SABER: Jurnal Teknik Informatika, Sains dan Ilmu Komunikasi
Vol.2, No.2 April 2024
e-ISSN: 3025-342X; p-ISSN: 3025-2776, Hal 283-294
DOI: https://doi.org/10.59841/saber.v2i2

Implementasi Algoritma Greedy Dalam Penukaran Uang Di Alfamart Di Kota Dan Algoritma String Matching Untuk Pencarian Cabang Alfamart Di Kota-Kota Di Indonesia

Tasya Apriliani[1], Athay Setya Dwi Putri[2], Jovanka Feranita[3], Meisya Pradana Mentari[4]

[1]Program Studi S1Teknik Informatika, Universitas Palangkaraya

Abstrak

Algoritma Greedy dan algoritma pencocokan string (*string matching*) adalah dua pendekatan penting dalam komputasi yang memiliki aplikasi luas dalam pemrosesan teks dan optimasi. Dalam penelitian ini, kami menjelaskan konsep, sejarah, dan penerapan kedua algoritma tersebut. Algoritma Greedy memilih solusi terbaik pada setiap langkah dengan harapan mencapai solusi global yang optimal, sedangkan algoritma pencocokan string digunakan untuk mencari kemunculan suatu pola dalam teks dengan efisien. Kami mengevaluasi implementasi algoritma Greedy dalam penukaran uang di Alfamart dan algoritma pencocokan string untuk pencarian cabang Alfamart di berbagai kota di Indonesia. Metode penelitian dievaluasi berdasarkan kelengkapan, optimalitas, dan kompleksitas waktu. Hasil menunjukkan bahwa implementasi algoritma Greedy efisien dalam mengelola proses penukaran uang dengan waktu eksekusi yang stabil. Sementara itu, algoritma pencocokan string menawarkan kemampuan untuk mencari cabang Alfamart dengan cepat dan efisien meskipun menunjukkan variasi dalam kinerjanya. Kesimpulannya, kedua algoritma ini memiliki potensi besar untuk meningkatkan efisiensi operasional dan kenyamanan pengguna dalam hal penukaran uang dan pencarian cabang minimarket. Studi ini memberikan kontribusi penting dalam memahami dan menerapkan algoritma Greedy dan pencocokan string dalam kasus nyata, dengan implikasi untuk pengembangan solusi yang lebih efisien dan optimal di masa depan.

Kata kunci: Algoritma, Algoritma Greedy. String Matching, Penukaran Uang.

Abstract

Greedy algorithm and string matching algorithm are two essential approaches in computing with wide-ranging applications in text processing and optimization. In this research, we elucidate the concepts, history, and applications of both algorithms. The Greedy algorithm selects the best solution at each step in hopes of achieving the optimal global solution, while the string matching algorithm is used to efficiently locate occurrences of a pattern within text. We evaluate the implementation of the Greedy algorithm in currency exchange at Alfamart and the string matching algorithm for finding Alfamart branches in various cities in Indonesia. The research methods are assessed based on completeness, optimality, and time complexity. The results demonstrate that the Greedy algorithm implementation efficiently manages the currency exchange process with stable execution times. Meanwhile, the string matching algorithm offers the ability to swiftly and efficiently locate Alfamart branches despite showing variations in performance. In conclusion, both algorithms have significant potential to enhance operational efficiency and user convenience in the context of currency exchange and minimarket branch search. This study provides a crucial contribution to understanding and applying Greedy and string matching algorithms in real-world cases, with implications for the development of more efficient and optimal solutions in the future.

***Keywords**: Algorithm, Greedy Algorithm, String Matching, Currency Exchange*.

Received Februari 06, 2024; Accepted Maret 06, 2024; Published April 30, 2024
* Tasya Apriliani

PENDAHULUAN

Algoritma merupakan langkah-langkah sistematis untuk menyelesaikan masalah dalam komputasi dan matematika. Dalam banyak kasus, terdapat beragam jenis algoritma yang dapat digunakan untuk menyelesaikan suatu masalah tertentu. Dua di antaranya adalah algoritma Greedy dan algoritma pencocokan string (string matching). Dalam pendahuluan ini, penulis akan menjelaskan sejarah dan memberikan gambaran singkat tentang kedua algoritma tersebut serta relevansinya dalam berbagai bidang.

Algoritma Greedy, atau sering disebut juga dengan pendekatan "serakah", adalah salah satu paradigma dalam desain algoritma yang memilih solusi yang terlihat terbaik pada setiap tahap dengan harapan bahwa pemilihan tersebut akan mengarah pada solusi global yang optimal. Meskipun Greedy cenderung menghasilkan solusi yang dekat dengan optimal dalam banyak kasus, namun tidak selalu menjamin solusi yang benar-benar optimal dalam hal masalah yang kompleks [1].

