Efektivitas Algoritma Greedy Pada Coin Change Problem: Studi Komparatif dengan Dynamic Programming

Ade Hastuty[1], Muh. Rifky Wahyudi Bambang[2], Muh. Zulkifli[3], Muh Syahril[4] [1,2,3,4]
Program Studi Teknik Informatika, Universitas Muhammadiyah Parepare, Parepare, Indonesia
Email: adehastuty@iainpare.ac.id, wahyudirifki09@gmail.com, kifli0137@gmail.com, arilb9950@gmail.com

Abstrak

Coin Change Problem merupakan salah satu permasalahan optimasi klasik yang bertujuan menentukan jumlah koin atau lembar uang minimum untuk membentuk suatu nilai nominal tertentu. Berbagai metode optimasi telah diusulkan untuk menyelesaikan permasalahan ini, salah satunya Algoritma Greedy yang dikenal sederhana, efisien, dan memiliki kebutuhan komputasi rendah. Penelitian ini menganalisis efektivitas Algoritma Greedy dalam menyelesaikan Coin Change Problem menggunakan bahasa pemrograman Object Pascal dengan struktur data larik (array), serta membandingkannya secara empiris dengan Dynamic Programming baik pada sistem pecahan Rupiah maupun sistem pecahan non-kanonik. Kebaruan penelitian ini terletak pada analisis eksperimental komparatif terhadap kompleksitas praktis, penggunaan memori, serta batas empiris optimalitas algoritma. Penelitian menggunakan metode eksperimen, dengan pengukuran waktu eksekusi melalui repeated-trial benchmarking pada sepuluh nilai nominal Rupiah dan lima belas nilai nominal pada sistem pecahan hipotetis {1, 3, 4}. Hasil pengujian menunjukkan bahwa Algoritma Greedy menghasilkan jumlah pecahan yang identik dengan solusi optimal Dynamic Programming pada seluruh data uji sistem Rupiah, dengan waktu eksekusi hingga sekitar 185.000 kali lebih cepat pada nominal terbesar. Namun, pada sistem pecahan non-kanonik, Algoritma Greedy gagal mencapai solusi optimal pada 20 persen data uji. Temuan ini menegaskan bahwa keunggulan praktis Algoritma Greedy sangat dibatasi oleh sifat kanonik dari sistem pecahan yang digunakan.

Kata Kunci: Algoritma Greedy; Coin Change Problem; Dynamic Programming; Optimasi; Benchmarking Komparatif.

Abstract

Coin Change Problem is a classical optimization problem that aims to determine the minimum number of coins or banknotes required to represent a given monetary value. Various optimization methods have been proposed to solve this problem, among which the Greedy Algorithm is recognized for its simplicity, efficiency, and low computational cost. This study analyzed the effectiveness of the Greedy Algorithm in solving the Coin Change Problem using Object Pascal with an array data structure, and empirically compared it with Dynamic Programming on both the Indonesian Rupiah denomination system and a non-canonical denomination system. The primary novelty of this research lies in the comparative experimental analysis of practical

complexity, memory usage, and the empirical limits of algorithm optimality. The research employed an experimental approach, measuring execution time through repeated-trial benchmarking across ten Rupiah nominal values and fifteen values under a hypothetical {1, 3, 4} denomination system. Results showed that the Greedy Algorithm produced results identical to the optimal Dynamic Programming solution across all Rupiah test cases, with execution times up to approximately 185,000 times faster at the largest tested nominal value. However, on the non-canonical system, the Greedy Algorithm failed to reach the optimal solution in 20 percent of test cases. These findings confirm that the practical advantages of the Greedy Algorithm are strictly limited by the canonical nature of the denomination system used.

Keywords: Greedy Algorithm; Coin Change Problem; Dynamic Programming; Optimization; Comparative Benchmarking.

PENDAHULUAN

Perkembangan teknologi informasi telah mendorong pemanfaatan algoritma yang semakin efisien dalam menyelesaikan berbagai permasalahan komputasi di berbagai sektor industri. Salah satu permasalahan fundamental yang banyak dijumpai dalam bidang ilmu komputer dan riset operasi adalah permasalahan optimasi kombinatorial, di mana salah satu bentuk paling klasiknya dikenal sebagai Coin Change Problem. Coin Change Problem adalah sebuah tantangan komputasi yang bertujuan untuk menentukan kombinasi pecahan uang (koin atau lembar kertas) yang diperlukan untuk membentuk suatu nilai nominal tertentu, dengan batasan utama (constraint) berupa penggunaan jumlah keping atau lembar uang yang seminimal mungkin.

Di era digitalisasi keuangan saat ini, permasalahan ini menjadi sangat relevan karena banyak diterapkan secara masif pada sistem pembayaran elektronik, arsitektur perangkat lunak mesin Anjungan Tunai Mandiri (ATM), mesin penjual otomatis (vending machine), sistem kasir terkomputerisasi (Point of Sale), hingga logika backend pada aplikasi dompet digital (e-wallet). Oleh karena itu, pemilihan algoritma yang tepat dan efisien menjadi faktor krusial untuk menghasilkan proses kalkulasi penukaran uang yang cepat, akurat, dan tidak membebani sumber daya sistem.

Pada praktiknya di lapangan, infrastruktur perangkat keras seperti mesin ATM, mesin kasir portabel, dan mesin penjual otomatis umumnya beroperasi menggunakan mikrokontroler atau prosesor embedded (sistem tersemat) yang memiliki spesifikasi sumber daya komputasi (CPU) dan kapasitas memori (RAM) yang sangat terbatas. Di saat yang bersamaan, perangkat-perangkat ini harus mampu menangani volume transaksi harian yang sangat tinggi tanpa mengalami jeda waktu pemrosesan (latency) yang berarti.

