Perbandingan Algoritma Forward Chaining dalam Sistem Pakar Rekomendasi Peminatan Bidang Teknologi

Putu Agus Dharma Kusuma[a1], I Made Widiartha[a2]

[a]Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Udayana
Jalan Raya Kampus Udayana, Bukit Jimbaran, Kuta Selatan, Badung, Bali Indonesia
[1]agusdharma48@gmail.com
[2]madewidiartha@unud.ac.id

Abstract

This research aims to compare the forward chaining algorithm with the Backward Chaining, Breadth-First Search (BFS), and Depth-First Search (DFS) algorithms in the context of an expert system for recommending specialization in the field of technology. The primary focus of this study is to analyze the runtime performance of each algorithm and determine the algorithm that provides the fastest runtime. The research methodology involves implementing the four algorithms in an expert system that provides recommendations for technology field specialization based on rules and user responses. The data used in the study consists of specialization rules in the technology field and user responses related to their interests in those fields. The results of the study demonstrate that the forward chaining algorithm outperforms the Backward Chaining, BFS, and DFS algorithms in terms of runtime performance. This indicates that the forward chaining algorithm is more efficient in generating technology field specialization recommendations. Based on the findings of this research, it is recommended to use the forward chaining algorithm in the development of expert systems for technology field specialization. This algorithm can assist users in obtaining recommendations quickly and efficiently, thereby enhancing user experience and the effectiveness of the expert system in providing suitable technology field specializations based on user interests.

Keywords: *Forward Chaining, Expert System, Expertise, Comparative Study, Runtime Performance*

1. Pendahuluan

Sistem pakar adalah sebuah program komputer yang pintar dan mengandalkan pengetahuan serta metode penalaran untuk menyelesaikan masalah yang kompleks dan membutuhkan keahlian khusus dari seorang ahli. Tujuan utama sistem pakar adalah meniru kemampuan pengambilan keputusan seorang ahli dalam suatu bidang tertentu. Dalam melakukan tugasnya, sistem pakar menggunakan pengetahuan khusus yang dimiliki oleh seorang ahli untuk memberikan solusi dan rekomendasi dalam pemecahan masalah [1]. Dalam penerapannya terdapat banyak sekali algoritma dalam pembangunan sistem pakar seperti *forward chaining*. Metode *forward chaining* memiliki banyak kekurangan bahkan bila dibandingkan algoritma serupa seperti backward chaining. Dari segi runtime, metode *forward chaining* memiliki kekurangan yang cukup signifikan yaitu sebesar 72,5% pengguna menunggu lebih lama dibandingkan dengan menggunakan algoritma backward chaining yaitu sebesar 87,5% [2]. Hal tersebut memberikan pengaruh yang cukup signifikan sehingga diperlukannya modifikasi pada algoritma *forward chaining* khususnya dalam runtime dan akurasi. Pada penelitian ini akan berfokus pada algoritma *forward chaining* dari segi implementasi sistem informasi.

2. Metode Penelitian

2.1. Analisis Environment

Pada fase ini akan dilakukan proses analisis environment khususnya dalam melakukan perbandingan algoritma forward chaining dengan algoritma lainnya. Penyamaan environment akan memperjelas perbedaan runtime antar algoritma sehingga perbedaan dapat diamati.

2.2. Analisis Algoritma

Tahapan ini merupakan tahapan untuk melakukan perhitungan algoritma dan melakukan perbandingan dengan algoritma lainnya. Analisis ini akan membandingkan beberapa kali runtime yang dilakukan dan menganalisis berdasarkan hasil perbandingan tersebut.

2.3. Metode Pengembangan SIstem

Pada fase ini akan dilakukan proses pengembangan sistem berdasarkan environment yang sudah ditetapkan. Dalam tahap ini akan dilakukan proses *Software Development Life Cycle* atau SDLC dalam pengembangan sistem.

2.4. Dokumentasi

Pada tahap ini akan dilakukan dokumentasi hasil jadi sistem dengan algoritma yang cukup efisien diterapkan pada sistem tersebut.

