaan penelitian ini dibagi ke dalam langkah-langkah sistematis sebagai berikut:

1. Studi Literatur: Tahap awal dimulai dengan mengkaji referensi untuk memahami konsep dasar algoritma perkalian matriks dan paradigma metode Divide and Conquer.
2. Perancangan Algoritma: Melakukan penentuan proses pembagian matriks menjadi sub-matriks yang lebih kecil, penyusunan alur kerja algoritma, serta pembuatan pseudocode.
3. Implementasi Sistem: Algoritma diimplementasikan ke dalam bahasa pemrograman Python. Proses ini mencakup pembuatan fungsi input matriks, pembagian matriks, perhitungan secara rekursif, serta penggabungan hasil menjadi matriks akhir.
4. Pengujian Sistem: Pengujian dilakukan dengan menggunakan berbagai ukuran matriks, baik skala kecil maupun besar. Hal ini bertujuan untuk mengetahui dan mengukur kinerja komputasi dari algoritma yang telah dibuat.
5. Analisis Hasil: Hasil dari pengujian dianalisis dengan membandingkan waktu komputasi antara metode Divide and Conquer dan metode konvensional. Analisis dilakukan untuk menilai efisiensi algoritma berdasarkan kecepatan eksekusi dan kompleksitas waktu. Berdasarkan hasil analisis tersebut, selanjutnya ditarik kesimpulan mengenai keunggulan metode Divide and Conquer dalam meningkatkan efisiensi komputasi pada proses perkalian matriks.

2.2 Lingkungan dan Perangkat Penelitian

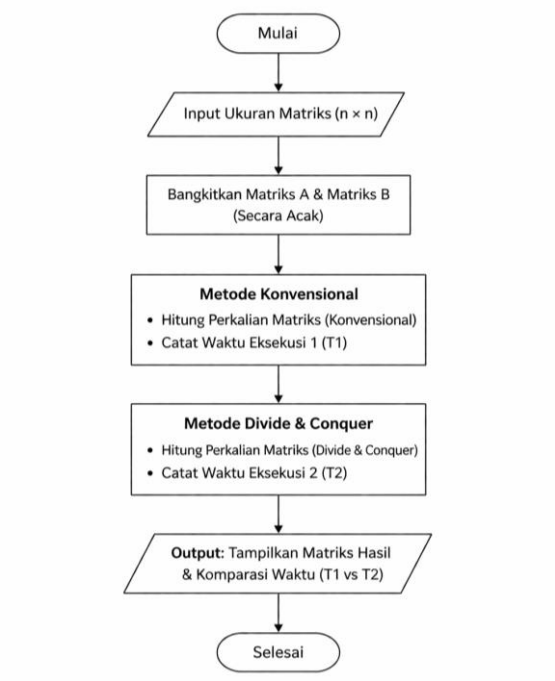
Untuk menjalankan eksperimen komputasi ini, penelitian didukung oleh beberapa teknologi dan pustaka perangkat lunak, yaitu:

1. Bahasa Pemrograman: Python digunakan sebagai bahasa utama untuk mengimplementasikan kedua metode algoritma perkalian matriks.
2. Perangkat Lunak (Software): Visual Studio Code difungsikan sebagai lingkungan pengembangan (Integrated Development Environment) untuk menulis, menjalankan, dan menguji kode program.
3. Pustaka (Library): Pustaka NumPy digunakan untuk membantu pengolahan matriks , sementara Library Time bawaan Python digunakan untuk mengukur waktu eksekusi dari setiap algoritma secara presisi sehingga dapat dilakukan analisis efisiensi komputasi.

2.3 Alur Kerja Sistem

Sistem komputasi yang dibangun memiliki alur kerja sekuensial untuk memfasilitasi perbandingan langsung. Alur kerja tersebut dapat dilihat pada Gambar 1.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12 Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3055



Gambar 1.1 Flowchart alur kerja program

Berdasarkan Gambar 1, alur kerja pengujian algoritma berjalan sebagai berikut:

1. Proses dimulai dengan inisialisasi program.
2. Sistem menerima masukkan (input) berupa ukuran matriks (n x n).
3. Program kemudian membangkitkan elemen untuk Matriks A dan Matriks B secara acak (random) sesuai dengan ukuran yang telah diinputkan.
4. Sistem mengeksekusi perhitungan perkalian matriks menggunakan Metode Konvensional. Modul waktu diaktifkan untuk mencatat durasi proses ini sebagai Waktu Eksekusi 1 (T1).
5. Selanjutnya, sistem mengeksekusi perhitungan pada matriks yang sama menggunakan Metode Divide & Conquer. Durasi pemrosesan pada tahap ini dicatat sebagai Waktu Eksekusi 2 (T2).
6. Pada tahap akhir, sistem menampilkan luaran (output) berupa matriks hasil perkalian beserta komparasi waktu komputasi antara T1 dan T2 untuk mempermudah analisis efisiensi.

3. ANALISA DAN PEMBAHASAN

Pengujian dalam penelitian ini dilakukan untuk mengevaluasi kinerja algoritma perkalian matriks dengan membandingkan metode konvensional (brute force) dan pendekatan Divide and Conquer. Selain itu, dilakukan pula pengujian terhadap pengembangan metode Divide and Conquer, yaitu algoritma Strassen, sebagai bentuk optimasi.

Eksperimen dilakukan dengan variasi ukuran matriks persegi, mulai dari ukuran kecil hingga menengah. Setiap percobaan diukur berdasarkan waktu eksekusi yang dibutuhkan untuk menyelesaikan proses perkalian matriks.

3.1 Analisis Perbandingan Metode

3.1.1 Analisis Metode Konvensional

Metode konvensional atau brute force merupakan pendekatan yang paling langsung dalam menyelesaikan perkalian matriks. Setiap elemen hasil diperoleh melalui proses iteratif yang melibatkan tiga tingkat perulangan.

Keunggulan utama dari metode ini terletak pada kesederhanaan struktur algoritmanya. Tidak terdapat proses rekursif maupun pembagian data, sehingga overhead komputasi relatif kecil. Hal ini

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12 Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3055

menyebabkan metode konvensional cenderung memberikan performa yang baik pada ukuran matriks kecil hingga menengah.

Namun demikian, peningkatan ukuran matriks akan berdampak langsung pada jumlah operasi yang harus dilakukan. Karena kompleksitas waktunya bersifat kubik, waktu eksekusi akan meningkat secara signifikan seiring bertambahnya ukuran matriks.

3.1.2 Analisis Metode Divide and Conquer

Pendekatan Divide and Conquer dalam penelitian ini diterapkan dengan membagi matriks ke dalam beberapa submatriks berukuran lebih kecil, kemudian setiap bagian diselesaikan secara rekursif sebelum akhirnya digabungkan kembali untuk memperoleh hasil akhir. Untuk meningkatkan efisiensi, digunakan algoritma Strassen sebagai bentuk optimasi dari pendekatan tersebut, terutama untuk mengurangi jumlah operasi perkalian yang dilakukan.

Secara konseptual, metode ini menawarkan alur penyelesaian yang lebih terstruktur dan memiliki potensi untuk dikembangkan lebih lanjut, misalnya dalam lingkungan komputasi paralel. Namun, dalam implementasi praktis khususnya menggunakan Pythonpendekatan Divide and Conquer tidak selalu memberikan peningkatan kinerja yang signifikan apabila tidak dioptimalkan dengan baik.

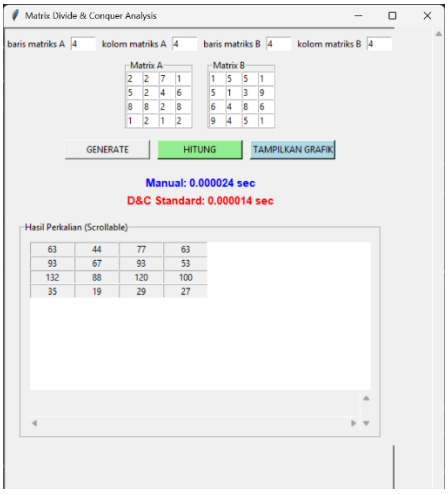
Beberapa faktor yang mempengaruhi hal tersebut antara lain adalah biaya tambahan dari pemanggilan fungsi rekursif yang berulang, kebutuhan alokasi memori baru saat proses pembagian matriks, serta overhead pada tahap penggabungan kembali hasil perhitungan. Untuk mengatasi permasalahan ini, algoritma Strassen digunakan secara selektif (hybrid), sehingga pada ukuran matriks tertentu perhitungan dialihkan ke metode konvensional guna meminimalkan overhead.