Kebutuhan akan efisiensi tingkat tinggi ini menjadikan Algoritma Greedy sebagai salah satu kandidat algoritma optimasi yang paling menarik untuk diimplementasikan. Algoritma Greedy menawarkan kesederhanaan logika yang memungkinkannya diterjemahkan ke dalam kode instruksi mesin yang sangat ringan.

Algoritma ini beroperasi dengan memegang prinsip "mengambil pilihan lokal terbaik pada setiap langkah" (local optimum) dengan harapan bahwa serangkaian pilihan lokal tersebut pada akhirnya akan bermuara pada solusi global yang optimal. Dalam konteks Coin Change Problem, Algoritma Greedy akan selalu memilih pecahan uang dengan nilai terbesar yang masih memungkinkan untuk mengurangi sisa

nominal, mengulangi proses tersebut hingga sisa nominal mencapai angka nol. Kesederhanaan ini memungkinkan implementasi pada perangkat berkapasitas rendah tanpa mengorbankan ketepatan hasil, namun dengan syarat yang sangat ketat: sistem pecahan uang yang digunakan harus bersifat kanonik (canonical).

Secara teoritis, permasalahan optimasi seperti Coin Change Problem dapat diselesaikan menggunakan beberapa paradigma pendekatan algoritmik, di antaranya adalah Brute Force, Dynamic Programming (DP), dan Algoritma Greedy.

Pendekatan Brute Force bekerja dengan cara mengevaluasi seluruh kemungkinan kombinasi pecahan uang yang ada secara menyeluruh (exhaustive search). Meskipun pendekatan ini memberikan garansi mutlak untuk menemukan solusi yang paling optimal, Brute Force memiliki kompleksitas waktu eksponensial yang menyebabkan waktu komputasi akan melonjak drastis secara tidak masuk akal seiring dengan bertambahnya nilai nominal yang harus dipecahkan, menjadikannya sama sekali tidak praktis untuk digunakan pada sistem nyata.

Di sisi lain, Dynamic Programming hadir sebagai solusi elegan untuk mengatasi inefisiensi pencarian menyeluruh tersebut. Dynamic Programming bekerja berdasarkan prinsip optimalitas Bellman, memecah masalah utama menjadi sub-submasalah yang lebih kecil, menyelesaikan setiap submasalah hanya satu kali, dan menyimpan hasilnya di dalam sebuah tabel memori (proses yang dikenal sebagai memoization atau tabulasi bottom-up).

Pendekatan DP menjamin penemuan solusi optimal untuk sembarang jenis sistem pecahan uang, tanpa mempedulikan apakah sistem tersebut kanonik atau tidak. Akan tetapi, keunggulan jaminan optimalitas universal ini harus dibayar mahal dengan kompleksitas ruang (kebutuhan memori) dan kompleksitas waktu yang berskala pseudo-polynomial. Kompleksitas DP sangat bergantung secara linier pada besarnya nilai nominal itu sendiri (bukan sekadar jumlah jenis keping pecahan yang tersedia), sehingga pemrosesan nominal bernilai besar (misalnya Rp1.000.000) akan memaksa program untuk mengalokasikan memori array sebesar satu juta elemen (Cormen et al., 2022).

Kondisi inefisiensi ruang inilah yang mendasari urgensi mengapa Algoritma Greedy, yang kompleksitasnya hanya bergantung pada jumlah jenis pecahan (O(n)), menjadi alternatif yang jauh lebih superior pada kasus-kasus khusus tertentu.

Sistem pecahan uang Rupiah yang berlaku di Indonesia saat ini terdiri atas uang kertas dan uang logam dengan variasi pecahan Rp100, Rp200, Rp500, Rp1.000, Rp2.000, Rp5.000, Rp10.000, Rp20.000, Rp50.000, dan Rp100.000. Jika diamati dengan saksama, struktur kesepuluh jenis pecahan tersebut membentuk suatu pola yang menyerupai deret 1-2-5.

Pola deret geometris modifikasi ini bukanlah suatu kebetulan; struktur matematis 1-2-5 sengaja dirancang dan diadopsi oleh berbagai bank sentral di seluruh dunia, termasuk sistem dolar Amerika Serikat (1¢, 5¢, 10¢, 25¢, 50¢, $1) dan mata uang Euro. Karakteristik struktural inilah yang menjadi landasan rasional mengapa sistem pecahan Rupiah dipilih sebagai objek pengujian utama pada penelitian ini. Selain sangat relevan dengan kebutuhan sistem transaksi di Indonesia, sistem Rupiah juga sangat representatif mewakili arsitektur sistem pecahan uang standar yang diakui dan digunakan secara global.

Tinjauan terhadap penelitian terdahulu menunjukkan bahwa kajian mengenai penerapan Algoritma Greedy pada Coin Change Problem telah dieksplorasi dari berbagai sudut pandang. Sunandar dan Pristiwanto (2019) telah mengimplementasikan Algoritma Greedy

untuk mengoptimalkan modul penukaran mata uang Rupiah menggunakan bahasa pemrograman berbasis aplikasi visual.

Penelitian mereka berhasil menunjukkan secara pragmatis bahwa algoritma tersebut mampu menghasilkan kombinasi pecahan secara efisien pada sistem mata uang di Indonesia. Pada tataran literatur internasional, Mehmood (2019) mengusulkan sebuah varian optimasi dari Algoritma Greedy yang didesain memiliki kompleksitas waktu spesifik O(m×f). Hasil studinya membuktikan bahwa varian Greedy tersebut jauh lebih efisien dan mengeksekusi iterasi lebih sedikit dibandingkan pendekatan standar Dynamic Programming pada sejumlah data kasus uji.