Sejarah algoritma Greedy dapat ditelusuri kembali ke awal perkembangan teori algoritma. Salah satu contoh awal penerapan Greedy adalah dalam algoritma Dijkstra untuk mencari jalur terpendek dalam graf. Algoritma ini diusulkan oleh Edsger W [2]. Dijkstra pada tahun 1956 dan menjadi salah satu fondasi dalam algoritma Greedy modern. Sejak saat itu, Greedy telah banyak diterapkan dalam berbagai konteks, termasuk optimasi kombinatorial, jaringan, dan kecerdasan buatan [3].

Prinsip dasar dari algoritma Greedy adalah membuat keputusan lokal yang terlihat terbaik pada setiap langkah dengan harapan bahwa solusi yang dihasilkan secara keseluruhan juga akan optimal. Langkah-langkah dalam algoritma Greedy tidak mempertimbangkan konsekuensi jangka panjang dari keputusan tersebut, melainkan hanya memperhatikan keadaan saat ini [4].

Misalnya, dalam masalah penukaran uang, algoritma Greedy akan memilih pecahan uang terbesar yang dapat ditukarkan untuk setiap permintaan penukaran. Dengan demikian, algoritma Greedy akan berulang kali memilih pecahan terbesar hingga jumlah yang tersisa menjadi nol atau mencapai nilai yang diinginkan. Namun, perlu dicatat bahwa algoritma Greedy mungkin tidak selalu menghasilkan solusi optimal, terutama jika ada kondisi khusus yang tidak memenuhi prinsip Greedy [5].

Algoritma pencocokan string adalah teknik yang digunakan untuk mencari kemunculan suatu pola (*pattern*) dalam teks (*text*). Tujuan dari algoritma ini adalah untuk menemukan posisi kemunculan semua kemungkinan pola dalam teks dengan efisien. Algoritma pencocokan string telah menjadi bagian integral dari banyak aplikasi dalam pemrosesan teks, seperti pencarian teks, pengindeksan teks, dan analisis bioinformatika [6].

Salah satu algoritma pencocokan string paling terkenal adalah algoritma Knuth-Morris-Pratt (KMP), yang dikembangkan oleh Donald Knuth, Vaughan Pratt, dan James H. Morris pada tahun 1970-an [7]. Algoritma ini menggunakan pendekatan yang cerdas untuk menghindari pencocokan ulang yang tidak perlu antara teks dan pola, sehingga meningkatkan efisiensi pencarian [8].

Algoritma pencocokan string berusaha untuk menemukan kemunculan pola dalam teks dengan menggunakan berbagai metode pencarian yang efisien. Salah satu pendekatan umum adalah dengan

menggunakan algoritma KMP. Algoritma KMP memanfaatkan informasi yang diperoleh dari pencarian sebelumnya untuk menghindari pencocokan ulang yang tidak perlu [9].

Pada dasarnya, algoritma KMP membagi teks menjadi bagian-bagian yang lebih kecil dan mencocokkan pola dengan bagian-bagian ini secara bergantian [10]. Jika ada ketidaksesuaian antara pola dan teks, algoritma KMP akan memanfaatkan informasi pra-pencocokan untuk memutuskan bagian mana dari teks yang perlu diabaikan dalam pencarian berikutnya [11].

Kedua algoritma Greedy dan pencocokan string memiliki relevansi yang besar dalam berbagai bidang, termasuk pemrosesan teks, jaringan komputer, kecerdasan buatan, dan optimasi kombinatorial. Penerapan algoritma Greedy dapat membantu dalam menyelesaikan masalah optimasi dengan cepat dan mudah dimengerti, meskipun tidak menjamin solusi yang optimal. Di sisi lain, algoritma pencocokan string memberikan kemampuan untuk mencari pola dalam teks dengan efisien, yang sangat penting dalam aplikasi seperti pengindeksan teks dan analisis bioinformatika [1], [3].

Dengan pemahaman yang mendalam tentang kedua algoritma ini, kita dapat mengembangkan solusi yang efisien dan optimal untuk berbagai masalah di berbagai bidang. Dalam penelitian ini, kami akan menjelajahi penerapan kedua algoritma ini dalam hal penukaran uang dan pencocokan lokasi cabang minimarket, dengan harapan memberikan kontribusi yang berarti terhadap efisiensi operasional dan kenyamanan pengguna dalam kehidupan sehari-hari.

LANDASAN TEORI

Systematic Literatur Review (SLR)

Systematic Literature Review (SLR) adalah sebuah metode penelitian yang sistematis untuk mengidentifikasi, mengevaluasi, dan menginterpretasikan semua penelitian yang relevan yang telah dilakukan dalam suatu bidang pengetahuan tertentu. Pendekatan ini memungkinkan peneliti untuk menyusun, menganalisis, dan menyimpulkan temuan dari berbagai sumber literatur yang ada secara objektif dan terstruktur. SLR memiliki beberapa langkah yang dapat diikuti, termasuk:

1. **Identifikasi Topik Penelitian**: Langkah pertama dalam SLR adalah mengidentifikasi topik atau pertanyaan penelitian yang jelas dan relevan. Hal ini memastikan fokus yang tepat dalam mencari literatur.
2. **Pencarian Literatur**: Peneliti melakukan pencarian literatur menggunakan basis data ilmiah yang terpercaya, seperti PubMed, IEEE Xplore, atau Google Scholar. Pencarian ini mencakup penggunaan kata kunci yang relevan dan pemilihan kriteria inklusi dan eksklusi yang sesuai.
3. **Seleksi Studi**: Setelah pencarian literatur dilakukan, peneliti meninjau abstrak dan teks penuh dari setiap artikel untuk menentukan apakah studi tersebut memenuhi kriteria inklusi. Artikel yang relevan kemudian dipilih untuk dianalisis lebih lanjut.
4. **Ekstraksi Data**: Data yang relevan dari setiap artikel yang dipilih diekstraksi secara sistematis. Ini bisa berupa temuan utama, metode penelitian, dan hasil dari studi yang dilaporkan.
5. **Evaluasi Kualitas**: Kualitas metodologi dari setiap studi yang dimasukkan dalam review dievaluasi. Ini memastikan bahwa temuan yang diambil dari literatur memiliki keandalan dan validitas yang tinggi.
6. **Analisis dan Interpretasi**: Data yang diekstraksi dari literatur dianalisis secara menyeluruh, baik secara kuantitatif maupun kualitatif. Hal ini membantu peneliti dalam memahami tren, pola, dan kesimpulan dari literatur yang telah ditinjau.

7. **Penyusunan Laporan**: Hasil dari SLR disusun dalam sebuah laporan sistematis yang mencakup gambaran menyeluruh tentang topik penelitian, temuan dari literatur, serta implikasi dan saran untuk penelitian selanjutnya.

SLR penting dalam hal penelitian ilmiah karena memberikan landasan yang kuat untuk memahami status pengetahuan yang ada dalam suatu bidang. Dengan memperoleh pemahaman yang komprehensif tentang literatur yang relevan, peneliti dapat mengidentifikasi celah pengetahuan, mengembangkan hipotesis baru, dan merancang studi penelitian yang lebih lanjut secara efektif.

Algoritma Greedy

Algoritma Greedy adalah paradigma algoritma yang berfokus pada pengambilan keputusan pada setiap langkah berdasarkan kriteria lokal terbaik, tanpa mempertimbangkan implikasi keputusan pada langkah-langkah selanjutnya. Dalam hal optimasi, algoritma Greedy bertujuan untuk menemukan solusi yang optimal secara global dengan melakukan serangkaian keputusan lokal yang optimal pada setiap langkahnya [12].

Konsep dasar dari algoritma Greedy adalah memilih pilihan terbaik yang tersedia pada setiap langkah, tanpa mempertimbangkan gambaran keseluruhan masalah. Kriteria pemilihan ini didasarkan pada heuristik atau aturan yang telah ditentukan sebelumnya. Meskipun algoritma Greedy tidak menjamin solusi yang optimal untuk setiap masalah, namun dalam banyak kasus, algoritma ini memberikan solusi yang cukup baik dalam waktu yang efisien. Langkah-langkah Algoritma Greedy [3], [5]:

1. **Inisialisasi**: Langkah pertama dalam algoritma Greedy adalah melakukan inisialisasi, yaitu menentukan nilai awal untuk setiap variabel yang diperlukan dalam proses algoritma.
2. **Seleksi Langkah**: Pada setiap langkah, algoritma Greedy memilih pilihan terbaik yang tersedia berdasarkan aturan atau heuristik yang telah ditentukan sebelumnya. Keputusan ini didasarkan pada evaluasi lokal dari setiap pilihan yang tersedia.
3. **Evaluasi**: Setelah langkah dipilih, algoritma mengevaluasi dampak dari keputusan tersebut pada solusi keseluruhan. Namun, evaluasi ini hanya mempertimbangkan kontribusi lokal dari langkah tersebut, tanpa memperhitungkan dampaknya pada langkah-langkah selanjutnya.
4. **Iterasi**: Proses seleksi langkah dan evaluasi dilakukan secara iteratif sampai kriteria berhenti terpenuhi. Kriteria berhenti ini dapat berupa mencapai solusi yang memenuhi syarat tertentu atau sampai tidak ada langkah lagi yang dapat diambil.
5. **Penyelesaian**: Algoritma Greedy menghasilkan solusi final ketika iterasi selesai. Solusi ini mungkin tidak selalu optimal, tetapi sering kali cukup baik untuk digunakan dalam banyak kasus.

Salah satu contoh penerapan algoritma Greedy adalah pada masalah penjadwalan kegiatan (scheduling problem). Misalkan terdapat sejumlah kegiatan dengan waktu mulai dan waktu selesai yang berbeda-beda, dan tujuan adalah menemukan urutan penjadwalan kegiatan yang memungkinkan untuk menyelesaikan sebanyak mungkin kegiatan tanpa tumpang tindih. Dalam hal ini, algoritma Greedy dapat digunakan dengan memilih kegiatan yang memiliki waktu selesai paling awal pada setiap langkahnya [3].

