Bab 4

Algoritma Pencarian (*Searching Algorithm*)

POKOK BAHASAN:

- ✓ Macam-macam Algoritma Pencarian
- ✓ Mendefinisikan Permasalahan sebagai Ruang Keadaan
- ✓ Pencarian Buta (Depth First, Breadth First, Hill Climbing, Beam, Best First)
- ✓ Pencarian Optimal (British Musium, Branch and Bound, Dynamic Programming, A*)
- ✓ Game 2 pemain (Minimax, Alpha Beta Prunning)

TUJUAN BELAJAR:

Setelah mempelajari bab ini, mahasiswa diharapkan mampu:

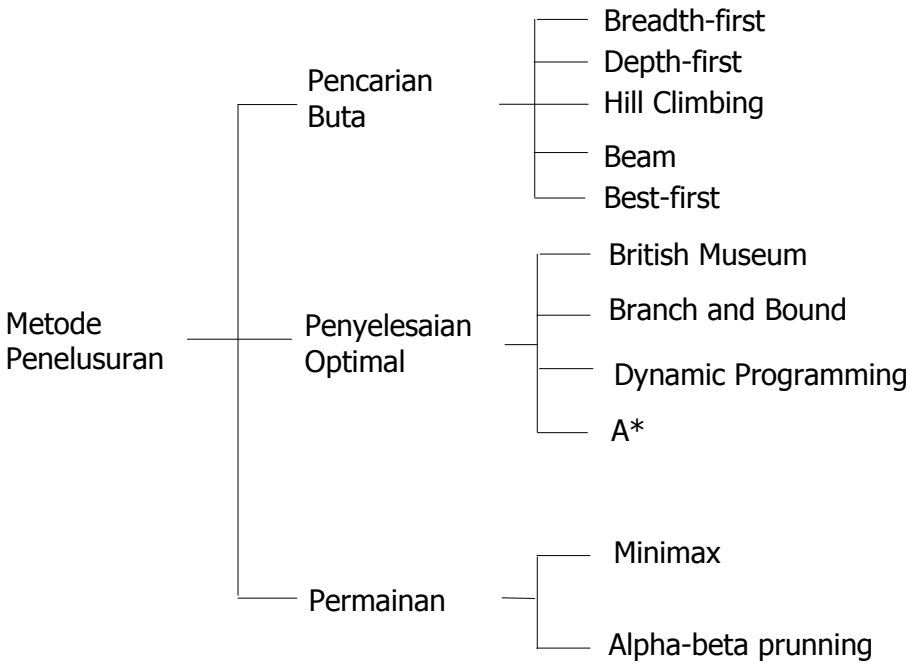
- ✓ Menjelaskan permasalahan kombinatorial, representasinya dan konsekuensinya.
- ✓ Memilih algoritma pencarian buta yang tepat untuk sebuah permasalahan, mengimplementasikan dan karakteristiknya pada kompleksitas waktu dan ruang.
- ✓ Memilih Algoritma Pencarian Heuristic yang tepat dan mengimplementasikan untuk sebuah permasalahan.
- ✓ Menjelaskan kondisi apa yang dijamin oleh sebuah algoritma heuristic sebagai solusi optimal.
- ✓ Mengimplementasikan pencarian minimax dan alpha-beta pruning untuk beberapa game 2 pemain.

4.1 MACAM-MACAM ALGORITMA PENCARIAN

Permasalahan pencarian adalah merupakan yang sering dijumpai oleh peneliti di bidang Kecerdasan Buatan. Permasalahan ini merupakan hal penting dalam menentukan keberhasilan system kecerdasan buatan. Dalam bab ini akan dipelajari 3 bagian dalam metode pencarian, yang pertama adalah metode yang sederhana yang hanya berusaha mencari kemungkinan penyelesaian. Metode yang termasuk pada bagian ini adalah dept-first search, hill climbing, breadth-first search, beam search dan best-first search.

Yang kedua, kita akan mempelajari metode yang lebih kompleks yang akan mencari jarak terpendek. Metode ini adalah British Museum Procedure, Branch and Bound, Dynamic Programming dan A*. Metode-metode ini digunakan pada saat harga perjalanan untuk mencari kemungkinan menjadi perhitungan

Yang ketiga, kita akan mempelajari beberapa prosedur/metode yang kita terapkan saat kita berhadapan dengan musuh. Prosedur ini adalah minimax search, alpha-beta prunning. Metode ini banyak digunakan pada program-program permainan seperti catur dsb. Dalam gambar 4.1 terdapat bagan untuk Metode Searching.



Gambar 4.1 Bagan Metode Penulusuran (Searching)

Metode pencarian dikatakan penting untuk menyelesaikan permasalahan karena setiap state(keadaan) menggambarkan langkah-langkah untuk menyelesaikan permasalahan.

Metode pencarian dikatakan penting untuk perencanaan karena dalam sebuah permainan akan menentukan apa yang harus dilakukan, dimana setiap state menggambarkan kemungkinan posisi pada suatu saat.

Metode pencarian adalah bagian dari kesimpulan, dimana setiap state menggambarkan hipotesis dalam sebuah rangkaian deduktif.

4.2 MENDEFINISIKAN MASALAH SEBAGAI SUATU RUANG KEADAAN

Secara umum, untuk mendeskripsikan suatu permasalahan dengan baik harus:

1 Mendefinisikan suatu ruang keadaan.

2 Menerapkan satu atau lebih keadaan awal.

3 Menetapkan satu atau lebih tujuan.

4 Menetapkan kumpulan aturan.

Contoh 4.2.1 : petani, serigala, angsa dan padi

Contoh ini kita ambil dari bab 4. Permasalahan petani, serigala, angsa dan padi. Seorang petani ingin memindah dirinya sendiri, seekor serigala, seekor angsa gemuk, dan seikat padi yang berisi menyeberangi sungai. Sayangnya, perahunya sangat terbatas; dia hanya dapat membawa satu objek dalam satu penyeberangan. Dan lagi, dia tidak bisa meninggalkan serigala dan angsa dalam satu tempat, karena serigala akan memangsa angsa. Demikian pula dia tidak bisa meninggalkan angsa dengan padi dalam satu tempat.

Dari permasalahan di atas untuk mendefinisikan masalah sebagai ruang keadaan kita tentukan langkah-langkah sebagai berikut:

A. Identifikasi ruang keadaan.

   Permasalahan ini dapat dilambangkan dengan (JumlahSerigala, JumlahAngsa, JumlahPadi, JumlahPetani). Sebagai contoh Daerah asal (0,1,1,1) berarti pada daerah asal tidak ada serigala, ada angsa, ada padi dan ada petani

   tani.

B. Keadaan awal dan tujuan.

- Keadaan awal, pada kedua seberang sungai:
  - Daerah asal: (1,1,1,1)
  - Daerah seberang: (0,0,0,0)
- Tujuan, pada kedua seberang sungai:
  - Daerah asal: (0,0,0,0)
  - Daerah seberang: (1,1,1,1)

C. Aturan-aturan

Aturan-aturan dapat digambarkan seperti pada tabel 4.1.

Tabel 4.1 Aturan-aturan masalah Petani dan Barang Bawaannya

Aturan Ke-	Aturan
1	Angsa menyeberang
2	Padi menyeberang
3	Serigala menyeberang
4	Angsa kembali
5	Padi kembali
6	Serigala kembali
7	Petani kembali

Salah satu solusi yang bisa ditemukan dapat dilihat pada tabel 4.2.