3. Hasil dan Pembahasan

3.1. Analisis Environment

Pada penelitian ini menggunakan beberapa tools dan framework yang akan membantu proses jalannya penelitian. Penelitian ini akan berbasis web menggunakan bahasa utama typescript dan framework next js sebagai sisi frontend. Dan pada sisi algoritma akan menggunakan python khususnya dalam perhitungan runtime algoritma yang telah dibuat. Penggunaan typescript pada proses pengembangan aplikasi ini dikarenakan typescript memiliki kemampuan dalam meminimalisir error dalam proses pengembangan. Hal ini dikarenakan pada bahasa typescript, wajib melakukan definisi tipe variabel sehingga setiap value akan secara tepat mengisi variabel yang ditentukan. Hal ini cukup berbeda dengan bahasa javascript yang secara bebas dapat memasukkan variabel tertentu sehingga akan rentan dalam memicu error pada proses developing bahkan production [3]. Pada proses komunikasinya akan menggunakan sistem rest API dengan menggunakan JSON sebagai perantara komunikasi antara algoritma dengan aplikasi frontendnya. Dari sisi client akan bertugas dalam mengambil input dan hasil inputan tersebut akan di Post ke dalam server untuk dilakukan perhitungan minat yang diinginkan [5]. Pada saat yang bersamaan server akan menghitung runtime algoritma sehingga penulis akan melakukan perbandingan runtime algoritma dengan algoritma lainnya.

3.2. Analisis Algoritma

Algoritma *forward chaining* yang diimplementasikan akan dilakukan proses sedikit modifikasi yaitu pada variabel “target” di rules. Pada variabel tersebut akan menggunakan tipe data list sehingga dalam 1 question kepakaran akan dapat menjurus ke 2 target minat yang berbeda atau lebih.

```
function forward_chaining(rules, answers):
    interests = {}
    for each rule in rules:
        for each target in rule['target']:
            interests[target] = 0

    for each answer in answers:
        matching_rules = find_matching_rules(rules, answer['id'])
        for each rule in matching_rules:
            for each target in rule['target']:
                interests[target] += answer['answer']

    total_scores = calculate_total_scores(interests)
    mapping = {}
    for each interest, score in interests.items():
        percentage = (score / total_scores) * 100
        mapping[interest] = round(percentage)

    return mapping

function find_matching_rules(rules, answer_id):
    matching_rules = []
    for each rule in rules:
        if rule['id'] == answer_id:
            matching_rules.append(rule)
    return matching_rules

function calculate_total_scores(interests):
    total_scores = 0
    for each score in interests.values():
        total_scores += score
    return total_scores

recommendations = forward_chaining(rules, answers)
print(recommendations)
```

Gambar 1. *Forward chaining*

Pseudocode tersebut merupakan hasil modifikasi dari algoritma f*orward chaining* yang akan diimplementasikan, pada algoritma tersebut memiliki input rules dan answers yang merupakan list map/dictionary dari kumpulan aturan yang telah ditetapkan dan kumpulan jawaban dari user.

```
Rule = Dict[str, List[str]]
Answer = Dict[str, int]
InterestMapping = Dict[str, int]


def forward_chaining(rules: List[Rule], answers: List[Answer]) -> InterestMapping:
    interests = {}
    for rule in rules:
        for target in rule['target']:
            interests[target] = 0

    for answer in answers:
        matching_rules = [rule for rule in rules if rule['id'] == answer['id']]
        for rule in matching_rules:
            for target in rule['target']:
                interests[target] += answer['answer']

    total_scores = sum(interests.values())
    mapping = {}
    for interest, score in interests.items():
        percentage = (score / total_scores) * 100
        mapping[interest] = round(percentage)

    return mapping


# Mulai menghitung runtime
start_time = time.time()

recommendations = forward_chaining(rules, answers)
print(recommendations)
```

Gambar 2. *Forward chaining implementation*

Source Code berikut merupakan hasil implementasi algoritma *forward chaining* pada sistem rekomendasi minat bidang informatika. Fungsi utama pada implementasi sistem pakar rekomendasi minat bidang informatika fungsi *forward_chaining()* dalam bahasa pemrograman Python. Fungsi ini menerima dua parameter, yaitu rules dan answers, dan mengembalikan sebuah *InterestMapping* yang berisi hasil rekomendasi peminatan berdasarkan aturan dan jawaban dari pengguna. Pertama, fungsi ini membuat sebuah dictionary interests yang akan menyimpan jumlah nilai awal untuk setiap peminatan. Setiap target peminatan dari setiap aturan diiterasi, dan nilai awalnya diinisialisasi dengan 0. Kemudian, fungsi melakukan iterasi pada setiap jawaban yang diberikan. Mencari aturan-aturan yang cocok dengan id jawaban, dan mengupdate nilai peminatan yang sesuai dalam dictionary interests dengan menambahkan nilai jawaban. Setelah itu, total skor peminatan dihitung dengan menjumlahkan semua nilai dalam interests. Selanjutnya, fungsi membuat dictionary mapping yang akan menyimpan persentase peminatan berdasarkan skor yang dihitung sebelumnya. Setiap peminatan diiterasi, dan persentase peminatan dihitung dengan membagi skor peminatan dengan total skor, kemudian dikalikan dengan 100. Nilai persentase tersebut dibulatkan ke angka terdekat dan disimpan dalam mapping. Akhirnya, fungsi mengembalikan mapping yang berisi rekomendasi peminatan berdasarkan hasil perhitungan. Selain fungsi *forward_chaining()*, dalam source code juga terdapat definisi variabel rules. Variabel ini berisi daftar aturan-aturan peminatan bidang teknologi yang digunakan dalam sistem pakar. Setiap aturan memiliki id, deskripsi, dan target peminatan yang terkait. Dalam penggunaan source code tersebut, perlu diperhatikan bahwa variabel rules dan answers harus didefinisikan dengan nilai yang sesuai sebelum pemanggilan fungsi *forward_chaining()*. Proses perbandingan akan dilakukan dari sisi server menggunakan bahasa Python sehingga akan memperlihatkan efektivitas dari aplikasi yang sedang dibangun. Selain itu rules yang akan digunakan akan sama dengan pembanding lainnya sehingga akan terlihat perbedaan runtime dari masing-masing algoritma. Adapun algoritma pembanding yang akan dipakai yaitu backward chaining, BFS, dan DFS.

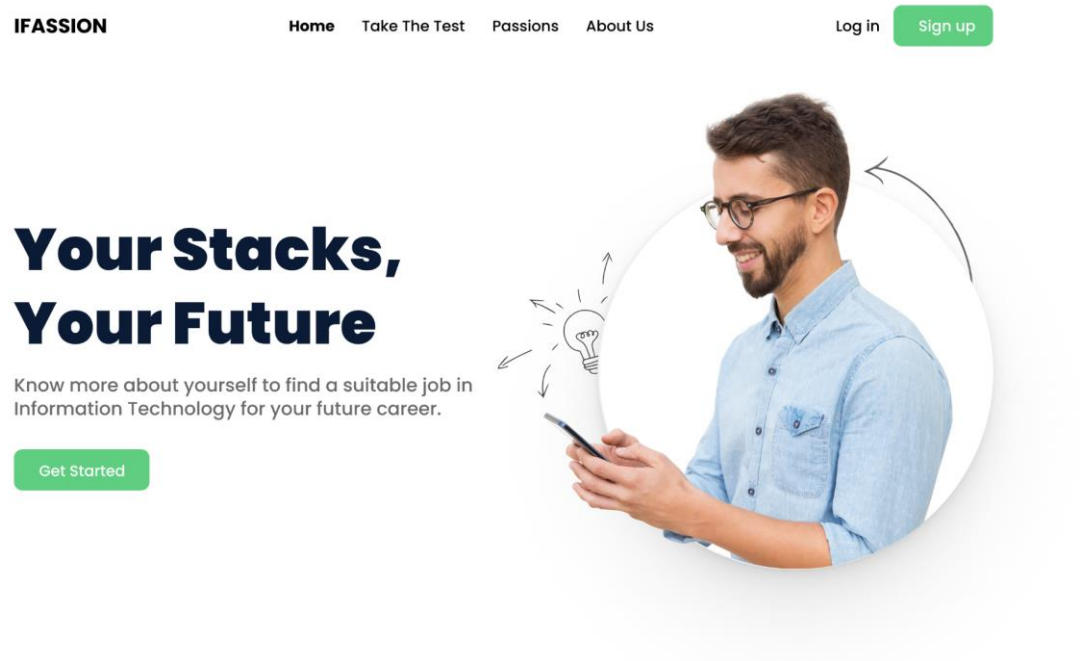
Tabel 1. Karakteristik Basis Data

	Forward Chaining (s)	**Backward Chaining (s)**	**BFS (s)**	**DFS (s)**
Runtime 1	6.67572021484375e-05	0.00010180473327636719	0.00010013580322265625	0.0006940364837646484
Runtime 2	6.484985351562 5e-05	0.0002269744873046875	9.89437103271484 4e-05	0.0007266998291015625
Runtime 3	6.198883056640 625e-05	9.17911529541015 6e-05	8.89301300048828 1e-05	0.0006427764892578125
Runtime 4	6.508827209472 656e-05	9.77516174316406 2e-05	9.70363616943359 4e-05	0.0006642341613769531
Runtime 5	6.675720214843 75e-05	9.70363616943359 4e-05	9.79900360107421 9e-05	0.0006699562072753906

Melihat tabel tersebut, menunjukkan bahwa algoritma *forward chaining* tersebut memiliki nilai runtime yang secara berturut-turut jauh lebih kecil dibandingkan dengan algoritma lainnya. Algoritma *forward chaining* memiliki waktu runtime dengan skala terlama yaitu sebesar 6.67572021484375e-05 second dan tercepat yaitu 6.198883056640625e-05 second. Kemudian pada posisi kedua memiliki akumulasi kedudukan yang berubah ubah antara backward chaining dengan BFS. Hal ini dikarenakan kedua algoritma tersebut memiliki perbedaan runtime yang cukup dekat. Sedangkan pada algoritma DFS selalu menduduki tempat terakhir dengan waktu runtime terbesar yaitu 0.0007266998291015625 second dan waktu tercepat sebesar 0.0006642341613769531 second

3.3. Metode Pengembangan Sistem

Pengembangan sistem ini akan mengikuti pendekatan Software Development Life Cycle (SDLC) yang terstruktur dan terencana. SDLC adalah serangkaian langkah yang dilakukan oleh tim pengembang perangkat lunak, termasuk programmer dan analis sistem, dalam proses pembuatan sistem informasi atau perangkat lunak. SDLC terdiri dari beberapa tahap utama, yaitu perencanaan, analisis kebutuhan, desain, pengembangan, pengujian, implementasi, evaluasi, dan peluncuran [4]. Tahap perencanaan merupakan langkah awal dalam SDLC, tim pengembang akan mengidentifikasi tujuan dan kebutuhan aplikasi yang akan dikembangkan. Mereka akan menganalisis kebutuhan pengguna, mempertimbangkan aspek teknis dan anggaran yang tersedia, serta merencanakan jadwal dan sumber daya yang dibutuhkan. Tahap berikutnya adalah analisis kebutuhan, di mana tim akan mengumpulkan informasi lebih lanjut tentang kebutuhan pengguna dan membuat spesifikasi yang jelas untuk aplikasi. Setelah itu, tim akan memasuki tahap desain, di mana mereka akan merancang struktur sistem, antarmuka pengguna, dan komponen-komponen lainnya. Dalam pengembangan aplikasi ini, framework frontend Next.js dipilih untuk membangun antarmuka pengguna yang responsif dan interaktif. Dengan bantuan framework CSS seperti Tailwind CSS, tim dapat dengan mudah mengatur tata letak dan gaya desain yang sesuai. Setelah tahap desain, tim akan memulai tahap pengembangan, di mana mereka akan mulai mengkodekan aplikasi berdasarkan desain yang telah ditetapkan. Dalam kasus ini, bahasa pemrograman Python dan framework Flask digunakan sebagai teknologi server-side. Data yang diterima dari klien dalam bentuk JSON akan diproses, dan nilai jawaban pengguna akan disimpan dalam database.



Gambar 3. Sistem Pakar Rekomendasi Peminatan

Setelah tahap pengembangan, tahap pengujian akan dilakukan untuk memastikan bahwa aplikasi berfungsi dengan baik dan sesuai dengan kebutuhan yang telah ditetapkan. Tim pengembang akan melakukan pengujian fungsionalitas, pengujian integrasi, dan pengujian kinerja untuk memastikan bahwa aplikasi memberikan hasil yang akurat dan responsif. Setelah melewati tahap pengujian, aplikasi siap untuk diimplementasikan. Tahap implementasi melibatkan proses peluncuran aplikasi ke lingkungan produksi, memastikan bahwa aplikasi dapat diakses dan digunakan oleh pengguna sesuai dengan rencana yang telah ditentukan. Setelah aplikasi diimplementasikan, tahap evaluasi dilakukan untuk mengevaluasi kinerja aplikasi dan merespon umpan balik dari pengguna. Tim pengembang akan memantau dan menganalisis data

penggunaan aplikasi, mengidentifikasi area perbaikan atau peningkatan yang diperlukan, dan melakukan pemeliharaan rutin untuk memastikan kelancaran aplikasi. Dengan pendekatan pengembangan yang terstruktur menggunakan SDLC dan penggunaan algoritma forward chaining yang efisien, diharapkan aplikasi ini dapat memberikan rekomendasi minat bidang teknologi secara cepat, akurat, dan dapat diandalkan kepada pengguna. Selain itu, dengan menggunakan teknologi seperti Next.js, Tailwind CSS, Flask, dan Python, aplikasi ini dapat memberikan pengalaman pengguna yang baik dan antarmuka yang menarik.

4. Kesimpulan

Berdasarkan pembahasan yang dijabarkan maka dapat disimpulkan bahwa pada segi implementasi sistem informasi algoritma sistem pakar yang memiliki nilai runtime terkecil adalah *forward chaining* dengan skala terlama yaitu sebesar 6.67572021484375e-05 second dan tercepat yaitu 6.198883056640625e-05 second. Sehingga pada pengembangan aplikasi sistem pakar peminatan bidang informatika sangat dianjurkan menggunakan *forward chaining* dari segi kecepatan runtime sistem.

Daftar Pustaka

[1] R. Rosnelly and U. P. Utama, *Sistem Pakar: Konsep dan Teori*. Penerbit Andi.

[2] R. Apriliyani1, F. Tyas Ayuning, and E. Permatasari Kristi, “Perbandingan Metode Forward Chaining dan Backward Chaining pada Sistem Pakar Identifikasi Gaya Belajar,” *Jurnal Informatika dan Teknologi Komputer*, vol. 03, no. 02, pp. 84–92, 2022.

[3] S. bin Uzayr, “TypeScript,” in *TypeScript for Beginners*, Boca Raton: CRC Press, 2022, pp. 1–46. Accessed: Jun. 11, 2023. [Online]. Available: http://dx.doi.org/10.1201/9781003203728-1

[4] Y. Dwanoko Seby, “Implementasi software development life cycle (sdlc) dalam penerapan pembangunan aplikasi perangkat lunak,” *Jurnal Teknologi Informasi*, vol. 7, no. 2, 2016.

[5] F. Doglio, “Developing Your REST API,” in *Pro REST API Development with Node.js*, Berkeley, CA: Apress, 2015, pp. 123–166. Accessed: Jun. 11, 2023. [Online]. Available: http://dx.doi.org/10.1007/978-1-4842-0917-2_7