Kelebihan dari penggunaan algoritma Greedy adalah sebagai berikut:

a. Algoritma Greedy cenderung lebih sederhana dan mudah dimengerti dibandingkan dengan beberapa teknik optimasi yang lebih kompleks.
b. Algoritma Greedy sering kali memiliki kompleksitas waktu yang rendah, sehingga dapat diimplementasikan secara efisien untuk masalah-masalah skala besar.
c. Solusi Cepat: Algoritma Greedy biasanya menghasilkan solusi dengan cepat karena mengambil keputusan berdasarkan aturan lokal.

Keterbatasan algoritma Greedy adalah sebagai berikut:

a. Algoritma Greedy tidak menjamin solusi optimal untuk setiap masalah, sehingga dapat menghasilkan solusi yang suboptimal dalam beberapa kasus.
b. Kinerja algoritma Greedy sangat dipengaruhi oleh pemilihan aturan atau heuristik yang digunakan untuk memilih langkah terbaik pada setiap langkahnya.
c. Algoritma Greedy cenderung kurang cocok untuk masalah-masalah yang memiliki struktur yang kompleks atau terkait dengan ketergantungan antar langkah.

String Matching

Pencocokan string adalah proses mencari kemunculan suatu pola (*pattern*) dalam sebuah teks. Teknik ini merupakan bagian penting dalam berbagai aplikasi komputasi, termasuk pengolahan bahasa alami, analisis teks, dan pengenalan pola dalam data biologis [13]. Dalam hal algoritma, terdapat berbagai pendekatan untuk mencocokkan pola dalam teks, mulai dari metode pencocokan string naif hingga algoritma yang lebih efisien seperti Knuth-Morris-Pratt (KMP) dan algoritma Boyer-Moore.

Algoritma Knuth Morris Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah sebuah algoritma pencocokan string yang efisien untuk mencari kemunculan sebuah pola (pattern) dalam sebuah teks. Dibandingkan dengan metode pencocokan string naif, KMP memanfaatkan informasi praproses dari pola untuk meminimalkan jumlah karakter yang harus dibandingkan [2].

Langkah-langkah Algoritma Knuth-Morris-Pratt (KMP) untuk pencocokan pola dalam sebuah teks adalah sebagai berikut [6]:

1. **Pra-pemrosesan Pola**: Langkah pertama algoritma KMP adalah membuat tabel lompatan (failure function) yang menyimpan informasi tentang kemungkinan ketidakcocokan antara pola dan teks. Tabel ini digunakan untuk menentukan langkah berikutnya ketika terjadi ketidakcocokan selama pencocokan.
2. **Pencocokan Teks dan Pola**: Algoritma KMP memulai pencocokan pada posisi awal teks dan pola. Ketika terjadi ketidakcocokan pada posisi tertentu, algoritma menggunakan tabel lompatan untuk menentukan jumlah karakter yang harus dilewati sebelum mencoba mencocokkan pola lagi.
3. **Pencarian Kemunculan Pola**: Algoritma KMP melanjutkan proses pencocokan sampai seluruh teks telah dijelajahi atau kemunculan pola telah ditemukan. Setiap kali pola cocok dengan segmen teks, posisi tersebut dicatat sebagai kemunculan pola.

Salah satu keunggulan utama dari Algoritma Knuth-Morris-Pratt (KMP) adalah efisiensi waktu yang tinggi dalam pencarian pola dalam sebuah teks. Dibandingkan dengan metode brute-force yang melakukan pencocokan karakter per karakter, KMP menggunakan tabel lompatan yang telah diproses sebelumnya untuk menghindari pembacaan ulang karakter yang sudah cocok. Hal ini menghasilkan kinerja yang lebih cepat terutama ketika pola memiliki kemiripan dengan teks yang luas. Selain itu, keunggulan lainnya adalah kemampuan algoritma KMP untuk menemukan semua kemunculan pola dalam teks dengan waktu yang konstan per karakter dalam teks, tanpa terpengaruh oleh panjang pola

yang dicari. Dengan demikian, algoritma KMP menjadi pilihan yang efisien dan andal dalam aplikasi yang memerlukan pencarian pola dalam teks dengan cepat dan efisien.

Misalkan kita memiliki sebuah teks "ABABABCABABABCABABABC" dan kita ingin mencari kemunculan pola "ABABAC" di dalamnya menggunakan algoritma KMP. Langkah-langkahnya adalah sebagai berikut [14]:

1. Pra-pemrosesan pola "ABABAC" untuk membuat tabel lompatan.
2. Pencocokan dimulai pada posisi awal teks dan pola.
3. Pada posisi keempat, terjadi ketidakcocokan pada karakter "B" dalam teks dan "C" dalam pola. Algoritma menggunakan tabel lompatan untuk menentukan langkah selanjutnya.
4. Algoritma melompat ke posisi berikutnya dalam teks di mana karakter "A" cocok dengan karakter pertama dalam pola.
5. Proses pencocokan berlanjut sampai seluruh teks telah dijelajahi atau kemunculan pola telah ditemukan.