Meskipun demikian, studi Mehmood (2019) juga dengan tegas memberikan peringatan bahwa keunggulan kecepatan Algoritma Greedy memiliki celah kelemahan fatal, yakni tidak selalu menghasilkan solusi dengan jumlah pecahan minimum (optimal) jika diterapkan pada sembarang sistem himpunan bilangan acak.

Dari perspektif teori komputasi fundamental, Kozen dan Zaks (1994) serta Pearson (2005) telah meletakkan landasan matematis yang sangat penting dengan merumuskan konsep sistem pecahan kanonik (canonical coin system). Definisi matematis dari sistem pecahan kanonik adalah suatu himpunan denominasi nilai di mana Algoritma Greedy secara absolut dijamin akan selalu memproduksi jumlah total pecahan minimum untuk representasi setiap bilangan bulat positif (nominal) apapun.

Penelitian eksploratif lanjutan yang dilakukan oleh Cai (2008) berhasil memperluas karakterisasi pembuktian matematis sistem kanonik ini hingga pada himpunan yang terdiri dari lima jenis pecahan. Di era komputasi modern yang lebih mutakhir, Gupta, Huang, dan Impagliazzo (2024) menelaah secara spesifik mengenai kompleksitas komputasi bawaan dari proses simulasi Algoritma Greedy itu sendiri. Kajian terbaru mereka mereafirmasi postulat bahwa, meskipun Algoritma Greedy tidak memiliki sifat optimalitas universal seperti DP, algoritma ini secara ajaib tetap berstatus optimal jika diaplikasikan pada hampir seluruh sistem mata uang riil yang diformulasikan oleh negara-negara di dunia saat ini.

Meskipun telah banyak kajian literatur yang tersedia, terdapat sebuah celah penelitian (research gap) yang cukup signifikan. Sebagian besar penelitian implementasi (seperti Sunandar & Pristiwanto, 2019) cenderung berfokus murni pada hasil akhir rekayasa perangkat lunak tanpa melakukan pengujian empiris analitik yang mengukur seberapa efisien algoritma tersebut berjalan di level perangkat keras (hardware level benchmarking).

Di sisi lain, penelitian-penelitian teoritis (seperti Cai, 2008; Gupta et al., 2024) cenderung membahas pembuktian matematis secara abstrak tanpa menyajikan data eksekusi nyata pada sistem arsitektur komputer modern. Selain itu, penelitian yang secara langsung menyandingkan dan mengevaluasi kompleksitas praktis (waktu komputasi aktual dalam milidetik) dan dampak alokasi memori secara komparatif head-to-head antara Algoritma Greedy dan Dynamic Programming pada lingkungan eksperimen terkontrol (khususnya untuk pecahan uang Rupiah vs Non-Kanonik) masih belum terpetakan dengan baik.

Berdasarkan pemaparan kesenjangan tersebut, penelitian ini diformulasikan untuk menjembatani kajian teoritis mengenai sistem kanonik dengan implementasi empiris tingkat rendah (low-level implementation). Penelitian ini mengusung kebaruan dengan berfokus pada evaluasi komparatif performa nyata (empirical benchmarking) dengan kontrol ketat terhadap faktor arsitektur sistem seperti optimasi kompiler dan interupsi cache prosesor. Implementasi sengaja dilakukan

menggunakan bahasa pemrograman terstruktur Object Pascal dengan struktur data larik (array) statis. P

emilihan Object Pascal didasarkan pada karakteristik sintaksnya yang prosedural dan minim overhead eksekusi dinamis (berbeda dengan Python atau Java yang berjalan di atas mesin virtual/interpreter), sehingga kemurnian logika dan kecepatan asli dari algoritma dapat diukur dengan lebih akurat (Jensen & Wirth, 1991).

Penelitian ini menetapkan tiga tujuan utama: (1) menganalisis secara komparatif efektivitas dan kompleksitas praktis (waktu eksekusi empiris) Algoritma Greedy berbanding Dynamic Programming dalam menyelesaikan Coin Change Problem menggunakan Object Pascal; (2) mengevaluasi efisiensi operasional dari penerapan struktur data larik sebagai fondasi memori (memory profiling); dan (3) membuktikan secara empiris batasan teori sistem pecahan kanonik dengan menguji algoritma pada sistem Rupiah dan menantangnya dengan sistem non-kanonik artifisial.

Luaran dari penelitian ini diharapkan dapat memberikan kontribusi literatur substansial mengenai batas-batas reliabilitas komputasi algoritmik, serta memberikan panduan saintifik bagi para insinyur perangkat lunak (software engineer) dalam merancang modul transaksi pada sistem tertanam (embedded systems).

METODE

Desain rancangan pada penelitian ini menggunakan metode eksperimen komparatif kuantitatif murni. Metode ini diimplementasikan untuk memastikan bahwa perbandingan kinerja antara Algoritma Greedy dan Dynamic Programming (DP) didasarkan pada data metrik kuantitatif yang terukur secara presisi, bukan berdasarkan asumsi teoretis semata. Proses penelitian direkayasa melalui tahapan sistematis yang meliputi studi literatur komprehensif mengenai konsep optimasi, perancangan diagram logika algoritma, translasi logika ke dalam sintaks bahasa Object Pascal, fase pelaksanaan pengujian komputasi (benchmarking), hingga analisis statistika deskriptif terhadap hasil luaran.