Tabel 4.2 Contoh Solusi Masalah Petani, Serigala, Angsa, dan Padi

Daerah Asal	Daerah Seberang	Aturan yang dipakai
(1,1,1,1)	(0,0,0,0)	1
(1,0,1,0)	(0,1,0,1)	7
(1,0,1,1)	(0,1,0,0)	3
(0,0,1,0)	(1,1,0,1)	4
(0,1,1,1)	(1,0,0,0)	2
(0,1,0,0)	(1,0,1,1)	7
(0,1,0,1)	(1,0,1,0)	1
(0,0,0,0)	(1,1,1,1)	solusi

Contoh 4.2.2: Masalah teko air

Ada 2 buah teko masing-masing berkapasitas 4 galon (teko A) dan 3 galon (teko B). Tidak ada tanda yang menunjukkan batas ukuran pada kedua teko tersebut. Permasalahannya : Bagaimanakah kita dapat mengisikan tepat 2 galon air ke dalam teko yang berkapasitas 4 galon?

Dari permasalahan di atas untuk mendefinisikan masalah sebagai ruang keadaan kita tentukan langkah-langkah sebagai berikut:

A. Identifikasi ruang keadaan.

Permasalahan ini dapat dilambangkan dengan (x,y). Dimana:

- x = air yang diisikan pada teko A
- y = air yang diisikan pada teko B

B. Keadaan awal dan tujuan.

- Keadaan awal, kedua teko dalam keadaan kosong: (0,0).
- Tujuan, keadaan dimana pada teko 4 galon berisi tepat 2 galon air: (2,n) untuk sembarang n.

C. Aturan-aturan

Aturan-aturan dapat digambarkan seperti pada tabel 4.2.

4.2.1 GRAPH KEADAAN

Graph terdiri dari node-node yang menunjukkan keadaan, yaitu keadaan awal dan keadaan baru yang akan dicapai dengan menggunakan operator. Node-node dalam graph keadaan saling dihubungkan dengan menggunakan arc (busur) yang diberi panah untuk menunjukkan arah dari suatu keadaan ke keadaan berikutnya.

Metode pencarian akan berusaha menemukan kombinasi dari item-item yang dimulai dari start menuju ke goal. Dalam gambar 4.2 dilukiskan mengenai graph yang menggambarkan sebuah ruang keadaan dalam Metode Pencarian

Tabel 4.3 Aturan-aturan Masalah Teko Air

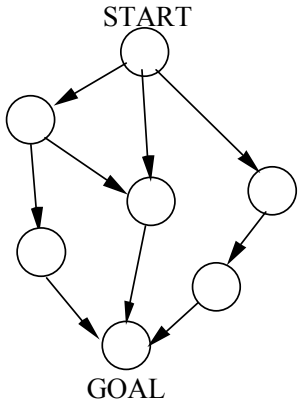
Aturan Ke-	Kondisi Awal	Kondisi Tujuan	Aturan
1	(x,y) dng x < 4	(4,y)	Isi teko A sampai penuh
2	(x,y) dng y < 3	(x,3)	Isi teko B sampai penuh
3	(x,y) dng x > 0	(x-d,y)	Tuangkan sebagian air keluar dari teko A
4	(x,y) dng y > 0	(x,y-d)	Tuangkan sebagian air keluar dari teko B
5	(x,y) dng x > 0	(0,y)	Kosongkan teko A dengan membuang airnya ke tanah
6	(x,y) dng y > 0	(x,0)	Kosongkan teko B dengan membuang airnya ke tanah
7	(x,y) dimana x+y≥4 dan y>0	(4,y-(4-x)	Tuangkan air dari teko B ke teko A sampai teko A penuh
8	(x,y) dimana x+y≥3 dan x>0	(x-(3-y),3)	Tuangkan air dari teko A ke teko B sampai teko B penuh
9	(x,y) dimana x+y≤4 dan y>0	(x+y,0)	Tuangkan seluruh air dari teko B ke teko A
10	(x,y) dimana x+y≤3 dan x>0	(0,x+y)	Tuangkan seluruh air dari teko A ke teko B
11	(0,2)	(2,0)	Tuangkan 2 galon air dari teko B ke teko A
12	(2,y)	(0,y)	Kosongkan 2 galon air di teko A dengan membuang airnya ke tanah

Salah satu solusi yang bisa ditemukan dapat dilihat pada tabel 4.2.

Tabel 4.4 Contoh Solusi Masalah Teko Air

Kondisi Awal	Kondisi Tujuan	Aturan yang dipakai
(0,0)	(0,3)	2
(0,3)	(3,0)	9
(3,0)	(3,3)	2
(3,3)	(4,2)	7
(4,2)	(0,2)	5
(0,2)	(2,0)	9

Transisi dari State atau Perpindahan



State-State dalam Pencarian atau Simpul dalam Graph atau Tree

Gambar 4.2 Sebuah Keadaan dalam Metode Pencarian

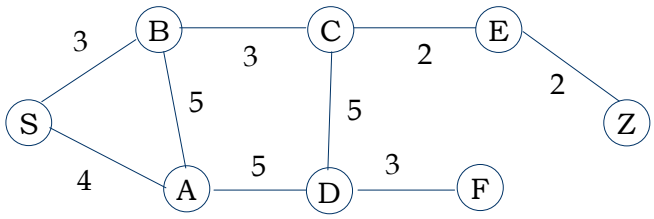
4.2.2 POHON PELACAKAN

Untuk menghindari kemungkinan adanya proses pelacakan suatu node secara berulang, maka digunakan struktur pohon.

Struktur pohon digunakan untuk menggambarkan keadaan secara hirarkis. Pohon juga terdiri dari beberapa node. Node yang terletak pada level-0 disebut juga "akar". Node akar menunjukkan keadaan awal yang biasanya merupakan topik atau obyek. Node akar ini terletak pada level ke-0. Node akar mempunyai beberapa percabangan yang terdiri atas beberapa node successor yang sering disebut dengan nama "anak" dan merupakan node-node perantara. Namun jika dilakukan pencarian mundur, maka dapat dikatakan bahwa node tersebut memiliki predecessor. Node-node yang tidak

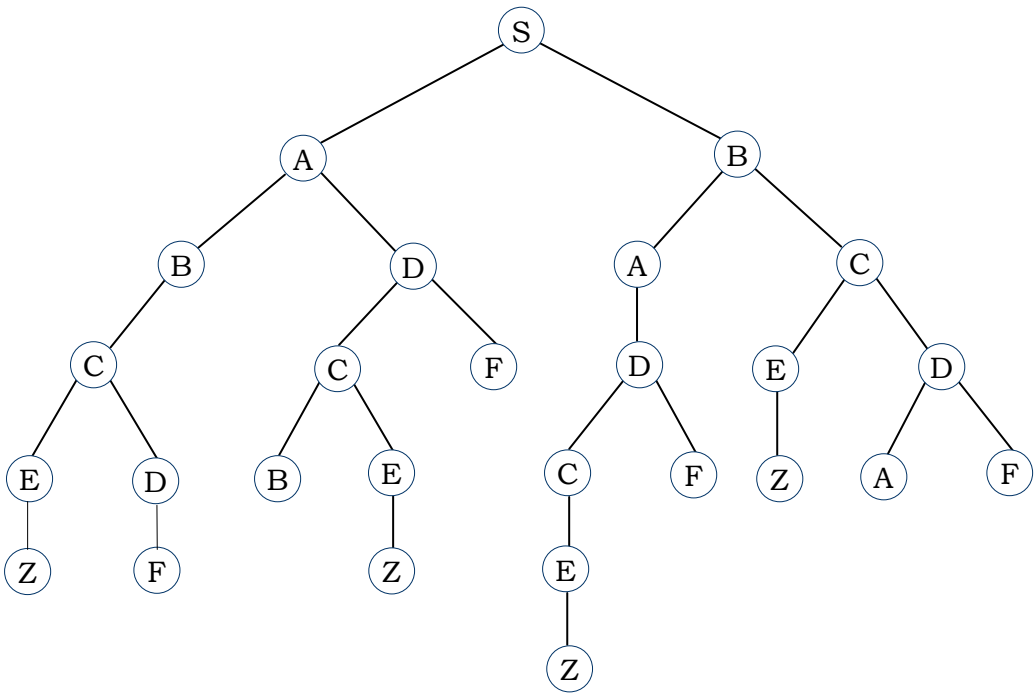
mempunyai anak sering disebut dengan nama node "daun" yang menunjukkan akhir dari suatu pencarian, dapat berupa tujuan yang diharapkan (goal) atau jalan buntu (dead end).