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma yang efisien dan andal untuk pencocokan string. Dengan memanfaatkan informasi praproses dari pola, KMP mampu mencari kemunculan pola dalam teks dengan kompleksitas waktu yang rendah [6], [8]. Hal ini membuatnya menjadi pilihan yang populer dalam berbagai aplikasi yang membutuhkan pencocokan string yang cepat dan efisien.

METODE PENELITIAN

Dalam penelitian ini, penulis akan mengevaluasi metode penelitian berdasarkan tiga kriteria utama: kelengkapan (*completeness*), optimalitas (*optimality*), dan kompleksitas waktu (*time complexity*).

Completeness

Kelengkapan mengacu pada sejauh mana metode penelitian mencakup semua aspek yang diperlukan untuk mencapai tujuan penelitian. Dalam hal ini, kelengkapan akan dievaluasi berdasarkan sejauh mana metode penelitian mencakup langkah-langkah yang diperlukan untuk mengimplementasikan algoritma Greedy dalam penukaran uang di Alfamart dan algoritma pencocokan string untuk pencarian cabang Alfamart di berbagai kota di Indonesia. Hal ini termasuk analisis, desain, pengembangan perangkat lunak, pengujian, evaluasi, perbaikan, dan penyempurnaan [15].

Optimality

Optimalitas mengacu pada sejauh mana metode penelitian dapat mencapai hasil yang optimal dalam hal penelitian yang diberikan. Dalam hal ini, optimalitas akan dievaluasi berdasarkan efektivitas dan efisiensi algoritma yang diimplementasikan. Algoritma Greedy dalam penukaran uang di Alfamart dan algoritma pencocokan string untuk pencarian cabang Alfamart diharapkan dapat memberikan solusi yang optimal dalam hal efisiensi operasional, akurasi, dan kinerja keseluruhan [16].

Time Complexity

Kompleksitas waktu mengacu pada seberapa efisien metode penelitian dalam menghabiskan waktu untuk menyelesaikan tugas yang diberikan. Dalam hal ini, kompleksitas waktu akan dievaluasi berdasarkan kompleksitas algoritma yang diimplementasikan. Algoritma Greedy dan algoritma

pencocokan string yang digunakan dalam penelitian ini diharapkan memiliki kompleksitas waktu yang rendah agar dapat memberikan kinerja yang cepat dan responsif dalam penukaran uang di Alfamart dan pencarian cabang Alfamart [17].

HASIL DAN PEMBAHASAN

Menggunakan Algoritma Greedy

Algoritma Greedy merupakan pendekatan yang digunakan dalam proses penukaran uang di Alfamart. Contoh implementasi algoritma Greedy dalam bahasa pemrograman C++ dapat dilihat di Gambar 1.

Gambar 1. Implementasi Algoritma Greedy

Langkah-langkah yang diambil dalam algoritma tersebut adalah sebagai berikut:

1. Mengurutkan stok uang dari yang terbesar ke yang terkecil untuk memaksimalkan jumlah uang yang ditukarkan.
2. Iterasi melalui stok uang dan mencoba
3. untuk menukarkan sebanyak mungkin uang dengan nilai pecahan yang lebih besar terlebih dahulu.
4. Memperbarui sisa uang yang belum ditukarkan setelah setiap iterasi.
5. Menampilkan hasil penukaran uang, termasuk jumlah lembar/koin pecahan uang yang berhasil ditukarkan.
6. Jika masih ada sisa uang yang tidak bisa ditukarkan, menampilkan pesan yang sesuai.

Implementasi algoritma Greedy dalam penukaran uang di Alfamart menawarkan pendekatan yang efisien dan cepat dalam mengelola proses penukaran uang bagi pelanggan. Dalam penjelasan interpretasi ini, kita akan mengeksplorasi beberapa aspek penting dari implementasi algoritma Greedy tersebut.

Pertama-tama, algoritma Greedy memungkinkan Alfamart untuk memprioritaskan penukaran uang dengan denominasi tertinggi terlebih dahulu. Dengan cara ini, jumlah uang yang dapat ditukarkan secara efektif ditingkatkan, memaksimalkan penggunaan stok uang yang tersedia. Hal ini memberikan keuntungan bagi pelanggan, karena mereka dapat menerima jumlah uang kembalian yang optimal dalam transaksi mereka. Selanjutnya, pengurutan stok uang dari denominasi tertinggi ke terendah memungkinkan algoritma untuk menyesuaikan penukaran uang dengan lebih baik terhadap jumlah yang diminta oleh pelanggan. Dengan cara ini, proses penukaran uang menjadi lebih efisien dan responsif terhadap kebutuhan pelanggan, mengurangi kemungkinan kesalahan atau kekurangan dalam pelayanan.

Selain itu, algoritma Greedy juga memungkinkan Alfamart untuk mengoptimalkan penggunaan stok uang tunai mereka. Dengan memaksimalkan jumlah uang yang ditukarkan dalam setiap transaksi, Alfamart dapat mengurangi risiko kekurangan uang tunai di kas mereka, yang dapat mengganggu kelancaran operasional dan pelayanan kepada pelanggan.