Berbeda dari desain penelitian perangkat lunak pada umumnya yang sekadar melihat output layar (GUI), parameter efektivitas pada eksperimen ini didefinisikan secara ketat pada empat metrik dimensi: (1) ketepatan absolut hasil kalkulasi jumlah minimum pecahan, (2) rata-rata waktu eksekusi empiris (runtime) dalam resolusi milidetik, (3) estimasi jejak penggunaan ruang memori operasional (space complexity profiling), dan (4) tingkat persistensi konsistensi hasil terhadap Dynamic Programming yang ditunjuk sebagai standar acuan kebenaran mutlak (ground truth baseline).

Tahap perancangan logika algoritmik untuk pendekatan Greedy diorkestrasi secara berurutan. Langkah krusial pertama adalah inisialisasi basis data himpunan koin. Himpunan pecahan uang yang akan diproses direpresentasikan menggunakan struktur data larik statis berdimensi tunggal (1D Array). Sifat imperatif dari Algoritma Greedy mewajibkan larik ini diurutkan secara menurun secara mutlak (descending order) dari nilai mata uang terbesar hingga terkecil.

Pengurutan ini berfungsi sebagai pra-kondisi matematis agar algoritma tidak perlu melakukan pencarian (searching) yang memakan waktu pada setiap iterasi. Pada inti perulangannya, algoritma akan membaca nilai input nominal, melakukan iterasi pointer dari indeks pertama larik, dan memverifikasi apakah denominasi pada indeks tersebut memiliki nilai yang kurang dari atau sama dengan nominal yang tersisa. Jika kondisi terpenuhi, program mengeksekusi operasi aritmatika pembagian bilangan bulat

(operator div dalam Pascal) untuk mencetak kuantitas lembar/keping yang digunakan.

Segera setelah itu, operasi modulo (operator mod) diaplikasikan untuk mereduksi nominal menjadi sisa tagihan, yang kemudian dilempar ke siklus iterasi pecahan berikutnya. Seluruh tahapan iteratif ini dirancang tanpa adanya pemanggilan fungsi secara rekursif demi mencegah fenomena penumpukan memori panggilan (stack overflow) serta mengamankan jaminan kompleksitas ruang pada konstan O(1).

Sebagai instrumen pembanding, algoritma Dynamic Programming didesain dengan model penyelesaian unbounded coin change minimum (di mana persediaan setiap koin diasumsikan tak terbatas). DP diimplementasikan dengan strategi tabulasi bergerak naik (bottom-up tabulation approach).

Logika DP memecahkan masalah dengan membuat sebuah array berdimensi (Nominal + 1), yang lazim disebut sebagai tabel dp. Tabel dp[0] diinisialisasi dengan angka 0, yang secara logis merepresentasikan bahwa dibutuhkan 0 koin untuk membentuk nilai nominal 0. Sisa indeks tabel mulai dari 1 hingga V (V = besar nilai Nominal) diinisialisasi sementara dengan konstanta infinity (nilai tak terhingga).

Program kemudian menggunakan dua lapis struktur pengulangan (nested loop): loop eksternal iterasi secara inkremental dari 1 hingga V, sementara loop internal mengevaluasi seluruh jenis pecahan n yang tersedia. Melalui penerapan persamaan state transition dari prinsip Bellman (dp[i] = min(dp[i], dp[i - koin] + 1)), algoritma secara progresif membangun pondasi solusi optimal dari angka 1 hingga bermuara pada nilai target V. Desain komputasi DP ini dipastikan kebal terhadap anomali sistem non-kanonik, namun secara teoretis mengikat program pada kompleksitas waktu kuadratik O(nV) dan kompleksitas ruang linear O(V).

Rancangan skenario pengujian komparatif (benchmarking) diatur melalui protokol eksperimen yang ketat guna mengeleminasi variabel pengganggu (confounding variables). Seluruh instrumen kode program ditulis, dikompilasi, dan dieksekusi menggunakan Free Pascal Compiler (FPC) versi 3.2.2. Untuk memastikan potensi tersembunyi dari algoritma terekstraksi maksimal, pengaturan kompilasi dipaksa menggunakan argumen optimasi level 2 (-O2), sebuah fitur instruksi kepada compiler untuk merapikan alur logika tingkat rendah pada bahasa assembly. Instrumen pengukur waktu yang digunakan adalah fungsi pemanggil waktu presisi dari sistem operasi (SysUtils.GetTickCount64).

Metode pengukuran mengadopsi prosedur repeated-trial benchmarking (pengujian berulang-ulang beruntun). Mekanisme ini dipilih karena durasi satu putaran eksekusi Algoritma Greedy terlampau instan (berkisar pada skala nanodetik hingga mikrodetik dasar), sehingga luput dari jangkauan presisi jam standar sistem operasi jika hanya diukur sekali jalan. Oleh karenanya, untuk Algoritma Greedy, sistem diperintahkan menjalankan perulangan buta pemrosesan fungsi sebanyak 1.000.000 (satu juta) repetisi. Total waktu akumulasi yang dicatat pada akhir perulangan kemudian dibagi kembali dengan 1.000.000 untuk menyaring dan menetapkan angka rata-rata waktu per pemanggilan tunggal.

