

Mohamad Farid Hendianto 2200018401 A Kecerdasan Buatan UKI

1. Jelaskan 2 hal yang membedakan aplikasi cerdas dengan aplikasi konvensional. Aplikasi cerdas (Intelligent Applications) dan aplikasi konvensional memiliki dua paradigma berbeda dalam pengembangan perangkat lunak. Perbedaan utama terletak pada kemampuan aplikasi cerdas untuk belajar, beradaptasi, dan berkembang seiring waktu. Kemampuan aplikasi konvensional tetap statis dan bertujuan pada instruksi yang telah diprogram sebelumnya.

Penikut dua aspek utama yang membedakan kedua tipe aplikasi:

1) Kemampuan pembelajaran dan adaptasi;

Sedangkan sifat perbedaan utama antara aplikasi cerdas dan aplikasi konvensional terdapat pada kemampuan pembelajaran dan adaptasi yang dimiliki oleh aplikasi cerdas. Aplikasi konvensional dirancang dengan algoritma arsitektur dan logika yang telah ditentukan sebelumnya, sedangkan aplikasi cerdas memiliki kemampuan untuk membelajari dan beradaptasi dengan data dan situasi baru.

Aplikasi cerdas menggunakan teknik-teknik seperti machine learning dan deep learning untuk menganalisis data dan mendefinisikan pola-pola yang tidak terlihat secara eksplisit. Melalui proses relasional yang intensif, aplikasi cerdas dapat menemukan hubungan kompleks antara input dan output, serta menjalankan modelnya secara dinamis sesuai dengan makna data baru.

Ciri khas dari kemampuan pembelajaran ini dapat dilihat dalam berbagai bidang, seperti rekomendasi produk, penerjemahan bahasa, dan rekomendasi film. Misalkan, sistem virtual cerdas seperti Alexa atau Siri memiliki basis dalam memahami dan memahami permintaan suara pengguna seiring dengan bertambahnya data percakapan yang mereka temui. Sistem rekomendasi produk di situs e-commerce juga semakin akurat dalam memberikan saran berdasarkan preferensi dan perilaku belanja pengguna.

Kemampuan adaptasi aplikasi cerdas memungkinkan mereka untuk terus berkembang dan meningkatkan kinerjanya seiring waktu, tanpa perlu diprogram ulang secara manual, ini sangat berbeda dengan aplikasi konvensional yang hanya dapat melakukannya tugas-tugas yang telah ditentukan sebelumnya dan tidak dapat beradaptasi secara otomatis dengan perubahan lingkungan atau permintaan pengguna.

Alasan dari di bawah perbedaan ini adalah bahwa dunia masa sekarang lebih kompleks, dinamis, dan tidak pasti. Dengan kemampuan pengetahuan dan adaptasi, aplikasi cerdas dapat lebih baik dalam mencari solusi yang tidak terduga dan memberikan solusi yang lebih relevan dengan akar permasalahan dibandingkan dengan aplikasi konvensional yang saat itu.

2) Pengembangan teknologi kompleks dan penalaran

Perbedaan utama lainnya antara aplikasi cerdas dan konvensional berletak pada kemampuan pengembangan teknologi kompleks dan penalaran yang dimiliki oleh aplikasi cerdas. Aplikasi konvensional biasanya menggunakan logika sederhana dan aturan-aturan yang telah ditentukan sebelumnya untuk mengambil keputusan, sedangkan aplikasi cerdas dapat melakukan penalaran yang lebih kompleks dan cermat. Keputusian yang dibuat!

Aplikasi cerdas menggunakan teknik-teknik seperti jaringan saraf tiruan (neural networks), logika fuzzy, dan sistem pakar untuk memproses informasi dan membuat keputusan yang lebih kompleks. Mereka dapat menganalisis data dari berbagai sumber, menangkap nuansa dan intuisi, serta membuat kesimpulan dan rekomendasi yang lebih canggih.