Namun, meskipun algoritma Greedy menawarkan pendekatan yang efisien dalam penukaran uang, penting untuk diingat bahwa itu bukanlah solusi yang sempurna dalam semua situasi. Ada

kemungkinan bahwa ada kekurangan dalam stok uang yang tersedia, atau bahwa pola permintaan pelanggan tidak sesuai dengan strategi algoritma Greedy yang diterapkan.

Menggunakan Algoritma String Matching

Algoritma String Matching digunakan dalam pencarian cabang Alfamart di berbagai kota di Indonesia. Pencocokan string adalah teknik yang penting dalam komputasi untuk menemukan kemunculan sebuah pola (pattern) dalam sebuah teks.

Dalam implementasi algoritma String Matching, digunakan algoritma Knuth-Morris-Pratt (KMP), yang telah terbukti efisien dalam mencocokkan pola dalam teks dengan kompleksitas waktu yang rendah. Berikut adalah implementasi pseudocode algoritma KMP string matching dengan bahasa C++:

```cpp
// Fungsi untuk mencari cabang Alfamart berdasarkan nama kota menggunakan algoritma String Matching
vector<string> findAlfamartBranches(const vector<string> &branches, const string &cityName)
{
    vector<string> matchingBranches;

    // Melakukan pencarian menggunakan algoritma String Matching
    for (const auto &branch : branches)
    {
        // Mengubah kedua string ke huruf kecil agar case-insensitive
        string lowercaseBranch = branch;
        string lowercaseCityName = cityName;
        transform(lowercaseBranch.begin(), lowercaseBranch.end(), lowercaseBranch.begin(), ::tolower);
        transform(lowercaseCityName.begin(), lowercaseCityName.end(), lowercaseCityName.begin(), ::tolower);

        // Jika nama kota ditemukan dalam nama cabang, tambahkan ke hasil pencarian
        if (lowercaseBranch.find(lowercaseCityName) != string::npos)
        {
            matchingBranches.push_back(branch);
        }
    }

    return matchingBranches;
}
```

Gambar 2. Implementasi Algoritma String Matching

Dengan menggunakan algoritma String Matching, Alfamart dapat secara efisien mencari dan menemukan kemunculan nama cabangnya dalam data cabang Alfamart di berbagai kota di Indonesia. Hal ini memungkinkan pelanggan untuk dengan mudah menemukan lokasi cabang Alfamart yang terdekat dengan mereka, meningkatkan kenyamanan dan aksesibilitas dalam berbelanja di Alfamart.

Namun, penting untuk dicatat bahwa meskipun algoritma KMP memiliki kompleksitas waktu yang rendah, perlu dilakukan evaluasi lebih lanjut terhadap kinerja dan efektivitas algoritma ini dalam hal spesifik pencarian cabang Alfamart. Evaluasi tersebut dapat membantu dalam mengidentifikasi potensi perbaikan dan penyesuaian yang diperlukan untuk meningkatkan kinerja pencarian cabang Alfamart menggunakan algoritma String Matching.

Analisis Time Complexity

Analisis Time Complexity pada kedua algoritma ini membantu kita memahami seberapa efisien algoritma tersebut dalam menghadapi masukan yang berbeda-beda. Ini memberikan perkiraan tentang seberapa cepat algoritma akan berjalan saat ukuran masukan meningkat.

Untuk algoritma Greedy dalam penukaran uang di Alfamart, kompleksitas waktu terutama ditentukan oleh langkah-langkahnya, termasuk pengurutan stok uang dan iterasi melalui stok uang untuk melakukan penukaran. Kompleksitas waktu umumnya tergantung pada jumlah denominasi uang yang tersedia.

Sementara itu, algoritma String Matching digunakan untuk mencari nama cabang Alfamart berdasarkan nama kota. Analisis kompleksitas waktu pada algoritma ini bergantung pada panjang teks (jumlah nama cabang) dan panjang pola (nama kota yang dicari).

Dalam kedua kasus, penulis ingin memahami seberapa cepat algoritma akan berjalan saat ukuran input meningkat. Ini membantu kita memilih algoritma yang paling cocok untuk situasi tertentu.

Tabel 1. Percobaan Algoritma Greedy

Percobaan Algoritma Greedy	
Percobaan	**Waktu (s)**
Percobaan ke-1	0.00194
Percobaan ke-2	0.00258
Percobaan ke-3	0.00195
Percobaan ke-4	0.00353
Percobaan ke-5	0.00194

Dari Tabel 1, dapat diamati bahwa Algoritma Greedy menunjukkan konsistensi yang baik dalam kinerjanya. Waktu eksekusi yang tercatat untuk setiap percobaan relatif singkat, dengan variasi yang terbatas antara percobaan satu dengan percobaan lainnya. Dalam hal ini, waktu eksekusi berkisar dari sekitar 0.00194 detik hingga 0.00353 detik.