Dalam gambar 4.3 ditampilkan sebuah contoh permasalahan mendasar untuk digunakan dalam penggunaan beberapa metode pencarian. Simpul S merupakan simpul awal dimulainya penelusuran, simpul Z adalah simpul yang akan menjadi tujuan.



Gambar 4.3 Contoh Graph yang Berisi Path Antar Kota

Dari graph di atas, dibuat struktur tree-nya. Pada gambar 4.4 menggambarkan tree yang didapat dari Graph gambar 4.3.

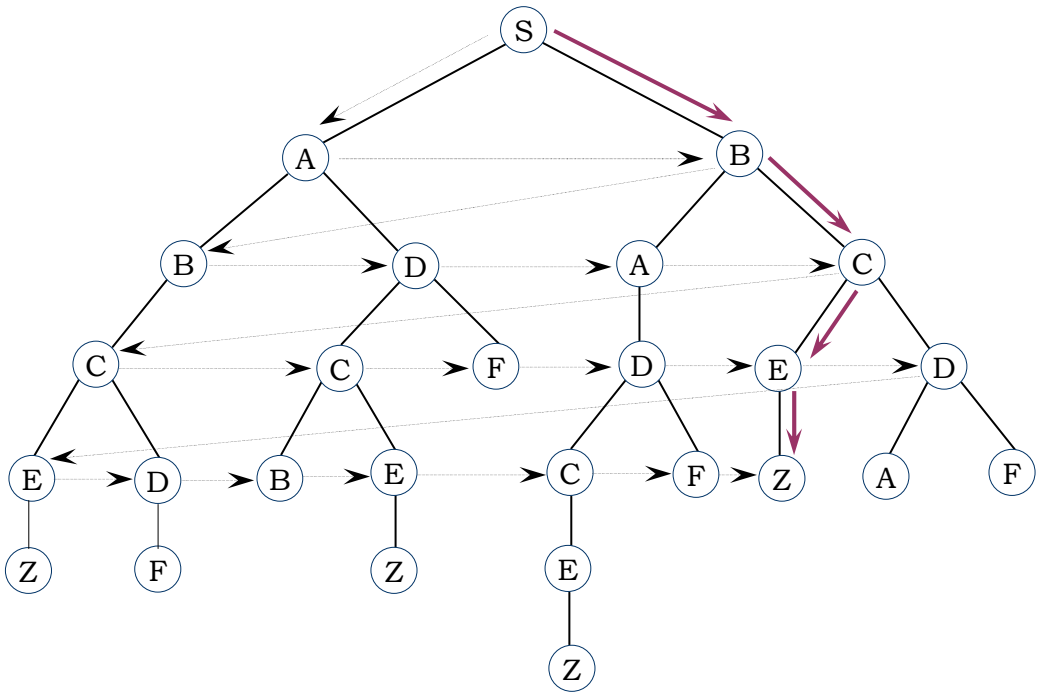


Gambar 4.4 Struktur Tree dari Graph Gambar 4.3

4.3 PENCARIAN BUTA (BREADTH FIRST, DEPTH FIRST, HILL CLIMBING, BEST FIRST)

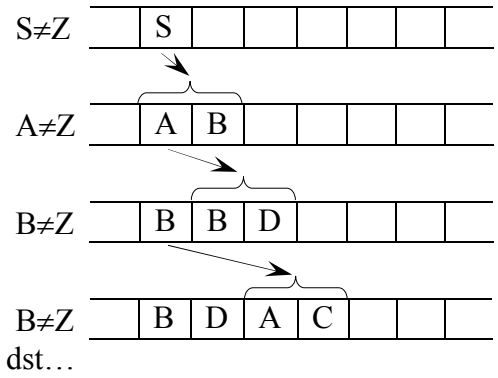
A. Pencarian Melebar Pertama (Breadth-First Search)

Pada metode Breadth-First Search, semua node pada level n akan dikunjungi terlebih dahulu sebelum mengunjungi node-node pada level n+1. Pencarian dimulai dari node akar terus ke level ke-1 dari kiri ke kanan, kemudian berpindah ke level berikutnya demikian pula dari kiri ke kanan sampai ditemukannya solusi.



Gambar 4.5 Tree untuk Breadth First Search

- Algoritma
  1. Buat sebuah Antrian, inisialisasi node pertama dengan Root dari tree
  2. Bila node pertama, jika ≠ GOAL, diganti dengan anak-anaknya dan diletakkan di belakang PER LEVEL
  3. Bila node pertama = GOAL, selesai



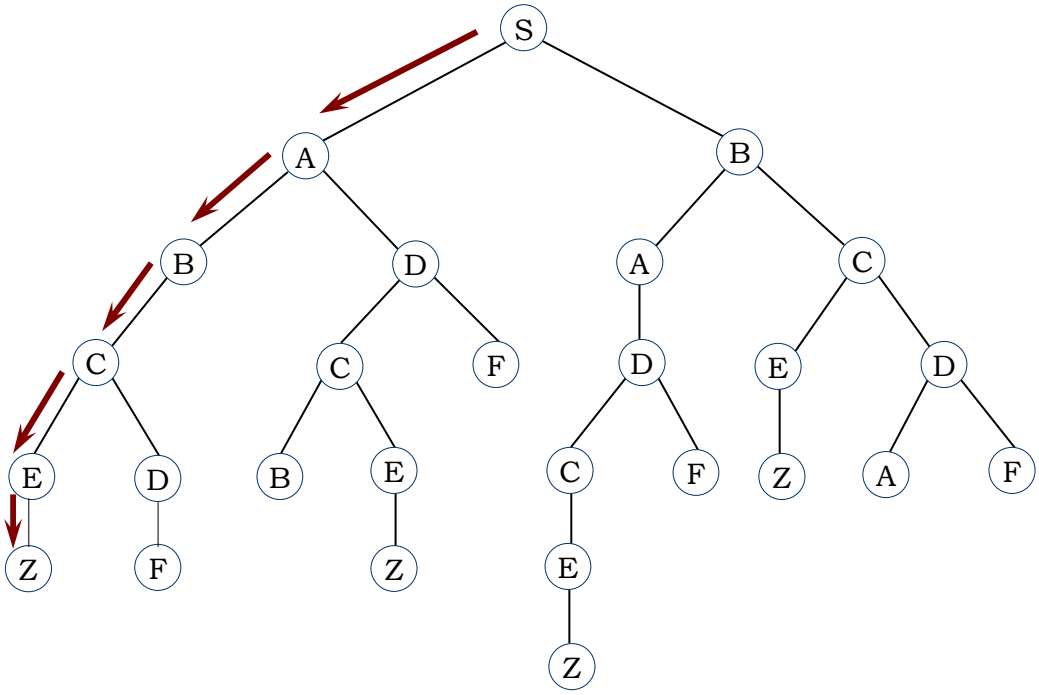
Lintasan yang didapat : S – B – C – E – Z

- Keuntungan
  1. Tidak akan menemui jalan buntu
  2. Jika ada satu solusi, maka breadth first search akan menemukannya. Dan jika ada lebih dari satu solusi, maka solusi minimum akan ditemukan.
- Kelemahan
  1. Membutuhkan memori yang cukup banyak, karena menyimpan semua node dalam satu pohon.
  2. Kemungkinan ditemukan optimal lokal.