Dengan kondisi tersebut, pada matriks berukuran kecil, waktu eksekusi metode ini masih berpotensi lebih lambat dibandingkan metode konvensional, meskipun pada ukuran yang lebih besar optimasi yang diterapkan dapat memberikan keuntungan performa.

3.2 Analisis Perbandingan Pengujian pada Matriks

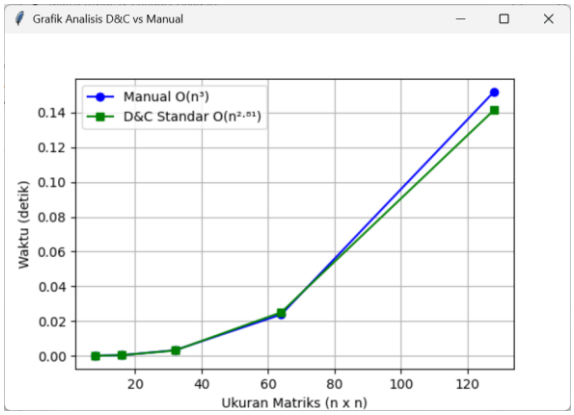
3.2.1 Pengujian pada Matriks Ukuran Kecil

Ukuran matriks yang dipakai yaitu 4×4 yang merujuk pada dimensi suatu matriks yang terdiri atas empat baris dan empat kolom. Penulisan ini mengikuti konvensi umum dalam matematika, di mana angka pertama menunjukkan jumlah baris, sedangkan angka kedua menunjukkan jumlah kolom. Dengan demikian, matriks berukuran 4×4 memiliki total enam belas elemen yang tersusun dalam empat baris horizontal dan empat kolom vertikal.



Gambar 3.1 Hasil Perbandingan waktu komputasi pada matriks ukuran kecil

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12 Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3055

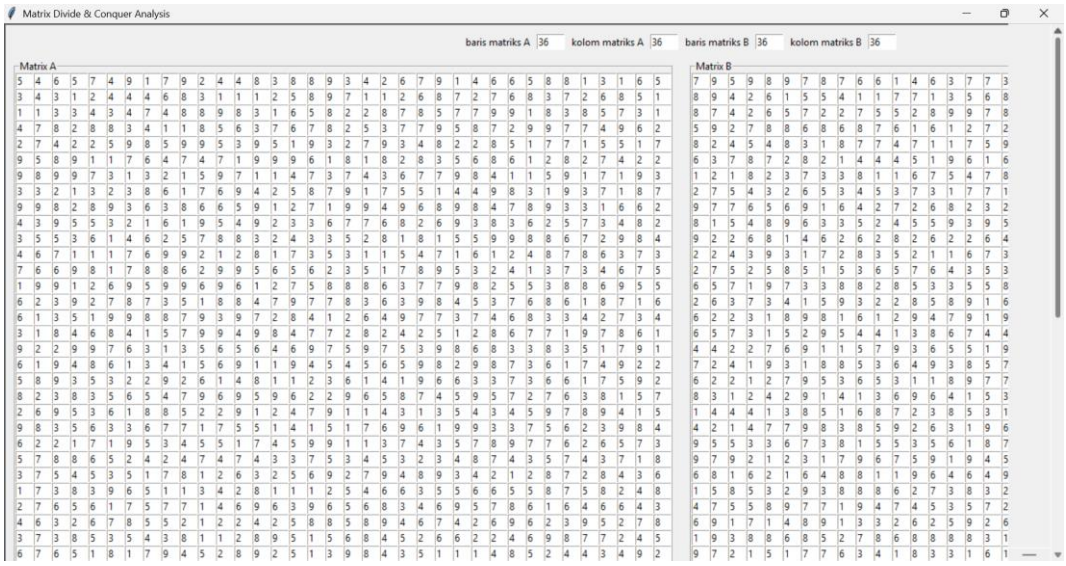


Gambar 3.2 Grafik perbandingan waktu komputasi pada matriks ukuran kecil

Berdasarkan gambar 3.1 dan gambar 3.2 dapat dilihat bahwa waktu komputasi algoritma divide and conquer sedikit lebih cepat dibandingkan metode konvesional.