Teknik saturasi perulangan masif ini memiliki keuntungan metodologis krusial yakni mengamankan instruksi program agar sepenuhnya terkunci di dalam memori cache L1/L2 pada arsitektur prosesor (cache hit), sehingga bias keterlambatan yang berasal dari akses RAM (cold cache miss latency) dan interupsi jadwal latar belakang (background scheduling) dari sistem operasi dapat direduksi secara signifikan. Berkebalikan dengan Greedy, eksekusi

pembanding algoritma Dynamic Programming hanya direpetisi sebanyak 30 kali iterasi per pemanggilan fungsi. Penetapan kuota 30 repetisi ini dilandasi oleh karakteristik pertumbuhan waktu algoritma DP yang meroket eksponensial saat memproses angka nominal skala besar; perulangan satu juta kali akan menyebabkan sistem mengalami kebuntuan komputasi (hang). Sesuai dengan Teorema Limit Pusat (Central Limit Theorem) dalam statistika eksperimental, ukuran sampel (n) sebanyak 30 iterasi telah dipandang cukup stabil dan representatif untuk menarik garis nilai rata-rata eksekusi (mean runtime) dari program berbobot komputasi berat tanpa merusak integritas perangkat keras penelitian.

Populasi objek data input yang dieksperimenkan diklasifikasikan menjadi dua domain skenario uji. Skenario eksperimen pertama mengevaluasi kinerja algoritma pada ekosistem mata uang konvensional (Sistem Pecahan Rupiah). Sepuluh titik uji (test cases) data nilai nominal ditetapkan secara terstruktur dan purposif untuk merangkum representasi siklus pembayaran di dunia nyata: kategori nominal kecil diwakili oleh Rp5.000, Rp12.500, dan Rp25.700; kategori transaksi ritel menengah diwakili oleh Rp78.500, Rp123.400, dan Rp250.000; serta kategori transaksi penarikan volume besar (ATM/korporasi) diwakili oleh Rp375.600, Rp500.000, Rp725.300, dan angka semu batas maksimum batas uang receh yakni Rp999.900. Skenario eksperimen kedua ditujukan secara khusus sebagai arena validasi stres (stress test validation) terhadap batas kelemahan (vulnerability) heuristik Algoritma Greedy. Pada skenario ini, aturan hukum ekonomi riil disingkirkan, digantikan dengan injeksi himpunan sistem pecahan hipotetis (buatan) yang sengaja dirancang menentang sifat matematis kanonik, yakni himpunan set koin bernilai {1, 3, 4}.

Rentang uji untuk eksperimen artifisial ini dilakukan dengan skala nominal sempit yang bergerak sekuensial (berurutan) mulai dari angka 1 hingga angka 15, untuk memetakan secara mikroskopis di titik mana persisnya pengambilan keputusan lokal (Greedy Choice Property) mengalami penyimpangan rute yang berujung pada hilangnya kemampuan menyentuh titik konvergensi optimal (solusi minimal).

HASIL DAN PEMBAHASAN

Hasil

Eksperimentasi implementasi kode program Object Pascal mengonfirmasi bahwa mesin logika dari Algoritma Greedy mampu beroperasi dengan sempurna memecahkan problematika pembentukan kombinasi koin. Pada domain Skenario Pertama (Sistem Pecahan Rupiah), seluruh 10 rentang variasi input nilai nominal yang direpresentasikan ke dalam algoritma berhasil diurai dengan rasio kegagalan identifikasi sebesar 0%. Luaran rekam data matriks komparasi jumlah pecahan berserta rekaman stop-watch komputasi empirik disarikan secara komprehensif pada Tabel 1.

Tabel 1. Komparasi Empiris Ketepatan Hasil dan Waktu Eksekusi (Greedy vs Dynamic Programming) pada Sistem Nominal Rupiah

Nilai Nominal Uji (Rp)	Jml Kombinasi Pecahan (Greedy)	Jml Kombinasi Pecahan (DP)	Waktu Eksekusi Empiris Greedy (ms)	Waktu Eksekusi Empiris DP (ms)
5.000	1	1	0,000079	0,066667
12.500	3	3	0,000077	0,200000
25.700	4	4	0,000078	0,300000
78.500	6	6	0,000080	1,466667

123.400	6	6	0,000079	1,533333
250.000	3	3	0,000078	3,500000
375.600	8	8	0,000082	5,633333
500.000	5	5	0,000078	7,300000
725.300	11	11	0,000078	10,733333
999.900	18	18	0,000081	15,000000

Membedah anatomi angka-angka yang disajikan pada Tabel 1, hasil investigasi kuantitatif menunjukkan temuan pertama bahwa tidak terdapat selisih satu keping pun pada rekomendasi keluaran Algoritma Greedy jika dihadapkan pada hasil benchmark absolut yang dicetak oleh Dynamic Programming (DP). Pada nominal acak Rp123.400 misalnya, baik metode Greedy maupun DP sepakat secara identik mengeluarkan hasil mutlak sebanyak 6 lembar pecahan (berisi perincian komposisi: 1 lembar Rp100.000, 1 lembar Rp20.000, 1 lembar Rp2.000, 1 keping Rp1.000, dan 2 keping Rp200).

Kasus dengan tumpukan variasi pecahan terlebar terpantau pada nominal maksimum Rp999.900, di mana algoritma berhasil melakukan pemetaan sebanyak 18 buah item mata uang yang melibatkan pengulangan (redundansi) pecahan bernilai tinggi seperti 9 lembar seratus ribuan.

Totalitas ekuivalensi (kesamaan fungsi) 100% ini menjadi pembuktian empirik valid mengenai keandalan heuristik Greedy dalam domain mata uang Rupiah standar. Sorotan utama yang mencolok dari eksperimen komparatif ini mewujud secara nyata pada matriks pengukuran waktu eksekusi (runtime).

Kurva pertumbuhan waktu dari rutinitas Dynamic Programming mendemonstrasikan sifat korelasional positif linier-eksponensial terhadap besaran beban nominal yang dibawanya. Dimulai dari angka eksekusi moderat 0,066667 milidetik pada penukaran kecil Rp5.000, DP tercekik oleh bebannya sendiri seiring inflasi nominal, sehingga waktu rambat kalkulasinya bengkak menyentuh angka 15,000000 milidetik saat dieksekusi menangani nominal Rp999.900.