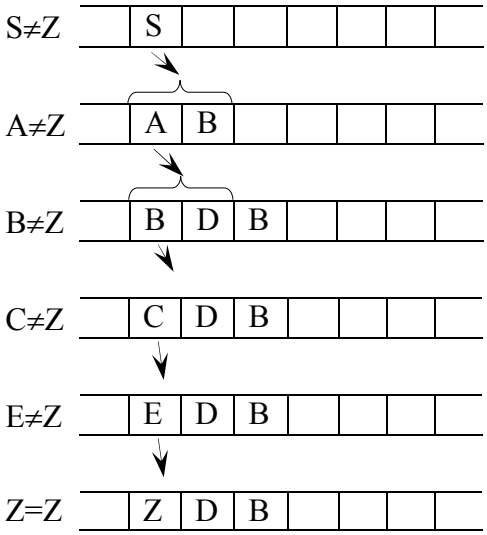
B. Pencarian Mendalam Pertama (Depth-First Search)

Pada Depth First Search, proses pencarian akan dilaksanakan pada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel. Pencarian dimulai dari node akar ke level yang lebih tinggi. Proses ini diulangi terus hingga ditemukaannya solusi.

- Algoritma
  1. Buat sebuah Antrian, inisialisasi node pertama dengan Root dari tree
  2. Bila node pertama, jika ≠ GOAL, node dihapus diganti dengan anak-anaknya dengan urutan LChild
  3. Bila node pertama = GOAL, selesai



Gambar 4.6 Tree untuk Depth First Search

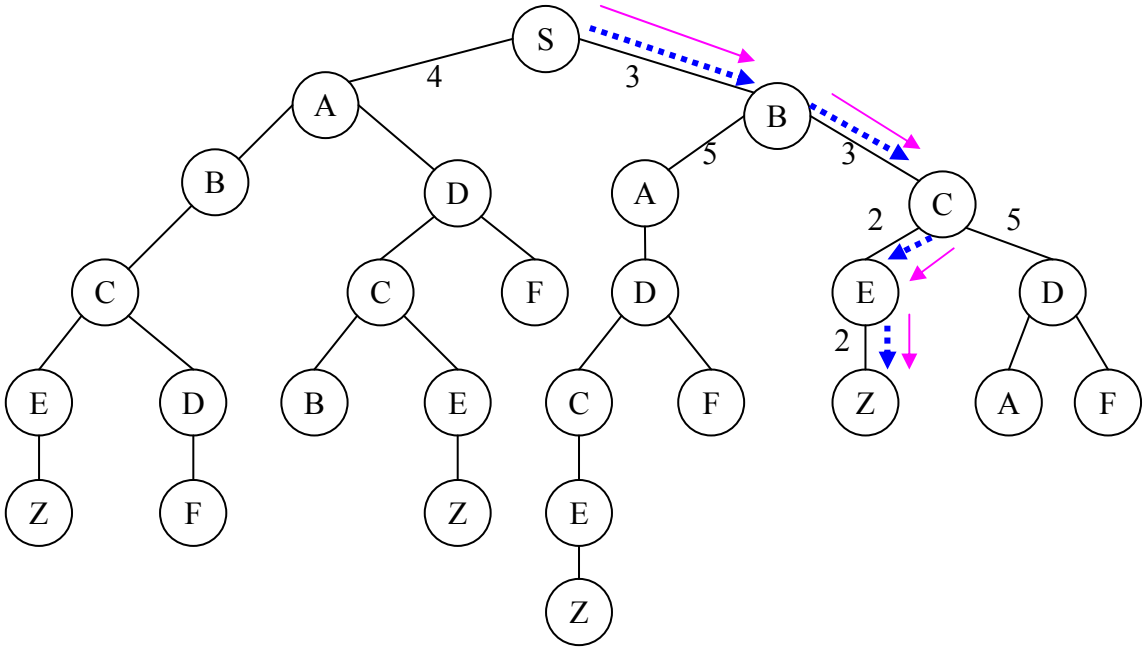


Lintasan yang didapat : S – A – B – C – E – Z

- Keuntungan
  1. Membutuhkan memori yang relative kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
  2. Menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.
- Kelemahan

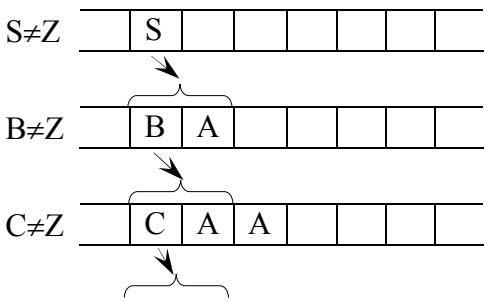
1. Kemungkinan terjebak pada optimal lokal.
2. Hanya akan mendapatkan 1 solusi pada setiap pencarian.

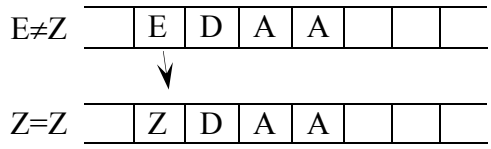
C. Pencarian dengan Mendaki Bukit (Hill Climbing)



Gambar 4.7 Tree untuk Hill Climbing

- Algoritma
  1. Buat sebuah Antrian, inisialisasi node pertama dengan Root dari tree
  2. Bila node pertama, jika ≠ GOAL, node dihapus diganti dengan anak-anaknya dengan urutan yang paling kecil jaraknya
  3. Bila node pertama = GOAL, selesai

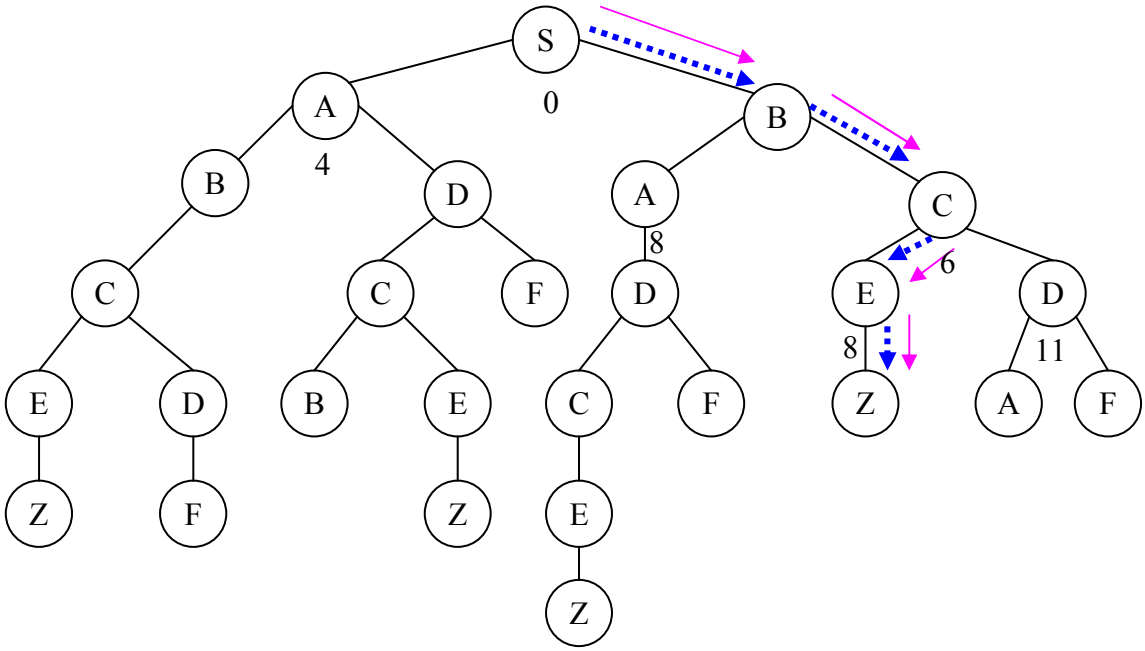




Lintasan yang didapat : S – A – B – C – E – Z

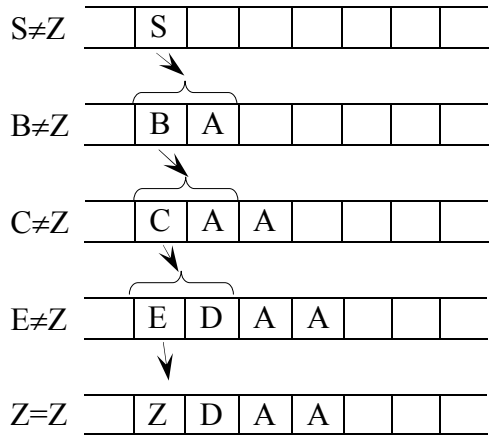
- Keuntungan
  1. Membutuhkan memori yang relative kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
  2. Metode hill climbing search akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.
- Kelemahan
  1. Algoritma akan berhenti kalau mencapai nilai optimum local.
  2. Perlu menentukan aturan yang tepat