Contoh nyata dari kelebihan pengembangan teknologi kompleks ini dapat dilihat dalam bidang seperti diagnosis medis, analisis seismik, dan perencanaan logistik. Misalkan, sistem diagnosis cerdas dalam bidang medis dapat menganalisis gejala, riwayat medis, dan hasil tes untuk memberikan diagnosis yang lebih akurat dan rekomendasi pengobatan yang lebih tepat. Sistem analisis seismik cerdas dapat memperkirakan berbagai faktor seperti frekuensi gelombang,幅值, dan durasi untuk mendekati sentral involunter yang lebih tepat.

Kemampuan penalaran aplikasi cerdas juga memungkinkan merasakan antara memberikan penjelasan dan alasan dibalik keputusan yang dibuat, sehingga yang sulit dibuktikan oleh aplikasi konvensional yang hanya mengikuti aturan yang telah ditentukan. Ini menjadikan aplikasi cerdas lebih transparan dan dapat di pertanggungjawabkan, serta meningkatkan kredibilitas pengguna untuk lebih memahami proses pengembangan teknologi.

Alasan lain di bawah perbedaan ini adalah bahwa dunia masa sekarang memiliki tingkat kompleksitas yang tinggi, dengan berbagai faktor yang saling berkait dan saling berpengaruh. Dengan kemampuan pengembangan teknologi yang kuat dan penalaran, aplikasi cerdas dapat menyajikan solusi yang lebih akurat, ketepatan, dan dapat diandalkan dalam menghadapi situasi yang sukar dan tidak pasti.

2. Komponen utama dalam Aplikasi Artificial Intelligence adalah mesin inferensi. Definisi cara kerja mesin inferensi dalam proses mencari solusi?

Mesin inferensi atau inferential engine merupakan komponen utama dalam algoritma pencarian berbasis Artificial Intelligence (AI) yang bertanggung jawab untuk melaksanakan perintah logis dan mengambil keputusan berdasarkan representasi yang tersedia. Cara kerja mesin inferensi dalam proses mencari solusi melibatkan berbagai tahapan dan metode yang kompleks. Pemahaman mendalam tentang mekanismenya sangat penting untuk memahami bagaimana sistem AI dapat memecahkan masalah dan membuat keputusan cerdas.

Sebelum membagikan cara kerja mesin inferensi, perl diperbaiki terlebih dahulu konsep dasar dari representasi pengetahuan dan perintah dalam sistem AI. Pengetahuan dalam sistem AI biasanya direpresentasikan dalam bentuk basis pengetahuan (knowledge base) yang terdiri dari faktor-faktor, aturan, dan hubungan antar simbol. Basis pengetahuan ini dapat dibangun menggunakan berbagai teknik representasi pengetahuan, seperti logika, predikat, jaringan semantik, atau sistem berbasis aturan. Mesin inferensi beroperasi pada basis pengetahuan ini untuk melakukan perintah dan melahirkan kesimpulan. Proses penalaran ini dilakukan dengan menyelesaikan metode inferensi, yang pada dasarnya adalah algoritma yang mengambil premis-premis (faktor-faktor yang dikenal) dan mengarurasinya dengan aturan-aturan tertentu untuk menarik kesimpulan baru dari premis tersebut.

Terdapat dua pendekatan utama dalam mesin inferensi, yaitu penalaran maju (forward chaining) dan penalaran mundur (backward chaining). Penalaran maju dimulai dari faktor-faktor yang diketahui dan menggariskan aturan-aturan untuk melahirkan kesimpulan baru, sedangkan penalaran mundur dimulai dari tujuan atau kesimpulan yang ingin dicapai dan melanjutkan faktor-faktor yang mendukung kesimpulan tersebut. Dalam penalaran maju mesin inferensi dibandingkan dengan menggunakan bukti sementara yang dikeluarkan dari basis pengetahuan. Kemudian, mesin inferensi mencocokkan faktor-faktor tersebut dengan premis atau aturan yang ada dalam basis pengetahuan. 2.2. Premis dari sifat-sifat terpenuhi, maka mesin inferensi akan menarik kesimpulan yang baru dari aturan tersebut dan melambahkannya dalam basis pengetahuan sementara (working memory). Proses ini berlanjut secara iteratif, dengan mesin inferensi hanya mencocokkan faktor-faktor yang dipenuhi dengan aturan-aturan yang ada, sehingga tidak ada lagi kesimpulan

bisa yang dapat dituliskan.

Dalam penalaran modus, mesin inferensi dimulai dengan tujuan atau kesimpulan yang ingin dicapai. Kemudian mesin inferensi mencari antecedent atau aturan yang dimulai & klaim pada yang selanjutnya dengannya berikan tersebut. Untuk setiap aturan yang ditemui, mesin inferensi akan mencari bukti yang mendukung premis aturan tersebut. Jika premis tidak terbukti, mesin inferensi akan mencari aturan-aturan lain yang dapat menunjang premis tersebut, dan proses ini berlanjut secara recursive hingga mesin inferensi mendapat suatu-satu yang mendukung bukti atau mencari kesimpulan berikan tujuannya tidak dapat dipenuhi dengan klaim-klaim yang ada.

Bahan proses penalaran mesin inferensi juga dapat diorganisasikan berdasarkan strategi dan teknik untuk meningkatkan efisiensi dan efektivitas penalaran. Beberapa teknik yang umum digunakan adalah:

•) Penalaran berbasis kasus (case-based reasoning):

Mesin inferensi menggunakan kasus-kasus yang telah disimpan dalam bentuk "IF - THEN" untuk memproses klasifikasi dari kasus-kasus yang diberikan.

•) Penalaran berbasis aturan (rule-based reasoning):

Mesin inferensi menggunakan aturan-aturan yang disampaikan dalam bentuk "IF - THEN" untuk memproses klasifikasi dari kasus-kasus yang diberikan.

•) Penalaran fuzzy (fuzzy reasoning):

Mesin inferensi dapat menggunakan ketiga bagian pengetahuan dalam penalaran dalam penggunaan logika fuzzy.

•) Penalaran probabilitik (probabilistic reasoning):

Mesin inferensi menggunakan teori probabilitas untuk mengambil ketiga bagian pengetahuan berdasarkan kemungkinan atau distribusi probabilitif.

•) Penalaran temporal (temporal reasoning):

Mesin inferensi dapat mengambil pengetahuan yang melibatkan aspek waktu, seperti urutan kegiatan atau durasi.

•) Penalaran spasial (spatial reasoning):

Mesin inferensi dapat mengambil pengetahuan yang melibatkan aspek spasial, seperti lokasi, jarak, atau orientasi.

Selain teknik-teknik di atas, mesin inferensi juga dapat dioptimalkan dengan menggunakan berbagai strategi optimal, seperti Pemotongan pencarian (search pruning), Pemangkasan (pruning), atau Penerapan pola (pattern recognition).

Dalam konteks aplikasi AI yang kompleks, mesin inferensi sebaiknya dilengkapi dengan komponen-komponen lain seperti sistem Pengenalan pola, Pembelajaran mesin (Machine learning), atau Pengolahan bahasa alami (natural language processing). Integrasi ini meningkatkan sistem AI untuk mengambil keputusan atau tindak yang lebih akurat dan sional dalam menyelesaikan berbagai tugas berperilaku dan teknik penalaran yang berbeda.

3. Jelaskan adanya relasi proses pencarian dengan mekanisme pencarian buta. Perhati sangat beras dalam memory atau media penyimpanan?

Pelacakan buta, yang juga dikenal sebagai pencarian tanpa informasi (uninformed search), merupakan sebuah metode dalam kecerdasan buatan yang digunakan untuk menemukan solusi terhadap suatu permasalahan dengan mencoba semua kemungkinan secara sistematis tanpa adanya Pengolahan atau informasi tambahan tentang lingkungan tersebut. Salah satu algoritma pelacakan buta yang paling sederhana dan terkenal adalah Penalaran breadth-first (breadth-first search, BFS).