Kesimpulan dari hasil ini adalah bahwa Algoritma Greedy efisien dalam menyelesaikan tugas yang diberikan. Meskipun waktu eksekusi bisa sedikit berbeda antar percobaan, variasi ini tetap dalam kisaran yang wajar dan tidak menunjukkan adanya pola yang signifikan. Hal ini menunjukkan bahwa Algoritma Greedy dapat diandalkan dalam berbagai situasi, dengan kemampuan untuk memberikan solusi dengan waktu eksekusi yang cepat dan stabil. Dengan demikian, hasil ini menegaskan bahwa Algoritma Greedy merupakan pilihan yang baik dalam kasus-kasus di mana efisiensi waktu adalah pertimbangan utama.

Tabel 2. Percobaan Algoritma String Matching

Percobaan Algoritma String Matching	
Percobaan	**Waktu**
Percobaan ke-1	0.00410
Percobaan ke-2	0.00413
Percobaan ke-3	0.00312
Percobaan ke-4	0.00199
Percobaan ke-5	0.00950

Dari Tabel 2, dapat diamati bahwa waktu eksekusi Algoritma String Matching bervariasi antara setiap percobaan. Rentang waktu eksekusi berkisar dari 0.00199 hingga 0.00950, menunjukkan variasi yang cukup signifikan dalam kinerja algoritma.

Percobaan pertama dan kedua menunjukkan waktu eksekusi yang cukup stabil, dengan nilai sekitar 0.00410 dan 0.00413 detik. Namun, percobaan ketiga menunjukkan waktu eksekusi yang lebih rendah, yaitu sekitar 0.00312 detik. Percobaan keempat menunjukkan waktu eksekusi terendah, hanya sekitar 0.00199 detik, sementara percobaan kelima menunjukkan waktu eksekusi tertinggi, sekitar 0.00950 detik.

Interpretasi dari hasil ini adalah bahwa Algoritma String Matching menunjukkan variasi yang signifikan dalam kinerjanya antar percobaan. Walaupun beberapa percobaan menunjukkan waktu eksekusi yang relatif stabil, ada juga percobaan yang menunjukkan waktu eksekusi yang jauh lebih tinggi atau lebih rendah dari yang lain. Hal ini mungkin disebabkan oleh faktor-faktor seperti ukuran input yang berbeda atau karakteristik khusus dari data yang sedang diproses.

Dengan demikian, hasil dari Tabel 2 menunjukkan bahwa kinerja Algoritma String Matching cenderung bervariasi, dan mungkin tidak konsisten dari satu percobaan ke percobaan lainnya. Variasi ini penting untuk dipertimbangkan dalam penggunaan algoritma ini, terutama jika waktu eksekusi yang konsisten sangat diinginkan. Evaluasi lebih lanjut dapat diperlukan untuk memahami penyebab variasi dalam kinerja algoritma ini dan mengidentifikasi cara untuk meningkatkan stabilitasnya.

KESIMPULAN DAN SARAN

Kesimpulan

Dari hasil penelitian ini, dapat disimpulkan bahwa penggunaan algoritma Greedy dalam proses penukaran uang di Alfamart menawarkan pendekatan yang efisien dan cepat dalam mengelola proses penukaran uang bagi pelanggan. Implementasi algoritma Greedy memungkinkan Alfamart untuk memaksimalkan penggunaan stok uang yang tersedia dengan memprioritaskan penukaran uang dengan denominasi tertinggi terlebih dahulu. Hal ini menghasilkan jumlah uang kembalian yang optimal bagi pelanggan dan mengoptimalkan penggunaan stok uang tunai Alfamart. Meskipun demikian, perlu diingat bahwa algoritma Greedy bukanlah solusi yang sempurna dalam semua situasi, dan kemungkinan terdapat kekurangan dalam stok uang yang tersedia atau pola permintaan pelanggan yang tidak sesuai dengan strategi algoritma Greedy yang diterapkan.

Sementara itu, penggunaan algoritma String Matching, khususnya algoritma Knuth-Morris-Pratt (KMP), dalam pencarian cabang Alfamart di berbagai kota di Indonesia, memberikan cara yang efisien untuk menemukan lokasi cabang Alfamart yang terdekat dengan pelanggan. Algoritma KMP menawarkan kompleksitas waktu yang rendah dalam mencocokkan pola dalam teks, memungkinkan Alfamart untuk dengan mudah mencari dan menemukan kemunculan nama cabangnya dalam data cabang Alfamart di berbagai kota di Indonesia. Meskipun demikian, variasi yang signifikan dalam kinerja algoritma String Matching perlu dipertimbangkan, dan evaluasi lebih lanjut mungkin diperlukan untuk memahami penyebab variasi tersebut dan mengidentifikasi cara untuk meningkatkan stabilitasnya.