D. Pencarian dengan Best First Search



Gambar 4.8 Tree untuk Best First Search

- Algoritma
  1. Buat sebuah Antrian, inisialisasi node pertama dengan Root dari tree
  2. Bila node pertama, jika ≠ GOAL, node dihapus & diganti dengan anak-anaknya. Selanjutnya keseluruhan node yang ada di Queue di-sort Ascending.
  3. Bila node pertama = GOAL, selesai



Lintasan yang didapat : S – A – B – C – E – Z

- Keuntungan
  1. Membutuhkan memori yang relative kecil, karena hanya node-node pada lintasan yang aktif saja yang disimpan.
  2. Secara kebetulan, metode best first search akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.
- Kelemahan
  1. Algoritma akan berhenti kalau mencapai nilai optimum local.
  2. Tidak diijinkan untuk melihat satupun langkah sebelumnya.

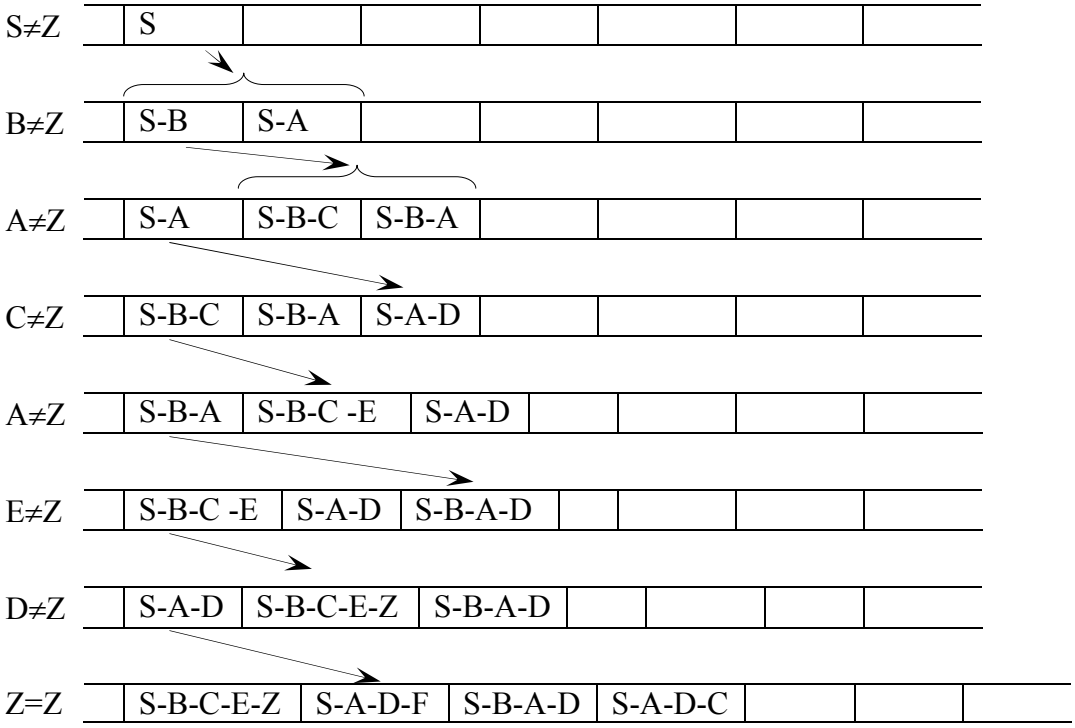
4.4 PENCARIAN OPTIMAL

Jika pada sub bab sebelumnya dipresentasikan metode non-optimal, maka pada sub bab ini dipresentasikan metode optimal.

A. Branch and Bound

Pada metode ini kita tidak memerlukan estimasi jarak tiap node menuju GOAL. Selain itu elemen-elemen pada queue bukan tiap node, melainkan lintasan parsial yang sudah tercapai.

- Algoritma
  1. Buat sebuah Antrian, inisialisasi node pertama dengan Root dari tree
  2. Bila lintasan parsial ≠ lintasan GOAL, maka lintasan parsial diganti dengan lintasan parsial + node child, semuanya diatur berdasarkan harga yang diurut secara ascending.
  3. Bila node pertama = lintasan GOAL, selesai

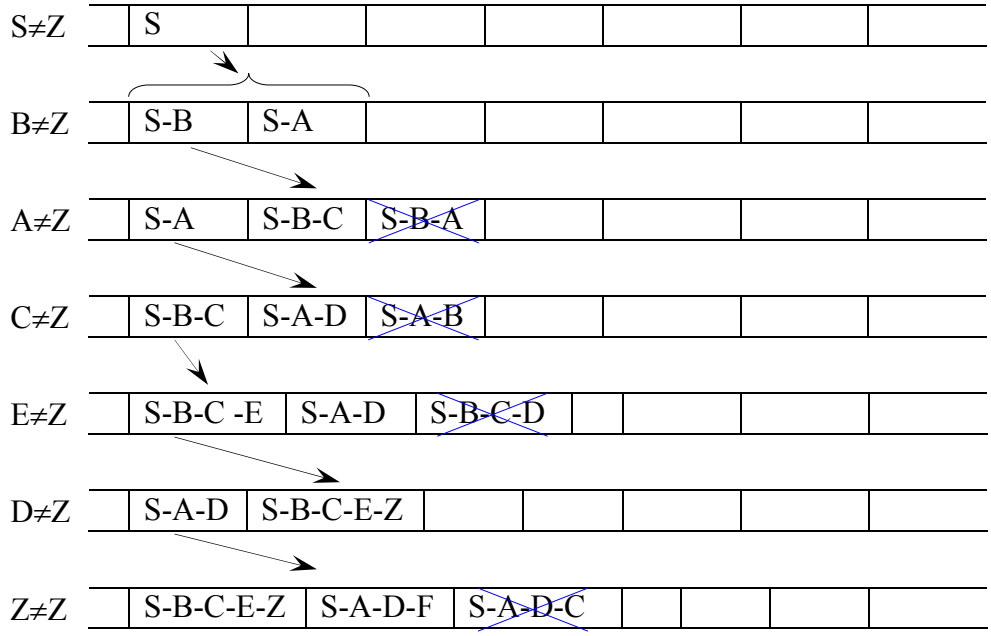


- Keuntungan
  1. Algoritma berhenti pada nilai optimum sebenarnya (menemukan optimum global).
- Kelemahan
  1. Membutuhkan memori yang cukup banyak, karena bisa jadi menyimpan semua lintasan parsial yang memungkinkan