Peristiwa pembengkakan beban komputasi ini terisolasi dan berbanding terbalik 180 derajat jika dihadapkan dengan rekam profil Algoritma Greedy. Dalam lanskap sistem berbasis Greedy, waktu yang dihabiskan prosesor untuk menuntaskan komputasi berjalan secara stagnan vertikal dan stabil statis di kisaran 0,000077 hingga maksimal 0,000082 milidetik per satu siklus pemanggilan.

Jika ditarik skala banding rasional, selisih performa pada pemrosesan beban tertinggi di Rp999.900 menunjukkan bahwa konstruksi Algoritma Greedy bermuatan Object Pascal menembus performa kecepatan sekitar 185.000 (seratus delapan puluh lima ribu) kali lipat lebih superior dibanding saudaranya DP. Stagnansi fluktuasi waktu ini merupakan refleksi langsung dari tidak terpengaruhnya algoritma terhadap inflasi besaran nominal.

Pada fase Skenario Eksperimen Kedua, arena simulasi dipindahkan pada domain Sistem Pecahan Non-Kanonik Sintetis dengan set pecahan {1, 3, 4}. Uji validitas menembakkan 15 tembakan sekuensial dengan rentang nominal angka murni 1, 2, 3... hingga 15. Dari hasil pembacaan output monitor, ditemukan keretakan fatal pada integritas pengambilan keputusan (decision logic fault) Algoritma Greedy yang gagal beradaptasi.

Dari semesta data berjumlah 15 uji tersebut, 12 nominal berhasil dipecahkan dengan rasio setara (kembar) dengan DP

(angka 1 hingga 5, serta angka 7, 8, 9, 11, 12, 13, 15). Namun, anomali keterbelakangan kalkulasi (sub-optimal) menetas pada 3 titik nominal krusial, yang diterjemahkan menjadi angka rasio margin kesalahan sebesar 20% cacat prosedur matematis. Detail breakdown mikroskopis kegagalan algoritma terlihat pada Tabel 2 berikut.

Tabel 2. Analisis Kasus Kegagalan Optimalitas Algoritma Greedy pada Sistem Non-Kanonik {1, 3, 4}

Nilai Target Nominal	Output Kombinasi Greedy	Total Item (Greedy)	Output Kombinasi Optimal (DP)	Total Item (Optimal)	Status Evaluasi
6	4 + 1 + 1	3 Keping	3 + 3	2 Keping	Gagal (Sub-Optimal)
10	4 + 4 + 1 + 1	4 Keping	4 + 3 + 3	3 Keping	Gagal (Sub-Optimal)
14	4 + 4 + 4 + 1 + 1	5 Keping	4 + 4 + 3 + 3	4 Keping	Gagal (Sub-Optimal)

Sebagaimana divisualisasikan pada pembedahan log Tabel 2, kelemahan mendasar Algoritma Greedy terpampang nyata. Ketika sistem disuapi target nominal '6', insting serakah (Greedy Choice Property) secara impulsif memungut koin terbesar yang tersedia tanpa pikir panjang, yaitu koin bernilai '4'. Transaksi pemungutan ini menggerus nominal tersisa menjadi '2'.

Selanjutnya, program terkunci dari mengakses koin '3' karena nilai sisa tagihan lebih kerdil (2 < 3). Secara terpaksa, mesin melengkapi keping-keping sisanya dengan serapan dua buah koin receh '1'. Mekanisme ini memuntahkan komposisi akhir berupa konfigurasi [4, 1, 1] yang menghabiskan slot kapasitas sebanyak 3 barang. Perbandingan terbalik terjadi pada eksekusi iteratif Dynamic Programming.

Mesin pencari jalan DP tidak gegabah mengeksekusi penarikan memori instan; ia menyusuri rute probabilitas dan menyadari bahwa pengambilan dua koin menengah [3, 3] akan menutup tagihan bernilai '6' dengan kuota konsumsi yang jauh lebih elegan dan minimalis (sebesar 2 barang saja). Terjebaknya mesin pencari pada ilusi efisiensi lokal ini menjadi bukti kuat yang tidak terbantahkan (indisputable evidence) bahwa keanggunan kecepatan Algoritma Greedy tidak bersifat maha jaya (omnipotent), dan sangat mudah dikelabuhi oleh komposisi set nilai kanonikal yang anomali.

Pembahasan

Kejayaan absolut Algoritma Greedy dalam menuntaskan 100% skenario konversi set mata uang Rupiah tanpa galat komputasi sedikitpun bukan merupakan sebuah mukjizat kebetulan di dalam ilmu sains komputer.

Ekuivalensi optimalitas luaran Greedy dengan patokan emas Dynamic Programming mengekstraksi kesimpulan mutlak bahwa struktur hierarki cetakan mata uang di Indonesia seutuhnya dilahirkan mengikuti pedoman tata krama matematis yang dinamakan sebagai sistem kanonik (canonical system).

Pemenuhan kaidah kriteria kanonik ini—yang sejatinya merupakan turunan modifikasi dari struktur deret sakral 1-2-5—menciptakan ilusi harmoni algoritma, di mana keputusan pengambilan secara instan

pada satu tahap spesifik (titik pandang lokal) selalu dikaruniai rute yang bersambung dengan garis takdir optimasi di garis finish (titik pandang global). Verifikasi kebenaran lapangan (empirical validation) ini memperkokoh tiang postulat fundamental Canonical Coin System yang pertama kali dipahat oleh akademisi Kozen dan Zaks (1994), dan turut mengamini ekspansi hipotesis lanjutannya yang disuarakan oleh pakar matematika komputasional Cai (2008).

Diskursus pembahasan yang jauh lebih esensial, bagaimanapun, terhampar di balik layar panggung evaluasi pengukuran instrumen praktis, terkhusus pada pembedahan kompleksitas praktikal arsitektural (practical complexity analysis) dan pemetaan tapak jejak memori semikonduktor yang terpakai selama durasi program bernapas (memory space profiling).

Penyusunan arsitektur karya tulis ilmiah yang komprehensif ini sengaja menyusun strategi penerapan struktur data berbasis larik statik dimensi tunggal (single-dimensional static array) pada mesin compiler Object Pascal, sebuah desain purba yang justru merestui kecepatan akses memori indeks ke titik rasio tak terkalahkan pada batas Big-O Notation O(1) di tahap pengaksesan sekuensial yang konsisten statis (Cormen et al., 2022).

Manfaat kolosal dari penyederhanaan ruang variabel penyimpan ini menjadikan kompleksitas ruang beban (space complexity) dari algoritma heuristik Greedy terjebak pada angka O(n)—di mana parameter konstanta n sebatas merepresentasikan banyaknya ragam tipe kertas uang yang beredar (dalam hal ini, 10 jenis uang Rupiah).

Hal ini mengindikasikan bahwa baik saat konsumen menarik tunai Rp5.000 maupun saat pengusaha mencairkan giro bilyet bernilai triliunan rupiah (jika ditambahkan angka nolnya), kapling RAM komputer yang dimonopoli oleh proses algoritma sama sekali tidak bergeser satu bit pun, alias stagnan mutlak.

Paradigma supremasi pergerakan konstan pada algoritma Greedy ini akan tampak bagaikan sebuah ketelanjangan arsitektur apabila dikontraskan tanpa ampun dengan kerumitan mesin pembedah absolut bernama Dynamic Programming. Mesin tabulasi bottom-up pada tulang punggung rute algoritma DP mengoperasikan mesin raksasa yang menuntut sesajen suplai pembentukan alokasi array dinamis elastis dengan panjang ekuivalen seukuran nominal input (dilambangkan secara teoritis sebagai notasi ruang dinamis O(V), dengan nilai V mewakili raksasa beban input).

Pada saat instrumen riset ini menekan tombol pelatuk eksekusi pemrosesan target beban maksimum senilai Rp999.900, program berplatform algoritma DP secara tiran tanpa memedulikan etika perangkat keras, secara paksa memerintahkan sirkuit logika untuk menyusun cetakan kapling, menginisiasi, menyortir memori, lalu menghamburkan set penyimpanan array memori dinamis hingga hampir satu juta indeks unit submasalah di dalam bilik Random Access Memory (RAM).

Bencana banjir antrean paket memori acak ke dalam lorong cache L1 dan cache L2 pada die (lempengan) sirkuit CPU komputer inilah yang menelurkan gesekan sinkronisasi laten memori parah (cache thrashing rate dan cold miss degradation). Rentetan kecelakaan birokratis pada tingkatan pertukaran sinyal arus listrik prosesor tingkat rendah inilah yang sejatinya menabur benih ketimpangan rasio perlambatan durasi stopwatch hingga skala kelipatan 185.000 (seratus delapan puluh lima ribu) kali lebih memble jika dihadapkan pada kedigdayaan respons instan kedipan mata algoritma ksatria tunggal Greedy (Gupta, Huang, & Impagliazzo, 2024).

Penyajian algoritma yang disusun secara terstruktur di dalam kerangka

compiler Object Pascal menyisipkan relevansi pembelajaran pedagogi informatika yang tidak kalah penting untuk didiskusikan. Kesederhanaan murni manipulasi sisa bagi (modulus) dan hasil bagi iteratif (div integer) mendemonstrasikan fenomena independensi algoritma di atas portabilitas universal sintaks.

Meskipun diimplementasikan menggunakan bahasa legendaris seperti Pascal, kerangka logika murni bebas objek dinamis yang dibungkus Algoritma Greedy ini dapat diderivasi secara persis (identik) ke dalam ekosistem kerangka bahasa industri modern masa kini (misalnya C/C++, Rust, Java, atau Golang) dan dipastikan akan menelurkan kurva waktu eksekusi linear datar yang sangat sejalan dan berimbang dengan raihan angka di studi eksperimental ini.

Temuan ini merangkum sebuah resolusi konklusif untuk domain sistem tersemat industri digital, seperti perakitan papan tunggal Raspberry Pi pada perangkat kiosk digital Vending Machine atau sirkuit otak arloji pembayaran pintar NFC, Algoritma Greedy tidak lagi sekadar menjadi alternatif; algoritma ini merajai takhta sebagai opsi fundamental yang sanggup memperpanjang sisa nyawa ketahanan baterai semikonduktor (hemat daya baterai).

Namun, kesombongan rasio O(1) ini harus diiringi dengan ketaatan hukum syariat matematis. Seperti yang telah dibuktikan pada Tabel 2, saat hukum pecahan kanonik dirusak secara sengaja (sebagaimana ilustrasi kecelakaan tragis kombinasi sub-optimal pada rute titik nominal {6}, {10}, dan {14} di matriks artifisial {1,3,4}), mata rantai prediksi masa depan Greedy menjadi lumpuh total (blindspot vulnerability).

SIMPULAN (PENUTUP)

Berdasarkan tahapan metode pengujian yang komprehensif, telaah analisis angka empiris, dan sinkronisasi sintesis teori, dapat diikat dalam konklusi tegas bahwa penerapan Algoritma Greedy memiliki efektivitas kinerja yang melampaui dan mengungguli performa superior dari Dynamic Programming manakala sistem algoritma ini ditugaskan untuk menghancurkan permasalahan kombinasi koin pada arsitektur sistem pecahan mata uang standar yang diadopsi Republik Indonesia (Rupiah).

Algoritma Greedy terbukti secara sahih menorehkan angka reproduksi ekuivalensi (ketepatan) identik seratus persen (100%) terhadap patokan emas (golden standard) yang diciptakan oleh penyelesaian Dynamic Programming pada sepuluh data sampel nyata yang bervariasi luas. Pembedahan metrik komparasi pengukuran kecepatan (benchmarking speed profiling) memuntahkan deretan angka fantastis yang membuktikan Algoritma Greedy menghemat waktu penantian pengguna dengan menembus zona akselerasi rata-rata hingga 185.000 kali lebih gesit membelah waktu eksekusi jika dikomparasi secara timpang pada penanganan muatan target titik puncak Rp999.900, disusul keunggulan absolut berupa minimasi penjajahan rakus ruang RAM sekunder lantaran konsistensi status hierarki konstan struktur kompleksitas memori primitif O(1).

Namun, supremasi kejayaan empirik ini tunduk bertekuk lutut dan berhasil divalidasi kelemahannya melalui skenario stres pengujian pada alam mikrokosmos sistem himpunan koin fiktif tak lazim bersistem non-kanonik artifisial {1, 3, 4}. Kerangka kalkulasi rasional dari Algoritma heuristik Greedy terjebak di dalam pusaran keputusan sepihak (tunnel vision defect) yang memporak-porandakan kemampuannya dalam membidik batas rasio irit kuantitas terbawah (fenomena defisiensi titik potong lokal terhadap titik global optimal) yang mewabah dan menjangkiti sekitar 20% populasi total bilangan real (3 titik kegagalan

di dalam 15 uji urutan angka uji nominal dasar).

Sintesa deduksi menyeluruh dari rute argumen saintifik ini menghadirkan sebuah konfirmasi final penegasan yang tidak dapat dianulir: bahwa sumbangsih luar biasa dan utilitas superlatif Algoritma Greedy pada reduksi tajam terhadap masa tayang respon aplikasi dan reduksi interupsi beban sirkuit semikonduktor, selamanya terkekang dan disandera tanpa ampun oleh syarat fundamental mutlak eksistensi desain kanonikal.

Menyongsong agenda arah perjalanan ekspedisi keilmuan teknologi algoritma optimasi nusantara pada esok hari, desain proposal riset jurnal ini merekomendasikan pelebaran sayap studi dengan fokus penyisipan agenda pengukuran jejak profil silikon memori tingkat mikro-ekstrem (on-board memory profiling analysis & benchmarking validation test) yang dilakukan secara harafiah di atas plat sirkuit fisik (motherboard) peranti kerdil Internet of Things (Sistem IoT) yang minim daya listrik.

Selain itu, rekomendasi komparasi triangulasi perbandingan algoritma yang turut mengundang algoritma elit pencari cabang seperti keluarga Branch and Bound dan sistem koloni semut (Ant Colony Optimization) perlu dipertimbangkan masak-masak sebagai penantang baru dalam usaha penelusuran tak henti demi mengekstrak solusi algoritma pengurai kembalian kasir modern terbaik di muka bumi.

UCAPAN TERIMA KASIH

Tim penulis artikel penelitian menghaturkan ungkapan puji syukur setinggi-tingginya dan deretan ucapan terima kasih yang tulus, tidak terhingga, dan apresiasi yang paling murni kepada institusi kebanggaan, yaitu Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Muhammadiyah Parepare. Dedikasi ini tidak mungkin terwujud tanpa aliran dukungan moril berkesinambungan, pemberian bimbingan dan akses izin penggunaan seluas-luasnya terhadap fasilitas infrastruktur laboratorium riset piranti keras universitas, serta pembentukan pengadaan atmosfer iklim kebebasan akademik yang kondusif, progresif, dan menunjang keilmuan secara utuh selama melangkah di jalan proses panjang penelitian empirik, penulisan manuskrip berulang kali, hingga sampai pada tahap penyusunan matang publikasi naskah komparasi perbandingan struktural ini selesai dipublikasikan dan layak dinikmati pembaca.

DAFTAR PUSTAKA

Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1983). *Data structures and algorithms*. Addison-Wesley.

Cai, X. (2008). *Canonical coin systems for change-making problems*. arXiv:0809.0400. from https://arxiv.org/abs/0809.0400

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

Gupta, S., Huang, B., & Impagliazzo, R. (2024). *The greedy coin change problem*. arXiv:2411.18137. from https://arxiv.org/abs/2411.18137

Jensen, K., & Wirth, N. (1991). *Pascal user manual and report*. Springer.

Kozen, D., & Zaks, S. (1994). Optimal bounds for the change-making problem. *Theoretical Computer Science*, 123(2), 377–388.

Mehmood, A. (2019). ASH CC Algo: Coin change algorithm optimization. *International Journal of Computer Applications*, 178(15), 21–25.

Pearson, D. (2005). A polynomial-time algorithm for the change-making problem. *Operations Research Letters*, 33(3), 231–234.

Sunandar, H., & Pristiwanto. (2019). Optimalisasi implementasi algoritma greedy dalam fungsi penukaran mata uang Rupiah. *Jurnal Teknik Informatika UNIKA Santo Thomas*, 4(2), 193–201.