Dalam BFS, pencarian dimulai dari sebuah simpul awal (root node) dan memperluas semua simbol tetangga (neighbor nodes) pada level yang sama sebelum melanjutkan ke level berikutnya. Proses berlangsung hingga solusi ditentukan atau semua kemungkinan telah dicobai.

Nah, alasan utama mengapa pelacakan buta, khususnya BFS, sangat boros dalam penggunaan memori atau media penyimpanan terletak pada fakta bahwa algoritma yang membutuhkan penyimpanan semesta (embarrassingly parallel) untuk mengetahui semua simbol yang telah dicobai, atau sedang dalam antara untuk diambilnya. Selain itu, sangat banyak perubahan (search space) dan semakin kompleks.

Permasalahan yang dihadapi, selain batasan memori yang dibutuhkan untuk menyimpan informasi ini, pelacakan buta sangat boros dalam penggunaan memori atau media penyimpanan dalam memperluas beberapa faktor penting yang berkaitan dengan permasalahan.

1) EXPLORATIVE EXHAUSTIVE (Exhaustive Exploration)

Grafik satu karakteristik yang dari pelacakan bisa dikenal saat eksplorasi exhaustif yang berarti algoritma akan mencoba semua kemungkinan secara sistematis tanpa ada informasi atau peragahan tambahan mana dapat membantu ruang pencakalian. Metrik pemaksaan ini membatasi bahwa sebagian besar algoritma (misalnya) nonun hal tersebut juga mengaburkan penggrafen drastis dalam simbol simbol yang perlu dieliminasi dalam penemuan.

Pada kasus terbatas, algoritma pelacakan butuh meningkatkan hasil pencakalian semua simbol dalam ruang pencakalian sebelum melanjutkan hal itu. Jumlah simbol ini dapat sangat besar, bahkan untuk permasalahan yang relatif sederhana. Sebagai contoh, dalam permainan satuk, jumlah kemungkinan posisi papan catur yang sedi pelacakan mencapai sekitar 10^{47} atau satu setitikun potisi. Tapi algoritma pelacakan butuh dijalankan untuk menyelesaikan permainan cumania hasil memimpin informasi tentang semua posisi dan di dalam algoritma yang butuh saja merupakan tugas yang sangat berat dan tidak praktis.

2) PANDUKI PENEMUAN (Search Heuristic)

Selain eksplorasi exhaustive, pelacakan butuh juga cenderung menggunakan teknik pencakalian yang berarti algoritma mungkin melewati simbol yang sama berulang kali sedangkan pelacakan yang berhenti. Hal ini disebabkan oleh sifatnya informasi atau peragahan yang simbol algoritma tentu lingkaran permasalahan sehingga ia tidak dapat membedakan simbol yang telah dicari sebelumnya.

Untuk mengatasi masalah tersebut ini, algoritma pelacakan butuh memimpin informasi tentang semua simbol yang telah dicari ini dalam bentuk sehingga dapat menghindari pencakalian simbol yang sama berulang kali. Namun, bagaimana informasi ini pengetahuan bantuan butuh dari memori yang signifikan, terutama jika jumlah simbol dalam ruang pencakalian sangat besar.

3) STRUKTUR DATA KAMPEKES (Complex Data Structures)

Untuk menyimpan informasi tentang simbol yang telah dicari ini, dan simbol yang belum dalam antara untuk dikunjungi dan simbol yang belum dalam antara untuk dikunjungi, algoritma pelacakan butuh sejumlah kunci manajemen struktur data yang kompleks seperti antara (array) atau tumpukan (stack). Struktur data ini membutuhkan algoritma memori yang besar dan operasi pencakalian serta penghapusan data yang harus serupa dengan

Selain itu, dalam beberapa kasus, algoritma pelacakan bisa juga perlu memimpin informasi tambahan tentang rasio profit dan risiko bisnis pelacakan (cost-to-reach) atau pun langsung yang diambil untuk mendukung rencana dan memastikan solusi yang ditemui adalah salah satu yang optimal. Namun, pemimpinan informasi tambahan ini juga memiliki batasan memori secara signifikan.

4.) Keterbatasan memori memotong pencarian (Habit to Prune Search)

Salah satu keterbatasan utama dari pelacakan bisa adalah ketidakcukupan ruang memori (Prml) atau membatasi ruang pelacakan secara efisien. Ketika kurangnya informasi atau pengetahuan tentang lingkungan permakalahan algoritma pelacakan bisa tidak dapat membedakan simbol yang berpotensi mengarah ke solusi dari simbol yang tidak produktif.

Akibatnya, algoritma terpaksa melepas jalur; seolah lewati solusi permukaan sederhana yang tidak produktif atau bahkan kiteratur-pada (cyclic). Hal ini merupakan pengalaman pentingnya sebuah pelacakan mencari solusi secara signifikan, karena sejauh informasi tentang simbol dalam suatu pencarian yang tidak produktif juga harus disimpan.

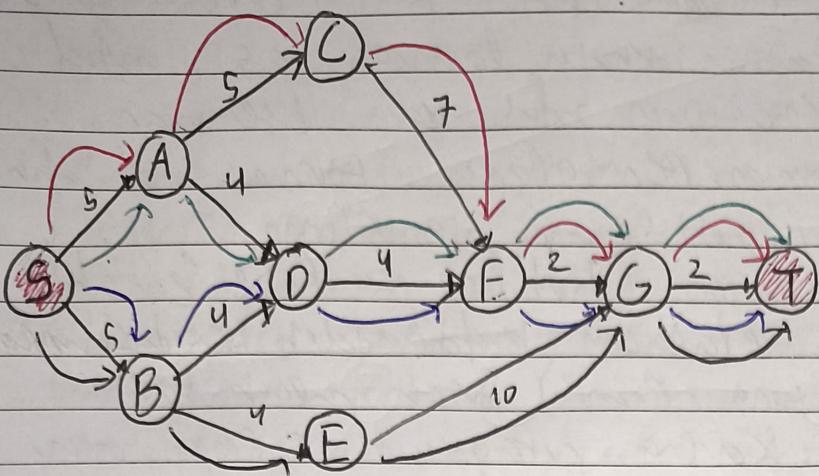
5.) Keterbatasan komputasi (Computational Limitations)

Selain masalah memori, pelacakan bisa juga menghadapi keterbatasan komputasi lainnya seperti waktu eksekusi yang lama dan penggunaan sumber daya proses yang banyak. Semakin besar ruang pencarian dan kompleks permakalahan, semakin banyak sumber daya komputasi yang dibutuhkan untuk melaksanakan eksplorasi exhaustif.

Dalam beberapa kasus, waktu pencarian yang dibutuhkan untuk melaksanakan permakalahan dengan pelacakan bisa sangat tidak praktis atau bahkan tidak mungkin. Selain itu, faktor-faktor seperti eksplorasi exhaustif, teknologi pencarian dan keterbatasan memori memotong pencarian yang telah dijabarkan sebelumnya.

Oleh karena itu, pelacakan bisa sendiri tidak cocok diaplikasikan untuk melaksanakan permakalahan yang kompleks dengan ruang pencarian yang sangat besar, kecuali jika sumber daya cukup banyak dan komputasi yang besar juga dijamin.

4) Graph berikut ini:



Rute I

$$5 + 5 + 7 + 2 + 2 = 21$$
$$S \rightarrow A \rightarrow C \rightarrow F \rightarrow G \rightarrow T$$

Rute II

$$5 + 4 + 4 + 2 + 2 = 17 \quad (\text{optimal})$$
$$S \rightarrow A \rightarrow D \rightarrow F \rightarrow G \rightarrow T$$

Rute III

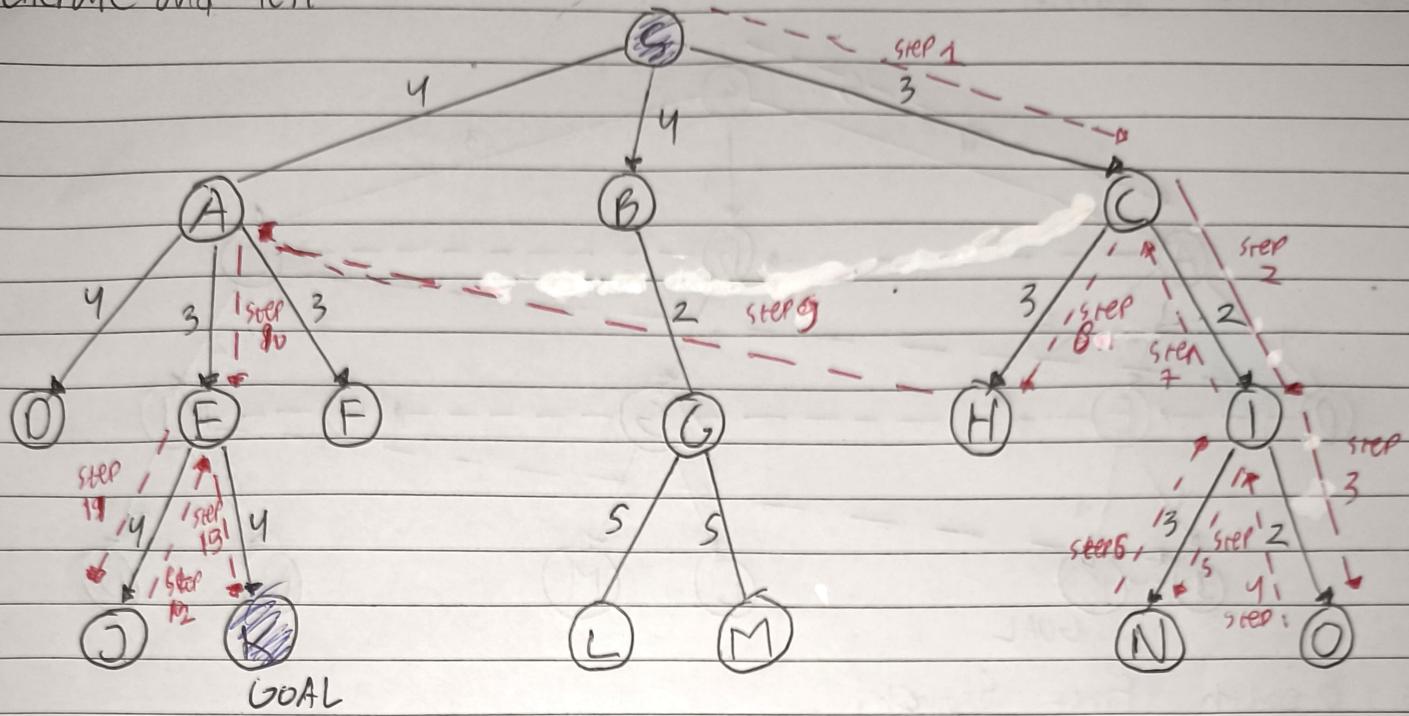
$$5 + 4 + 4 + 1 + 2 = 17 \quad (\text{optimal})$$
$$S \rightarrow B \rightarrow D \rightarrow F \rightarrow G \rightarrow T$$

Rute IV

$$5 + 4 + 10 + 2 = 21$$
$$S \rightarrow B \rightarrow E \rightarrow G \rightarrow T$$

- Maka solusi optimalnya dengan weight 17 yaitu memiliki 2 rute
- $S \rightarrow A \rightarrow D \rightarrow F \rightarrow G \rightarrow T$. dan
- $S \rightarrow B \rightarrow D \rightarrow F \rightarrow G \rightarrow T$

Generate and Test

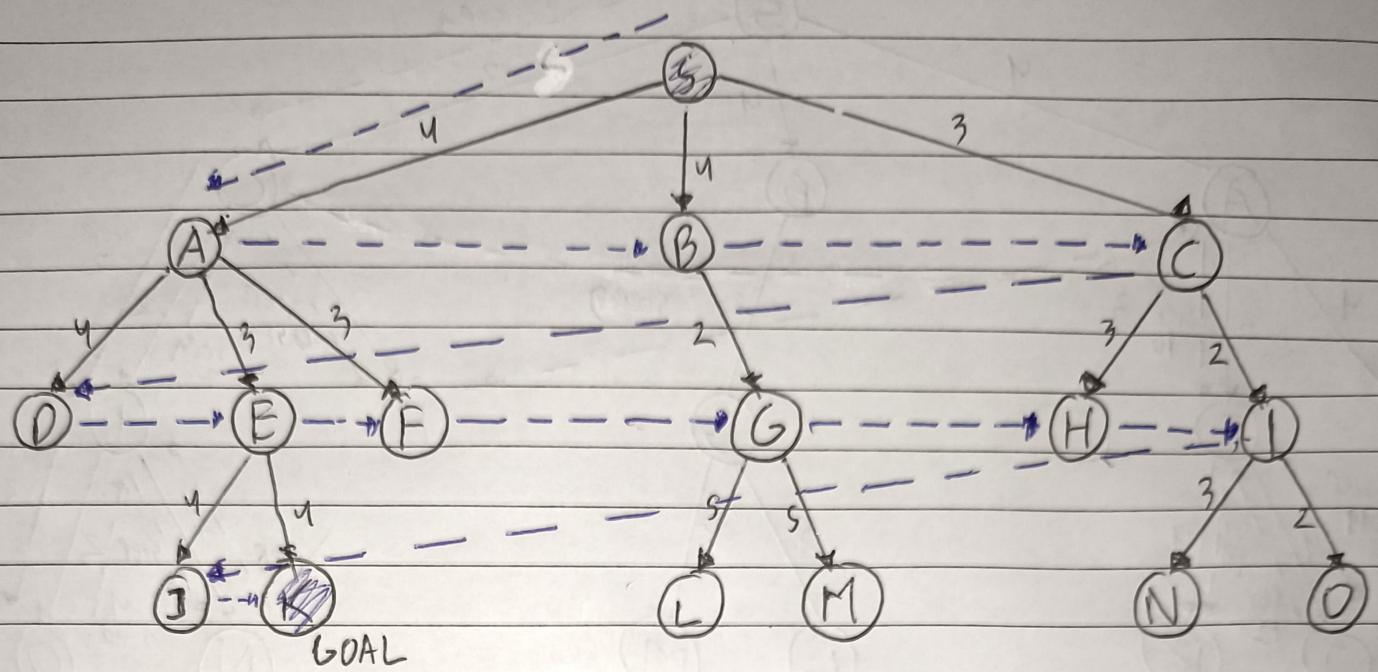


Solusi untuk generate and test:

$$S \rightarrow C \rightarrow I \rightarrow O \rightarrow I \rightarrow N \rightarrow I \rightarrow C \rightarrow H \rightarrow A \rightarrow E \rightarrow J \rightarrow E \rightarrow K$$

Jika seluruh di telusuri dengan generate and test maka ditemukan solusi optimal
 $S \rightarrow A \rightarrow E \rightarrow K$ dengan weight 11

Breadth First Search



Solusi Breadth First Search:

$S \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow H \rightarrow I \rightarrow J \rightarrow K$

Dari Breadth First Search diketahui rule optimal yaitu

$S \rightarrow A \rightarrow E \rightarrow K$ dengan weight 11