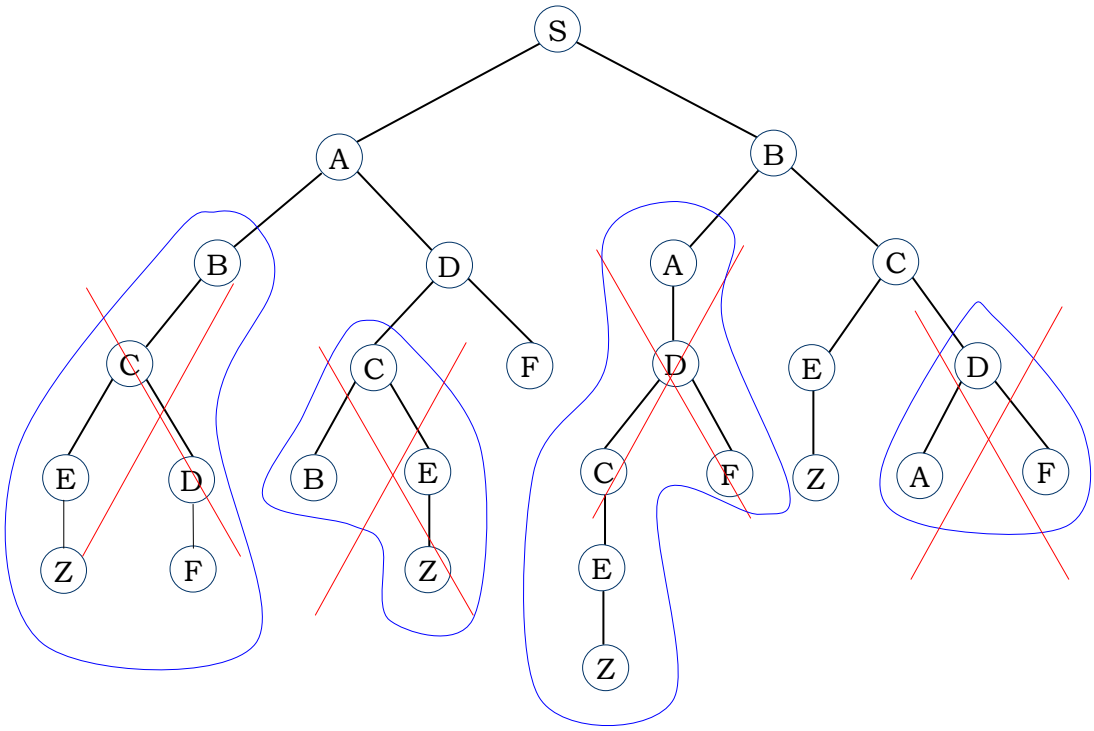
B. Branch and Bound dengan Dynamic Programming

Metode ini sama dengan Branch and Bound, tetapi lebih efisien karena bisa mengurangi lebar / melakukan pemotongan terhadap lebar dari tree. Hal ini dilakukan dengan cara mereduksi lintasan parsial yang menuju ke suatu node yang sudah pernah dikunjungi sebelumnya

- Algoritma
  1. Buat sebuah Antrian, inisialisasi node pertama dengan Root dari tree
  2. Bila lintasan parsial ≠ lintasan GOAL, jika ada lintasan parsial dengan node terakhir yang sama (dalam satu queue) maka diambil yang harganya paling minimal, sedangkan yang lebih mahal dihapus dari queue, sehinggan tree akan lebih kurus.
  3. Bila node pertama = lintasan GOAL, selesai



- Keuntungan
  1. Algoritma berhenti pada nilai optimum sebenarnya.
  2. Lebih efisien dari metode Branch & Bound dalam penggunaan memori dan waktu eksekusi karena ada pemotongan.
- Kelemahan
  1. Harus mengingat node terakhir dari lintasan parsial yang sudah dicapai sebelumnya.



Gambar 4.8 Gambar Tree Yang Dihasilkan dengan Metode Branch & Bound dengan Dynamic Programming

4.5 PENCARIAN UNTUK GAME

Permainan (Game) adalah sesuatu yang sangat menarik dan menjadi sub topik tersendiri di dalam Kecerdasan Buatan. Terdapat beberapa alasan kenapa permainan (game) menjadi menarik, yaitu:

- Kriteria menang atau kalah jelas
- Dapat mempelajari permasalahan
- Alasan histori
- Menyenangkan
- Biasanya mempunyai search space yang besar (misalnya game catur mempunyai 35100 nodes dalam search tree dan 1040 legal states)

Terdapat beberapa ciri umum pada permainan (Game) dalam Kecerdasan Buatan, yaitu:

- Terdapat 2 pemain

- Kesempatan pemain bergantian
- Zero-sum: kerugian seorang pemain adalah keuntungan pemain lain
- Perfect information: pemain mengetahui semua informasi state dari game
- Tidak mengandung probabilistik (seperti dadu)
- Contoh: Tic-Tac-Toe, Checkers, Chess, Go, Nim, Othello
- Game tidak termasuk Bridge, Solitaire, Backgammon, dan semisalnya

Beberapa computer game player telah mencatat kehebatannya, diantaranya adalah yang disebut di bawah ini:

– **Catur**:
  - Deep Blue mengalahkan Gary Kasparov pada tahun 1997
  - Gary Kasparav vs. Deep Junior (Feb 2003): seri

– **Checkers**:
  - Chinook adalah juara dunia

– **Go**:
  - Computer player adalah sangat tangguh

– **Bridge**:
  - Computer players mempunyai “Expert-level”

4.5.1 Bagimana Komputer menjadi Salah Satu Pemain dalam Game

Terdapat beberapa hal yang harus diperhatikan untuk menjadikan komputer sebagai salah satu pemain dalam Game, yaitu:

- Cara bermain game:
  – Pertimbangkan semua kemungkinan jalan
  – Berikan nilai pada semua kemungkinan jalan
  – Jalankan pada kemungkinan yang mempunyai nilai terbaik
  – Tunggu giliran pihak lawan jalan
  – Ulangi cara diatas
- Permasalahan kunci:
  – Representasikan “board” atau “state”
  – Buatlah next board yang legal
  – Lakukan evaluasi pada posisi

4.5.2 Fungsi Evaluasi

Evaluation function atau static evaluator digunakan untuk mengevaluasi nilai posisi yang baik.

Zero-sum assumption membolehkan untuk menggunakan single evaluation function untuk mendeskripsikan nilai posisi

- **f(n) >> 0**: posisi n baik untuk saya dan jelek untuk lawan
- **f(n) << 0**: posisi n jelek untuk saya dan baik untuk lawan
- **f(n) near 0**: posisi n adalah posisi netral/seri
- **f(n) = +infinity**: saya menang
- **f(n) = -infinity**: lawan menang

Beberapa contoh fungsi evaluasi yang digunakan pada pada game:

- Tic-Tac-Toe

  f(n) = [# of 3-lengths open for me] - [# of 3-lengths open for you]

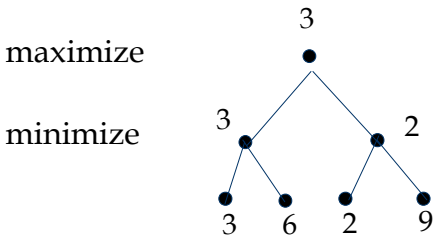
  dimana 3-length adalah complete row, column, atau diagonal yang terisi
- Alan Turing's function untuk catur
  - **f(n) = w(n)/b(n)** dimana w(n) = jumlah point value bidak putih and b(n) = jumlah point value dari bidak hitam
- Deep Blue (yang mengalahkan Gary Kasparov tahun 1997) mempunyai lebih dari 8000 features untuk evaluation function

4.5.3 Sistem Pencarian dalam Game

Ada dua metode mendasar dalam pencarian untuk game ini, yaitu:

A. METODE MINIMAX

John von Neumann pada tahun 1944 menguraikan sebuah algoritma search pada game, dikenal dengan nama Minimax, yang memaksimalkan posisi pemain dan meminimalkan posisi lawan