Saran

Berdasarkan hasil penelitian ini, beberapa saran dapat diajukan untuk pengembangan lebih lanjut:

1. Evaluasi lebih lanjut terhadap kinerja algoritma String Matching, termasuk algoritma KMP, dapat dilakukan untuk memahami penyebab variasi dalam kinerja algoritma tersebut. Hal ini dapat membantu dalam mengidentifikasi cara untuk meningkatkan stabilitasnya dan menjaga konsistensi kinerja dalam berbagai situasi.
2. Penelitian lebih lanjut dapat dilakukan untuk mengidentifikasi dan memahami pola permintaan pelanggan dalam proses penukaran uang di Alfamart. Informasi ini dapat digunakan untuk menyesuaikan strategi algoritma Greedy yang diterapkan, sehingga dapat lebih efektif dalam mengelola proses penukaran uang.
3. Pengembangan perangkat lunak yang mengintegrasikan algoritma Greedy dan algoritma String Matching dapat dilakukan untuk menciptakan solusi yang komprehensif dalam mengelola proses penukaran uang dan pencarian lokasi cabang Alfamart. Hal ini dapat meningkatkan efisiensi operasional Alfamart dan memberikan pengalaman yang lebih baik bagi pelanggan.

Dengan menerapkan saran-saran tersebut, diharapkan dapat meningkatkan efisiensi dan efektivitas dalam penggunaan algoritma Greedy dan algoritma String Matching dalam hal penukaran uang di Alfamart dan pencarian lokasi cabang Alfamart, serta memberikan kontribusi yang lebih besar terhadap kepuasan pelanggan dan kinerja operasional Alfamart secara keseluruhan.

DAFTAR PUSTAKA

[1] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, "A greedy algorithm for aligning DNA sequences," *Journal of Computational biology*, vol. 7, no. 1–2, pp. 203–214, 2000.

[2] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *Eur J Oper Res*, vol. 177, no. 3, pp. 2033–2049, 2007.

[3] A. Vince, "A framework for the greedy algorithm," *Discrete Appl Math (1979)*, vol. 121, no. 1–3, pp. 247–260, 2002.

[4] R. D. Septiana, D. A. Punkastyo, and N. Nugroho, "Implementasi Algoritma Greedy dan Algoritma A* Untuk Penentuan Cost Pada Routing Jaringan," *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 3, no. 2, pp. 181–187, 2022.

[5] E. Darnila, M. Ula, and C. D. A. Soraya, "Optimasi Kelayakan Kondisi Pembangunan Jalan di Kota Lhokseumawe Menggunakan Algoritma Greedy," *JUKI: Jurnal Komputer dan Informatika*, vol. 1, no. 1, pp. 9–14, 2019.

[6] D. Anggreani, D. P. I. Putri, A. N. Handayani, and H. Azis, "Knuth Morris Pratt algorithm in enrekang-indonesian language translator," in *2020 4th International Conference on Vocational Education and Training (ICOVET)*, IEEE, 2020, pp. 144–148.

[7] B. Commentz-Walter, "A string matching algorithm fast on the average," in *International Colloquium on Automata, Languages, and Programming*, Springer, 1979, pp. 118–132.

[8] A. Nizar, P. Harsani, and I. Anggraeni, "Robot Virtual Menggunakan Metode Knuth Morris Pratt: Virtual Robot Using Knuth Morris Pratt Method," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 1, pp. 282–292, 2024.

[9] M. Equi, V. Mäkinen, A. I. Tomescu, and R. Grossi, "On the complexity of string matching for graphs," *ACM Transactions on Algorithms*, vol. 19, no. 3, pp. 1–25, 2023.

[10] P. Charalampopoulos, T. Kociumaka, and P. Wellnitz, "Faster approximate pattern matching: A unified approach," in *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2020, pp. 978–989.

[11] I. Ahmad, R. I. Borman, G. G. Caksana, and J. Fakhrurozi, "Implementasi String Matching Dengan Algoritma Boyer-Moore Untuk Menentukan Tingkat Kemiripan Pada Pengajuan Judul Skripsi/Ta Mahasiswa (Studi Kasus: Universitas XYZ)," *SINTECH (Science and Information Technology) Journal*, vol. 4, no. 1, pp. 53–58, 2021.

[12] A. Roihan, K. Nasution, and M. Z. Siambaton, "Implementasi Algoritma Greedy Kombinasi dengan Perulangan pada Aplikasi Penjadwalan Praktikum," *sudo Jurnal Teknik Informatika*, vol. 1, no. 2, pp. 42–50, 2022.

[13] Y. Faqih, Y. Rahmanto, A. A. Aldino, and B. Waluyo, "Penerapan String Matching Menggunakan Algoritma Boyer-Moore Pada Pengembangan Sistem Pencarian Buku Online," *Bulletin of Computer Science Research*, vol. 2, no. 3, pp. 100–106, 2022.

[14] M. Ilham and A. H. Mirza, "Penerapan Algoritma Knuth Morris Pratt Dalam Fitur Pencarian Pengarsipan Dokumen Pada Sma Plus Negeri 17 Palembang," *Journal of Software Engineering Ampera*, vol. 1, no. 2, pp. 110–121, 2020.

[15] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *Int J Rob Res*, vol. 37, no. 1, pp. 46–61, 2018.