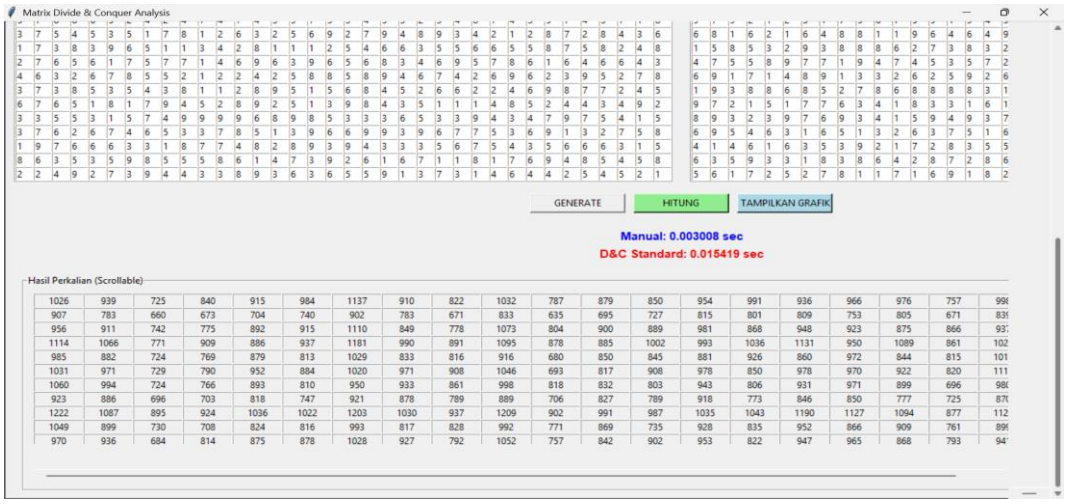
3.2.2 Pengujian pada Matriks Ukuran Menengah

Ukuran matriks yang dipakai yaitu 36 × 36 yang merujuk pada dimensi suatu matriks yang terdiri atas tiga puluh enam baris dan tiga puluh enam kolom.

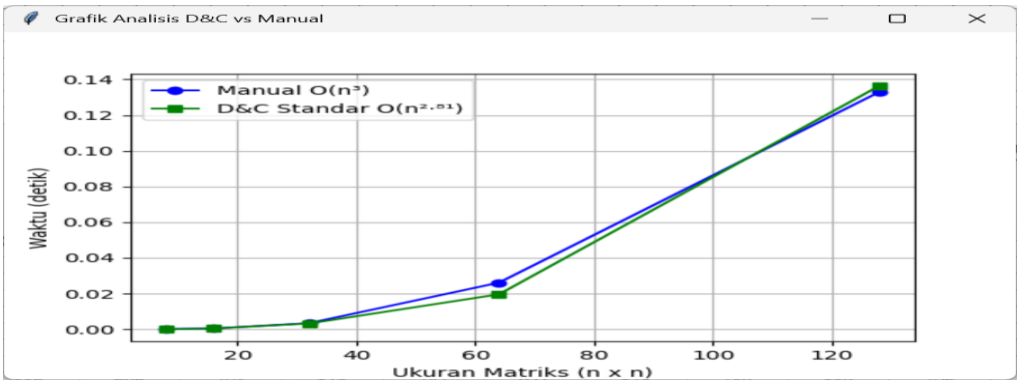


Gambar 3.3 Matriks 36 X 36

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12 Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3055



Gambar 3.4 Hasil perbandingan waktu komputasi pada matriks berukuran menengah



Gambar 3.5 Grafik Hasil perbandingan waktu komputasi pada matriks berukuran menengah

Berdasarkan gambar 3.4 dan gambar 3.5 dapat dilihat perbedaan signifikan bahwa waktu komputasi algoritma divide and conquer lebih cepat dibandingkan metode konvesional. Dengan selisih 0,29 detik, hal ini mennunjukkan bahwa divide and conquer memiliki waktu komputasi yang lebih cepat dibandingkan metode konvesional

4. KESIMPULAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat disimpulkan bahwa metode konvensional (brute force) dan pendekatan Divide and Conquer memiliki karakteristik performa yang berbeda tergantung pada ukuran matriks yang digunakan. Pada matriks berukuran kecil, metode konvensional menunjukkan kinerja yang cukup kompetitif karena tidak memiliki overhead tambahan seperti rekursi dan pembagian data. Meskipun demikian, hasil pengujian menunjukkan bahwa algoritma Divide and Conquer yang dioptimalkan dengan Strassen tetap mampu memberikan waktu komputasi yang sedikit lebih baik, meskipun selisihnya tidak terlalu signifikan.

Seiring dengan bertambahnya ukuran matriks, keunggulan pendekatan Divide and Conquer menjadi semakin terlihat. Pada pengujian matriks berukuran menengah (36×36), algoritma ini menunjukkan performa yang lebih unggul dibandingkan metode konvensional dengan selisih waktu yang cukup signifikan. Hal ini menunjukkan bahwa optimasi melalui algoritma Strassen efektif dalam mengurangi kompleksitas perhitungan, sehingga lebih efisien untuk menangani data berukuran lebih besar. Dengan demikian, dapat disimpulkan bahwa pendekatan Divide and Conquer dengan optimasi Strassen lebih direkomendasikan untuk matriks berukuran menengah hingga besar, sementara metode konvensional masih relevan digunakan pada matriks berukuran kecil.

JRIIN : Jurnal Riset Informatika dan Inovasi
Volume 3, No. 12 Tahun 2026
ISSN 3025-0919 (media online)
Hal 3049-3055

REFERENCES

Allman, E. S., Baños, H., Rhodes, J. A., & Wicke, K. (2025). NANUQ+: A divide-and-conquer approach to network estimation. *Algorithms for Molecular Biology*, *20*(14). https://doi.org/10.1186/s13015-025-00274-w

Anggraeni, D. T., & Wibawa, C. (2024). Implementasi dan analisis akurasi pengukuran luas wilayah Kota Bekasi menggunakan algoritma divide and conquer & metode grid. *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, *14*(1), 9–15.

Azka, S., Liebenlito, M., & Sutanto, T. E. (2025). Analisis performa klaster single board computer dalam implementasi singular value decomposition. *Euler: Jurnal Ilmiah Matematika, Sains dan Teknologi*, *13*(3), 286–292. https://doi.org/10.37905/euler.v13i3.33367

Harjono, S. (2023). Pengembangan aplikasi pembuat surat kontrak kerja, peringatan dan perjalanan dinas dengan algoritma divide and conquer. *INFOMATEK: Jurnal Informatika, Manajemen dan Teknologi*, *25*(1), 1–10. https://doi.org/10.23969/infomatek.v25i1.5962

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms* (4th ed.). Cambridge, MA: MIT Press.

Khojasteh Toussi, G., Naghibzadeh, M., Abrishami, S., Taheri, H., & Abrishami, H. (2022). EDQWS: An enhanced divide and conquer algorithm for workflow scheduling in cloud. *Journal of Cloud Computing*, *11*(13). https://doi.org/10.1186/s13677-022-00284-8

Santi, I. H., Karina, P. M. D., Kusna, A. H., & Priyanto, S. A. P. (2024). Pengurutan stok barang toko Andis's Collection menggunakan algoritma divide and conquer. *Jurnal Teknologi Informasi: Teori, Konsep dan Implementasi*, *15*(1), 35–40. https://doi.org/10.36382/jti-tki.v15i1.522

Sitorus, Z., Renyaan, A. S., Kmurawak, R. M. B., & Lokollo, P. D. (2024). *Tinjauan mendalam tentang ilmu komputer: Konsep dasar, algoritma, dan perkembangan terkini*. Medan: PT Media Penerbit Indonesia.

Walangi, S. T., Thoyyibah, T., & Rachmatika, R. (2024). *Algoritma Analisis*. Yogyakarta: PT Penamuda Media.

Zou, Y., Liu, H., Gui, T., Wang, J., Zhang, Q., Tang, M., Li, H., & Wang, D. (2022). Divide and conquer: Text semantic matching with disentangled keywords and intents. *Findings of the Association for Computational Linguistics: ACL 2022*, 3622–3632.