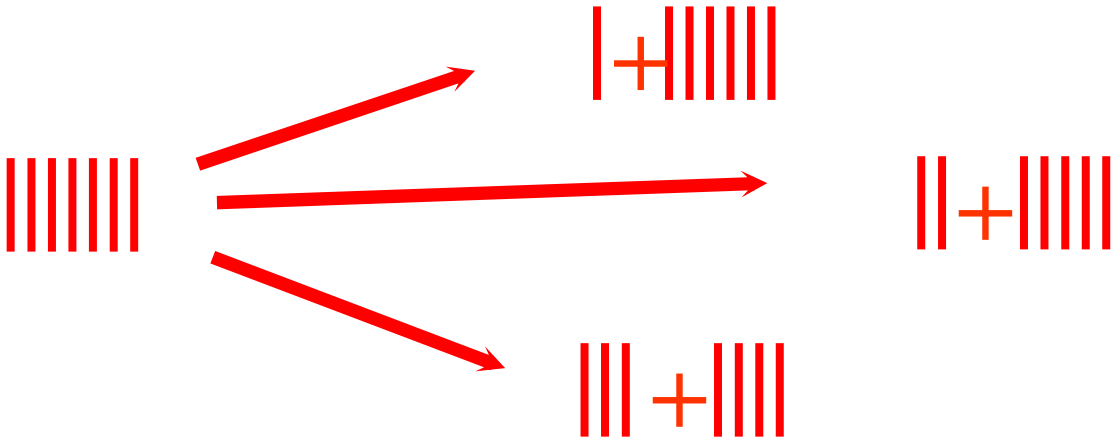


Gambar 4.9 Tree pada Metode Minimax

Metode ini berjalan dengan melakukan proses pada tiap level. Proses yang dilakukan bergantian yaitu mencari nilai minimum, kemudian pada node parent-nya dilakukan pencarian untuk nilai maksimum.

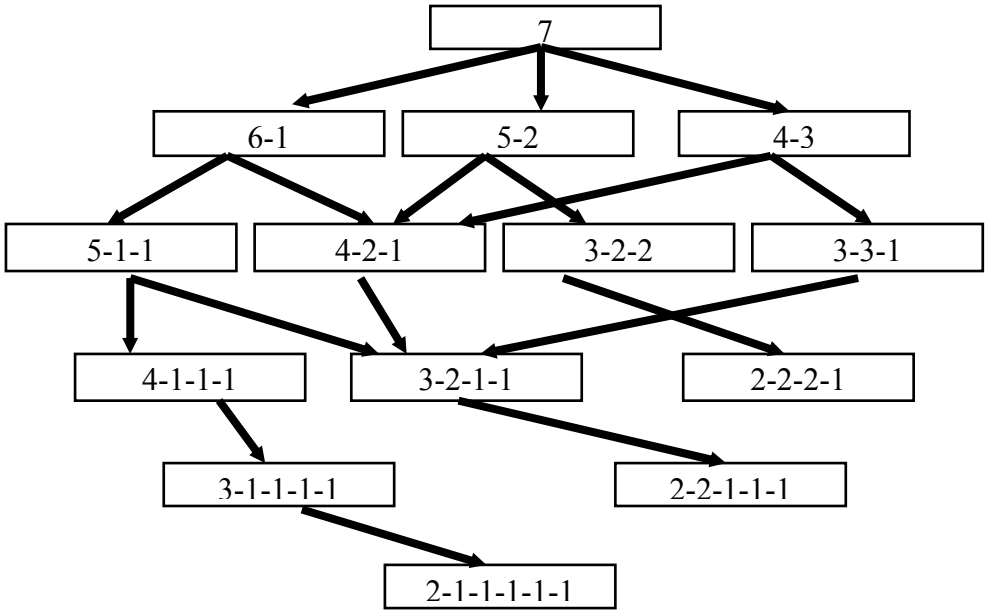
Contoh penggunaan metode ini adalah penerapannya pada Game Nim, dengan aturan sebagai berikut:

- Diawali serangkaian batang
- Setiap pemain harus memecah serangkaian batang menjadi 2 kumpulan dimana jumlah batang di tiap kumpulan tidak boleh sama dan tidak boleh kosong



Gambar 4.10 Ilustrasi Game Nim dengan Jumlah Batang 7

Contoh Game Nim dengan jumlah batang 7 akan menghasilkan semua kemungkinan jalan seperti yang ditunjuk pada Gambar 4.11.



Gambar 4.11

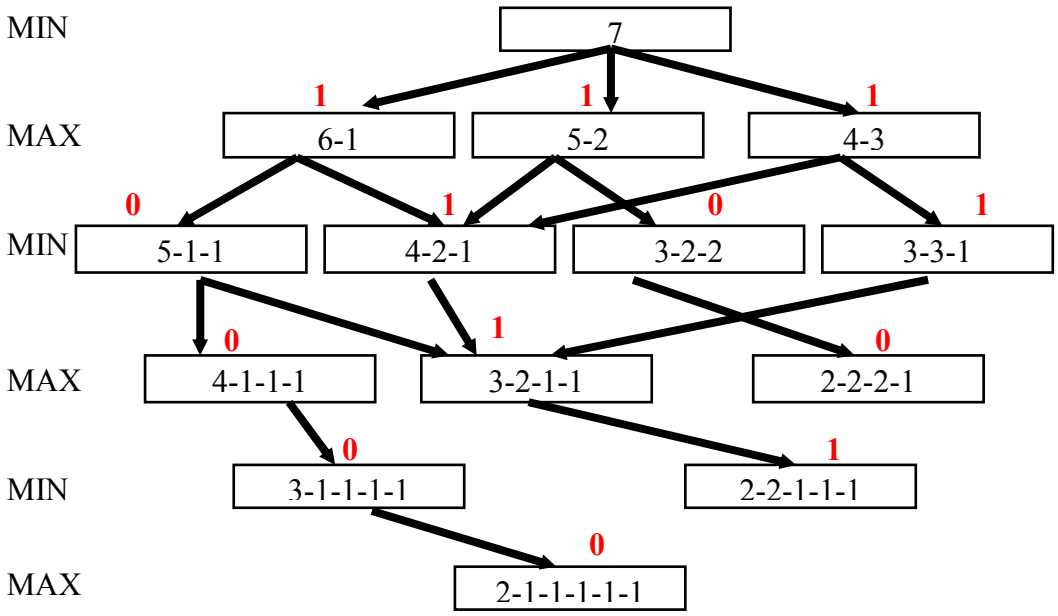
Semua Kemungkinan Jalan pada Game Nim dengan Jumlah Batang 7

Asumsi yang diberikan adalah MIN bermain dulu dan fungsi evaluasi 0 → MIN menang dan 1 → MAX menang. Setelah tree pada Gambar 4.11 didapatkan, sesuai dengan cara bermain pada Game seperti yang disebutkan di atas, maka langkah selanjutnya adalah memberi nilai pada semua kemungkinan. Pemberian nilai ini dimulai dari simpul yang menjadi daun(leaf) dengan nilai kemenangan berdasarkan siapa yang bermain terakhir kali, sehingga tree pada Gambar 4.11 akan mempunyai nilai seperti yang digambarkan pada Gambar 4.12.

Dari Semua Kemungkinan Jalan seperti yang digambarkan pada Gambar 4.12, maka komputer dapat menentukan jalan mana yang ditempuh agar komputer bisa menang dalam Game. Jika pihak lawan memilih kondisi 6-1 pada langkah pertama, maka komputer akan memilih kondisi yang mempunyai nilai 1 yaitu 4-2-1. Jika lawan memilih kondisi 5-2, komputer akan memilih kondisi yang mempunyai nilai 1, yaitu 4-2-1. Jika lawan memilih kondisi 4-3, maka komputer bisa memilih kondisi 4-2-1 atau 3-3-1 karena keduanya bernilai 1. Begitu seterusnya.

Kelemahan dari metode ini adalah waktu eksekusi yang dibutuhkan sebanding dengan jumlah leaf-nya. Sehingga jika #leaf lebih besar, maka permasalahan akan

semakin kombinatorik. Hal ini diperbaiki dengan sebuah metode yang dinamakan Alpha-beta prunning, seperti yang diuraikan pada sub-bab di bawah ini.

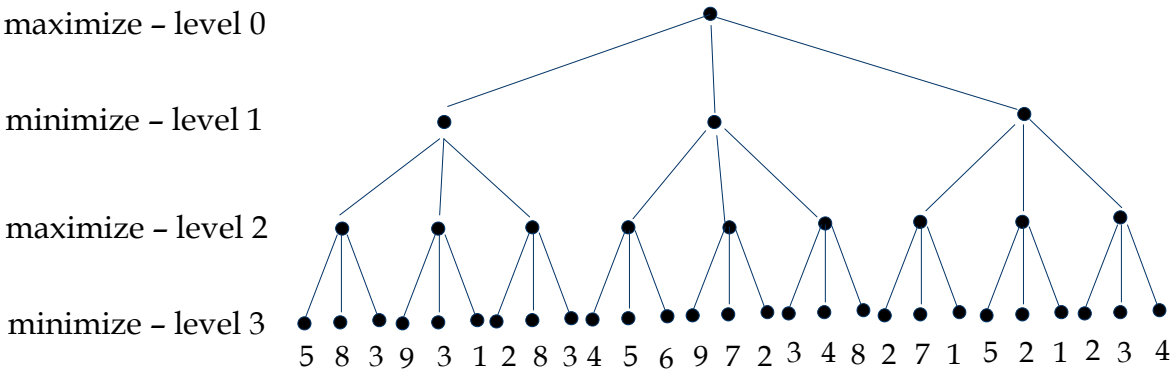


Gambar 4.12

Nilai pada Semua Kemungkinan Jalan pada Game Nim dengan Jumlah Batang 7

B. ALPHA BETA PRUNNING

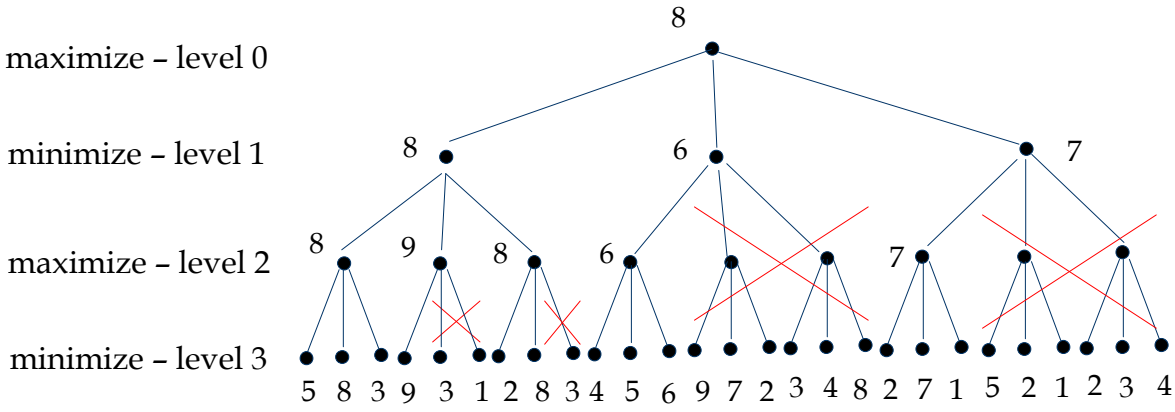
Kondisi awal dari sebuah keadaan seperti digambarkan pada gambar 4.13 akan didapatkan hasil dengan metode Alpha Beta Prunning seperti pada gambar 4.14.



Gambar 4.13 Tree untuk Kondisi Awal Metode Alpha Beta Prunning

Algoritma

1. Hampiri node pertama pada leaf dengan nilai 5, naik ke parent pada level 2 masukkan nilai 5, hampiri 8, karena 8 > 5 maka ganti parent dengan 8, hampiri 3.
2. Setelah ketiga leaf pertama terhampiri, naik lebih tinggi lagi ke level 1 masukkan nilai 8.
3. Hampiri node keempat pada leaf dengan nilai 9, naik ke parent pada level 2 masukkan nilai 9. Jika kita menghampiri leaf berikutnya, kita mencari nilai yang lebih tinggi dari 9, sementara pada level 1 kita mencari yang lebih kecil dari 8, maka leaf 3 dan 1 kita potong (tidak kita hampiri).
4. Begitu seterusnya hingga kita dapatkan hasil akhir dari tree di atas adalah seperti gambar 4.14



Gambar 4.14 Tree untuk Hasil Metode Alpha Beta Prunning

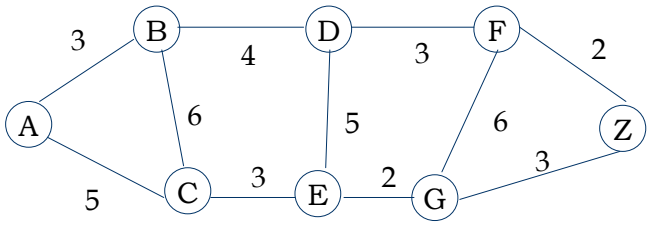
4.6 RINGKASAN

1. Untuk dapat diselesaikan oleh sebuah sistem Kecerdasan Buatan, maka suatu permasalahan harus ditentukan dulu ruang keadaannya.
2. Langkah-langkah dalam menentukan ruang keadaan adalah : Menentukan State Awal, Menentukan Tujuan yang akan Dicapai, Menentukan Aturan-Aturan .
3. Graph dapat digunakan untuk menotasikan ruang keadaan (perpindahan dari state awal ke GOAL).

4. Untuk menghindari proses pencarian yang berulang, maka graph harus digambarkan ke dalam tree.
5. Algoritma Pencarian terbagi menjadi 3 : Algoritma Buta, Algoritma Optimal, dan Algoritma untuk Permainan (Game).

4.7 LATIHAN

1. Buatlah graph ruang keadaan untuk permasalahan petani dan barang bawaannya di atas. Dari sana buatlah tree-nya.
2. Dari tree yang telah anda buat pada no.1 tuliskan perubahan path untuk mencari GOAL jika algoritma yang digunakan Depth First.
3. Dari tree yang telah anda buat pada no.1 tuliskan perubahan path untuk mencari GOAL jika algoritma yang digunakan Breadth First.
4. Buatlah graph ruang keadaan untuk permasalahan teko air di atas. Dari sana buatlah tree-nya.
5. Dari tree yang telah anda buat pada no.4 tuliskan perubahan path untuk mencari GOAL jika algoritma yang digunakan Depth First.
6. Dari tree yang telah anda buat pada no.4 tuliskan perubahan path untuk mencari GOAL jika algoritma yang digunakan Breadth First.
7. a. Suatu proses dapat dinyatakan sebagai serangkaian perubahan dari suatu keadaan (state) ke keadaan lainnya. Tentukan urutan proses dengan biaya minimum yang memerlukan perubahan dari keadaan A ke keadaan Z dengan menggunakan metode Best First (Biaya dinyatakan dengan jarak antara dua node)



b. Lakukan langkah pencarian untuk graph di atas dengan menggunakan metode Branch and Bound with Dynamic Programming.

8. Buatlah semua kemungkinan jalan dalam Game Nim dengan jumlah batang 11. Dari semua kemungkinan jalan tersebut berikan semua nilainya, dan berikan salah satu contoh bagaimana permainan berjalan.

9. Suatu permainan (game) dapat ditelusuri secara optimal dengan menggunakan prosedur "Alpha-beta Prunning". Jelaskan langkah-langkah hampiran untuk pohon di bawah ini: