Penerapan Branch and Bound untuk Menentukan Strategi Optimal pada Satu Ronde Permainan UNO

A. Fawwaz Azam Wicaksono - 13524070
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: fawwazazam09670@gmail.com , 13524070@std.stei.itb.ac.id

***Abstrak*—UNO merupakan permainan kartu berbasis giliran yang memiliki banyak kemungkinan langkah karena adanya pencocokan warna, angka, simbol, serta kartu aksi. Makalah ini membahas penerapan algoritma Branch and Bound untuk menentukan strategi optimal pada satu ronde permainan UNO dalam model yang dibatasi. Permainan dimodelkan sebagai pohon ruang status, dengan state berupa kondisi kartu pemain, kartu lawan, kartu teratas, warna aktif, giliran, dan kedalaman pencarian. Tujuan optimisasi adalah meminimalkan jumlah kartu pemain yang tersisa. Program membandingkan tiga pendekatan, yaitu brute force, Branch and Bound, dan greedy. Hasil pengujian pada lima skenario default menunjukkan bahwa Branch and Bound menghasilkan solusi yang sama dengan brute force, tetapi dengan jumlah node yang lebih sedikit. Pada beberapa skenario, pruning mampu mengurangi lebih dari separuh node yang perlu diperiksa. Dengan demikian, Branch and Bound dapat digunakan sebagai pendekatan efektif untuk pencarian strategi UNO pada ruang pencarian terbatas.**

Kata kunci—Branch and Bound, UNO, strategi permainan, pohon ruang status, pruning

I. PENDAHULUAN

UNO merupakan permainan kartu berbasis giliran yang mengharuskan pemain mencocokkan kartu berdasarkan warna, angka, atau simbol dengan kartu teratas pada tumpukan buangan. Setiap pemain berusaha menghabiskan kartu di tangannya lebih dahulu dibandingkan pemain lain. Selain kartu angka, UNO juga memiliki kartu aksi seperti Skip, Reverse, Draw Two, Wild, dan Wild Draw Four yang dapat mengubah kondisi permainan [1]. Keberadaan kartu aksi tersebut membuat keputusan pemain tidak hanya bergantung pada kartu yang dapat dimainkan saat ini, tetapi juga pada kemungkinan kondisi permainan setelah kartu tersebut dimainkan. Oleh karena itu, meskipun aturan dasar UNO relatif sederhana, permainan ini tetap memiliki unsur strategi yang menarik untuk dianalisis.

Dalam sudut pandang komputasi, pemilihan kartu pada permainan UNO dapat dipandang sebagai persoalan pencarian pada ruang kemungkinan langkah. Setiap kondisi permainan dapat direpresentasikan sebagai sebuah state, sedangkan setiap kartu legal yang dimainkan dapat dianggap sebagai transisi menuju state berikutnya. Penelitian Demaine et al. menunjukkan bahwa UNO memiliki karakter kombinatorial yang tidak sederhana, bahkan pada variasi permainan satu pemain [2]. Hal ini memperlihatkan bahwa urutan kartu yang dimainkan dapat sangat memengaruhi tercapai atau tidaknya kondisi yang diinginkan. Dengan demikian, pemilihan strategi dalam UNO dapat dianalisis menggunakan pendekatan algoritmik, khususnya melalui pembentukan pohon ruang status.

Salah satu pendekatan yang dapat digunakan untuk menelusuri ruang pencarian tersebut adalah algoritma Branch and Bound. Algoritma ini umum digunakan pada persoalan optimisasi karena dapat memilih simpul yang lebih menjanjikan berdasarkan nilai cost atau bound tertentu [4]. Berbeda dengan brute force yang mencoba seluruh kemungkinan langkah, Branch and Bound dapat memangkas cabang yang tidak berpotensi menghasilkan solusi lebih baik. Pada konteks satu ronde permainan UNO, Branch and Bound dapat diterapkan untuk mencari urutan kartu legal yang meminimalkan jumlah kartu pemain yang tersisa. Dengan cara tersebut, persoalan strategi UNO tidak hanya dipandang sebagai pencarian langkah yang sah, tetapi juga sebagai persoalan optimisasi sederhana.

Makalah ini membahas penerapan algoritma Branch and Bound untuk menentukan strategi optimal pada satu ronde permainan UNO dalam model yang dibatasi. Model permainan yang digunakan mencakup kartu pemain, kartu lawan, kartu teratas, warna aktif, giliran pemain, serta batas kedalaman pencarian. Program yang dibuat membandingkan tiga pendekatan, yaitu brute force, Branch and Bound, dan greedy. Brute force digunakan sebagai acuan hasil optimal pada ruang pencarian terbatas, Branch and Bound digunakan untuk mengamati efektivitas pruning, sedangkan greedy digunakan sebagai pembanding strategi sederhana. Melalui perbandingan tersebut, makalah ini bertujuan menunjukkan bahwa Branch and Bound dapat mengurangi jumlah state yang perlu diperiksa sambil tetap mempertahankan kualitas solusi dalam skenario pengujian yang dirancang.

II. DASAR TEORI

A. Permainan UNO

UNO adalah permainan kartu berbasis giliran yang dimainkan dengan mencocokkan kartu dari tangan pemain terhadap kartu teratas pada tumpukan buangan. Suatu kartu dapat dimainkan apabila memiliki warna, angka, atau simbol yang sesuai dengan kartu teratas tersebut [1]. Selain kartu angka, terdapat kartu aksi yang dapat memengaruhi jalannya

permainan, seperti Skip, Reverse, Draw Two, Wild, dan Wild Draw Four. Kartu Wild memiliki peran khusus karena pemain dapat menentukan warna aktif berikutnya setelah kartu tersebut dimainkan. Dengan aturan tersebut, setiap giliran dalam UNO menuntut pemain untuk memilih kartu yang tidak hanya sah secara aturan, tetapi juga dapat memengaruhi kemungkinan langkah berikutnya.

Gambar 1. Contoh kartu pada permainan UNO

(Sumber: Uno Rules [1])

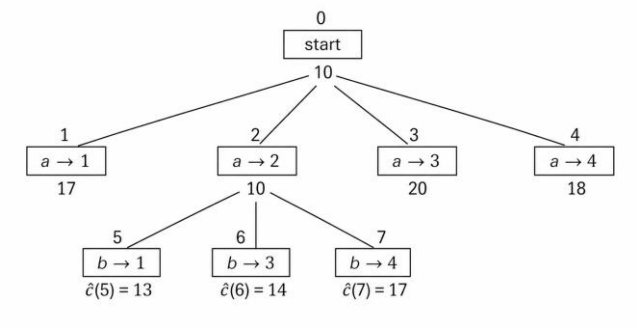
Gambar 1 menunjukkan contoh kartu yang digunakan dalam permainan UNO. Kartu-kartu tersebut memperlihatkan bahwa keputusan pemain tidak hanya ditentukan oleh angka, tetapi juga oleh warna dan jenis aksi yang tersedia. Karena setiap pilihan kartu dapat mengubah kartu teratas, warna aktif, dan giliran berikutnya, UNO dapat dipandang sebagai permainan yang memiliki unsur strategi. Demaine et al. juga menunjukkan bahwa UNO memiliki karakter kombinatorial yang tidak sederhana, bahkan pada variasi permainan yang dibatasi [2].

Permainan UNO juga dapat dikaitkan dengan konsep pohon keputusan karena setiap pilihan kartu menghasilkan kondisi permainan baru. Atmadja membahas penerapan pohon keputusan dalam strategi permainan UNO dan menunjukkan bahwa pilihan kartu dapat direpresentasikan sebagai cabang keputusan [3]. Representasi tersebut berguna untuk memahami bahwa strategi dalam UNO bergantung pada rangkaian keputusan, bukan hanya satu pilihan tunggal. Dalam konteks strategi algoritma, karakter ini membuat UNO dapat dimodelkan sebagai ruang pencarian. Oleh karena itu, UNO menjadi contoh yang sesuai untuk mengamati penerapan algoritma pencarian dan optimisasi.

B. Pohon Ruang Status

Pohon ruang status adalah struktur pohon yang merepresentasikan kemungkinan keadaan yang dapat dicapai dari suatu keadaan awal. Dalam struktur ini, akar menyatakan keadaan awal, simpul menyatakan keadaan antara, dan sisi menyatakan aksi yang menyebabkan perpindahan dari satu keadaan ke keadaan lain. Setiap lintasan dari akar menuju suatu simpul menggambarkan urutan keputusan yang telah diambil. Pohon ruang status banyak digunakan dalam persoalan pencarian karena memungkinkan solusi dibangun secara bertahap. Dengan membentuk pohon tersebut, algoritma dapat menelusuri kemungkinan solusi secara sistematis.

Pada persoalan permainan, pohon ruang status dapat digunakan untuk menggambarkan perkembangan kondisi permainan dari satu langkah ke langkah berikutnya. Sebuah simpul dapat menyatakan konfigurasi permainan, sedangkan cabang dari simpul tersebut menyatakan aksi legal yang dapat dipilih. Jika suatu simpul memiliki banyak cabang, berarti terdapat banyak pilihan yang dapat diambil dari kondisi tersebut. Semakin besar jumlah pilihan pada setiap simpul dan semakin dalam pencarian dilakukan, semakin besar pula ukuran pohon yang terbentuk. Oleh karena itu, persoalan berbasis permainan sering kali memiliki ruang pencarian yang besar meskipun aturan dasarnya terlihat sederhana.



Gambar 2. Contoh pohon ruang status pada algoritma Branch and Bound

(Sumber: PPT Rinaldi Munir [4])

Gambar 2 memperlihatkan contoh pohon ruang status pada algoritma Branch and Bound. Setiap simpul menyatakan keadaan yang dapat diperiksa, sedangkan cabang menyatakan keputusan yang menghasilkan keadaan berikutnya. Struktur ini menjadi dasar bagi beberapa pendekatan pencarian, seperti brute force, greedy, dan Branch and Bound. Perbedaan utama dari pendekatan-pendekatan tersebut terletak pada cara memilih simpul yang diperiksa dan cara menghentikan pencarian pada cabang tertentu.

C. Brute Force dan Greedy

Brute force adalah pendekatan penyelesaian masalah dengan mencoba seluruh kemungkinan solusi dalam ruang pencarian. Pada pendekatan ini, setiap kandidat solusi diperiksa untuk menentukan apakah kandidat tersebut memenuhi tujuan yang dicari. Keunggulan brute force adalah sifatnya yang sederhana dan lengkap, sehingga dapat menemukan solusi terbaik apabila ruang pencarian yang diperiksa mencakup seluruh kemungkinan. Namun, pendekatan ini dapat menjadi tidak efisien ketika jumlah kemungkinan solusi sangat besar. Levitin menjelaskan bahwa exhaustive search sering kali menghadapi pertumbuhan ruang pencarian yang besar pada persoalan kombinatorial [5].

Dalam konteks pohon ruang status, brute force dapat dipandang sebagai penelusuran semua cabang sampai kondisi berhenti tertentu. Jika setiap simpul memiliki rata-rata b cabang dan pencarian dilakukan sampai kedalaman d, jumlah simpul yang mungkin diperiksa dapat bertambah sangat cepat. Pertumbuhan ini membuat brute force cocok digunakan sebagai acuan pada persoalan kecil, tetapi kurang praktis untuk persoalan yang lebih besar. Meskipun demikian, brute force tetap penting dalam analisis karena dapat digunakan sebagai pembanding terhadap algoritma yang lebih selektif. Dengan membandingkan hasil algoritma lain terhadap brute force, kualitas solusi dan efisiensi pencarian dapat dievaluasi.

Greedy adalah pendekatan yang memilih keputusan terbaik secara lokal pada setiap langkah. Strategi ini tidak mencoba seluruh kemungkinan lanjutan, melainkan langsung memilih opsi yang tampak paling menguntungkan berdasarkan kriteria tertentu. Keunggulan greedy terletak pada kesederhanaan dan kecepatannya. Akan tetapi, karena tidak mempertimbangkan dampak keputusan terhadap keseluruhan lintasan, greedy tidak selalu menghasilkan solusi optimal. Pada persoalan permainan, keputusan lokal yang tampak baik dapat menyebabkan keadaan berikutnya menjadi kurang menguntungkan.

D. Branch and Bound

Branch and Bound adalah algoritma yang digunakan untuk menyelesaikan persoalan optimisasi dengan membangun pohon ruang status secara bertahap. Algoritma ini bekerja dengan membangkitkan cabang-cabang solusi, menghitung nilai bound untuk setiap simpul, lalu memilih simpul yang paling menjanjikan untuk diekspansi. Pada persoalan minimisasi, simpul dengan nilai bound terkecil diprioritaskan karena dianggap memiliki peluang menghasilkan solusi dengan biaya paling rendah. Simpul yang telah dibangkitkan tetapi belum diperiksa disebut simpul hidup. Simpul yang dipilih dari kumpulan simpul hidup untuk diperluas disebut simpul ekspansi atau simpul-E [4].

Perbedaan utama Branch and Bound dengan brute force terletak pada penggunaan bound untuk membatasi pencarian. Brute force memeriksa seluruh cabang yang tersedia, sedangkan Branch and Bound dapat menghentikan cabang tertentu sebelum ditelusuri lebih jauh. Penghentian tersebut dilakukan apabila cabang tersebut tidak mungkin menghasilkan solusi yang lebih baik daripada solusi terbaik yang telah ditemukan. Dengan cara ini, Branch and Bound tetap mempertahankan kerangka pencarian sistematis, tetapi tidak harus memeriksa seluruh ruang pencarian. Teknik ini membuat Branch and Bound sesuai untuk persoalan optimisasi yang memiliki banyak kemungkinan solusi.

Levitin menjelaskan bahwa Branch and Bound membutuhkan dua komponen penting, yaitu bound pada setiap simpul dan nilai solusi terbaik yang telah ditemukan sejauh ini [5]. Bound digunakan untuk memperkirakan kualitas terbaik yang masih mungkin dicapai dari suatu simpul. Sementara itu, solusi terbaik sementara digunakan sebagai pembanding untuk menentukan apakah suatu cabang masih layak diperiksa. Jika bound suatu simpul tidak lebih baik daripada solusi terbaik sementara, simpul tersebut dapat dihentikan. Dengan demikian, efektivitas Branch and Bound sangat dipengaruhi oleh kualitas fungsi bound yang digunakan.

E. Bound dan Pruning

Bound adalah nilai batas yang digunakan untuk memperkirakan kualitas solusi terbaik yang masih mungkin diperoleh dari suatu simpul. Pada persoalan minimisasi, bound biasanya berupa batas bawah, yaitu nilai terkecil yang secara teoritis masih dapat dicapai dari simpul tersebut. Pada persoalan maksimisasi, bound dapat berupa batas atas, yaitu nilai terbesar yang masih mungkin dicapai. Bound tidak harus sama dengan nilai solusi akhir, tetapi harus cukup representatif untuk membantu menentukan simpul mana yang lebih menjanjikan. Dengan adanya bound, algoritma dapat membedakan cabang yang masih layak diperiksa dari cabang yang tidak lagi berpotensi menghasilkan solusi terbaik.

Pruning adalah proses pemangkasan cabang pada pohon ruang status. Dalam Branch and Bound, pruning dilakukan ketika bound suatu simpul menunjukkan bahwa cabang tersebut tidak dapat menghasilkan solusi yang lebih baik daripada solusi terbaik yang sudah ditemukan. Pemangkasan juga dapat dilakukan jika suatu simpul melanggar batasan persoalan atau tidak lagi memiliki kemungkinan untuk dikembangkan menjadi solusi yang layak. Dengan pruning, algoritma tidak perlu membangkitkan seluruh keturunan dari simpul tersebut. Hal ini dapat mengurangi jumlah simpul yang diperiksa dan mempercepat proses pencarian.

Konsep bound dan pruning banyak digunakan dalam penerapan Branch and Bound pada persoalan kombinatorial. Pada permainan kartu Tri-Peaks, Branch and Bound digunakan untuk membatasi ruang pencarian agar penyelesaian tidak harus memeriksa seluruh kemungkinan langkah [6]. Prinsip yang sama dapat diterapkan pada persoalan permainan lain yang memiliki banyak pilihan keputusan. Selama persoalan dapat dimodelkan sebagai ruang pencarian dan memiliki fungsi objektif yang jelas, bound dapat dirancang untuk mengarahkan proses pencarian. Dengan demikian, kualitas bound menjadi faktor penting dalam menentukan seberapa efektif pruning dapat dilakukan.

III. Pemodelan dan Perancangan Algoritma

A. Batasan Model Permainan

Permainan UNO yang dibahas pada makalah ini dimodelkan sebagai persoalan pencarian strategi pada satu ronde permainan. Model permainan dibatasi pada dua pemain, yaitu pemain utama dan lawan. Informasi kartu pada tangan pemain dan lawan dianggap diketahui sejak awal agar ruang pencarian dapat dibentuk secara deterministik. Kondisi awal permainan terdiri atas kartu pemain, kartu lawan, kartu teratas pada tumpukan buangan, warna aktif, dan giliran pemain. Dengan batasan tersebut, persoalan dapat difokuskan pada pemilihan urutan kartu legal yang menghasilkan sisa kartu pemain sesedikit mungkin.

Aturan legalitas kartu mengikuti aturan dasar UNO, yaitu kartu dapat dimainkan apabila memiliki warna, angka, atau simbol yang sesuai dengan kartu teratas, sedangkan kartu Wild dapat dimainkan dengan menentukan warna aktif berikutnya [1]. Kartu aksi yang dimodelkan meliputi Skip, Reverse, Draw Two, Wild, dan Wild Draw Four. Pada model dua pemain, efek Skip, Reverse, Draw Two, dan Wild Draw Four dipandang sebagai aksi yang membuat pemain yang sama memperoleh giliran berikutnya. Pengambilan kartu dari deck tidak dimodelkan karena makalah ini berfokus pada pencarian urutan kartu dari kondisi tangan yang telah diketahui. Batasan ini membuat ruang pencarian lebih terkendali dan sesuai untuk membandingkan brute force, Branch and Bound, dan greedy.

Pencarian dibatasi dengan kedalaman maksimum sebesar 4 langkah. Batas kedalaman digunakan karena tujuan makalah ini bukan menyelesaikan seluruh permainan UNO sampai akhir, melainkan menentukan strategi terbaik dalam satu ronde terbatas. Nilai kedalaman tersebut dipilih agar ruang pencarian

masih cukup besar untuk memperlihatkan perbedaan jumlah node yang dikunjungi, tetapi tidak terlalu besar sehingga output pohon pencarian menjadi sulit dibaca. Dengan demikian, hasil program tetap dapat digunakan sebagai bahan analisis pada makalah. Batas kedalaman juga membuat perbandingan antaralgoritma dilakukan pada ruang pencarian yang sama.

B. Representasi State dan Move

Setiap keadaan permainan direpresentasikan sebagai sebuah state. State menyimpan informasi yang diperlukan untuk menentukan kartu legal dan membentuk state berikutnya. Komponen utama state meliputi daftar kartu pemain, daftar kartu lawan, kartu teratas, warna aktif, arah permainan, pemain yang sedang mendapat giliran, dan kedalaman pencarian. Kedalaman pencarian menunjukkan jumlah langkah yang telah diambil dari state awal. Representasi ini memungkinkan setiap simpul pada pohon ruang status memuat kondisi permainan yang lengkap.

Sebuah move merepresentasikan kartu legal yang dimainkan dari suatu state. Untuk kartu biasa, move cukup menyimpan kartu yang dimainkan. Untuk kartu Wild dan Wild Draw Four, move juga menyimpan warna aktif yang dipilih setelah kartu dimainkan. Setelah suatu move diterapkan, kartu tersebut dihapus dari tangan pemain yang sedang berjalan, kartu teratas diperbarui, warna aktif diperbarui, dan giliran berikutnya ditentukan berdasarkan jenis kartu. Dengan cara ini, setiap edge pada pohon ruang status menyatakan satu aksi legal yang mengubah satu state menjadi state lain.

Komponen state yang digunakan pada model ini dapat diringkas sebagai berikut.

TABEL 1. KOMPONEN STATE PERMAINAN

Komponen	***Keterangan***
Hand Pemain	Daftar kartu yang dimiliki pemain utama
Hand Lawan	Daftar kartu yang dimiliki lawan
Top Card	Kartu teratas pada tumpukan buangan
Warna Aktif	Warna yang menjadi acuan legalitas kartu berikutnya
Giliran	Pemain yang berhak memainkan kartu
Depth	Jumlah langkah dari state awal

C. Fungsi Objektif

Tujuan optimisasi pada model ini adalah meminimalkan jumlah kartu pemain yang tersisa setelah pencarian dilakukan. Jika H(n) menyatakan jumlah kartu pemain pada state n, maka nilai objektif yang ingin dicari adalah state dengan nilai H(n) sekecil mungkin. Semakin kecil nilai H(n), semakin baik strategi yang dihasilkan karena semakin banyak kartu pemain yang berhasil dikeluarkan. Apabila terdapat dua lintasan dengan jumlah sisa kartu yang sama, lintasan yang mengeluarkan lebih banyak kartu pemain dapat dipilih sebagai solusi yang lebih informatif. Dengan fungsi objektif ini, kualitas solusi dapat dibandingkan secara langsung antaralgoritma.

Brute force digunakan sebagai pembanding utama karena algoritma tersebut menelusuri seluruh cabang legal sampai batas kedalaman yang ditentukan. Solusi brute force dianggap sebagai acuan optimal pada ruang pencarian terbatas tersebut. Branch and Bound diharapkan menghasilkan solusi dengan nilai objektif yang sama, tetapi dengan jumlah node yang dikunjungi lebih sedikit. Greedy digunakan sebagai pembanding strategi sederhana yang memilih satu langkah terbaik secara lokal pada setiap giliran. Perbandingan ketiganya menunjukkan perbedaan antara pencarian menyeluruh, pencarian dengan pruning, dan pencarian heuristik.

Pada pendekatan greedy, pemilihan kartu dilakukan berdasarkan prioritas lokal. Kartu aksi dengan dampak lebih besar diprioritaskan terlebih dahulu, yaitu Wild Draw Four, Draw Two, Skip atau Reverse, Wild, kemudian kartu angka. Jika terdapat beberapa kartu dengan prioritas yang sama, dipilih kartu yang menghasilkan warna aktif dengan jumlah kartu terbanyak pada tangan pemain yang sedang berjalan. Kriteria ini digunakan agar pemain memiliki peluang lebih besar untuk memainkan kartu pada giliran berikutnya. Apabila masih terdapat lebih dari satu pilihan, kartu dipilih berdasarkan urutan kode kartu agar hasil pemilihan tetap deterministik.

D. Fungsi Bound pada UNO

Fungsi bound pada model ini dirancang sebagai batas bawah jumlah kartu pemain yang masih mungkin tersisa dari suatu state. Misalkan H menyatakan jumlah kartu pemain pada state tertentu dan r menyatakan sisa kedalaman yang masih dapat ditempuh. Dalam satu langkah, pemain paling banyak dapat mengeluarkan satu kartu dari tangannya. Oleh karena itu, jumlah kartu pemain paling kecil yang masih mungkin dicapai dari state tersebut tidak dapat lebih kecil daripada H - r. Karena jumlah kartu tidak mungkin bernilai negatif, fungsi bound didefinisikan sebagai berikut.

$$c(n) = \max(0, H(n) - r(n)) \quad (1)$$

Pada persamaan tersebut, c(n) menyatakan nilai bound pada state n. Nilai H(n) adalah jumlah kartu pemain pada state n, sedangkan r(n) adalah sisa kedalaman pencarian dari state tersebut. Jika c(n) bernilai kecil, state tersebut dianggap lebih menjanjikan karena masih memiliki peluang menghasilkan sisa kartu yang lebih sedikit. Sebaliknya, apabila nilai c(n) sudah tidak lebih baik daripada solusi terbaik sementara, cabang tersebut tidak perlu diperiksa lebih lanjut. Dengan demikian, fungsi bound digunakan untuk menentukan prioritas ekspansi dan dasar pruning.

Branch and Bound pada model ini menggunakan pendekatan best-first search terhadap simpul hidup. Setiap simpul hidup disimpan bersama nilai cost atau bound-nya. Pada setiap iterasi, simpul dengan nilai c(n) terkecil dipilih sebagai simpul ekspansi. Setelah simpul tersebut diekspansi, seluruh move legal dari state tersebut dibangkitkan sebagai simpul anak. Apabila suatu simpul memiliki nilai c(n) lebih besar atau sama dengan solusi terbaik sementara, simpul tersebut dipangkas karena tidak dapat menghasilkan solusi yang lebih baik.

E. *Alur Penyelesaian*

Alur penyelesaian dimulai dari pembentukan state awal berdasarkan skenario yang diberikan. Program kemudian membangkitkan seluruh move legal dari state tersebut. Pada brute force, setiap move legal akan ditelusuri sampai mencapai batas kedalaman, kondisi terminal, atau tidak ada move legal yang dapat dimainkan. Pada Branch and Bound, move legal juga dibangkitkan, tetapi simpul yang akan diperiksa dipilih berdasarkan nilai bound terkecil. Pada greedy, hanya satu move dipilih pada setiap giliran berdasarkan prioritas kartu aksi dan kecocokan warna.

Secara umum, langkah Branch and Bound pada model ini adalah sebagai berikut.

1. Bentuk simpul akar dari state awal.
2. Hitung nilai c(n) untuk simpul akar.
3. Masukkan simpul akar ke daftar simpul hidup.
4. Pilih simpul hidup dengan nilai c(n) terkecil sebagai simpul ekspansi.
5. Jika simpul mencapai batas kedalaman atau kondisi terminal, perbarui solusi terbaik.
6. Jika nilai c(n) tidak lebih baik daripada solusi terbaik sementara, lakukan pruning.
7. Jika simpul masih layak diperiksa, bangkitkan seluruh move legal sebagai simpul anak.
8. Ulangi proses sampai tidak ada simpul hidup yang tersisa.

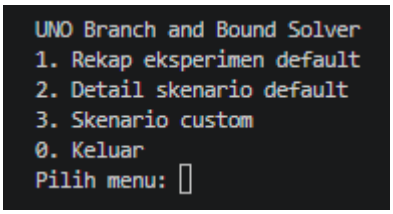
IV. Implementasi dan Pengujian

A. *Implementasi Program*

Program yang dibuat berupa aplikasi berbasis command line interface dengan bahasa Java. Program ini menerima skenario permainan UNO, membentuk state awal, lalu menjalankan tiga pendekatan pencarian, yaitu brute force, Branch and Bound, dan greedy. Setiap algoritma menerima masukan yang sama berupa kartu pemain, kartu lawan, kartu teratas, warna aktif, dan batas kedalaman pencarian. Hasil dari setiap algoritma kemudian ditampilkan dalam bentuk rute solusi, jumlah kartu tersisa, jumlah node yang dikunjungi, jumlah node yang dipangkas, persentase pruning, dan waktu eksekusi. Dengan rancangan tersebut, program dapat digunakan untuk membandingkan kualitas solusi dan efisiensi pencarian pada skenario yang sama.

Struktur program dibagi menjadi beberapa bagian utama. Bagian model digunakan untuk merepresentasikan kartu, pemain, langkah, state permainan, dan skenario. Bagian engine bertanggung jawab untuk memeriksa legalitas kartu, membangkitkan langkah legal, dan menghasilkan state baru setelah suatu kartu dimainkan. Bagian solver berisi implementasi brute force, Branch and Bound, dan greedy. Bagian experiment dan report digunakan untuk menjalankan skenario pengujian serta menampilkan hasil program dalam bentuk tabel dan pohon pencarian.

Program menyediakan tiga menu utama, yaitu rekap eksperimen default, detail skenario default, dan skenario custom. Menu rekap eksperimen default digunakan untuk menjalankan seluruh skenario pengujian yang telah disiapkan. Menu detail skenario default digunakan untuk melihat hasil satu skenario secara lebih rinci, termasuk pohon pencarian dari masing-masing algoritma. Menu skenario custom digunakan untuk memasukkan kartu secara manual sehingga pengguna dapat mencoba kondisi permainan lain. Tampilan awal program dapat dilihat pada Gambar 3.



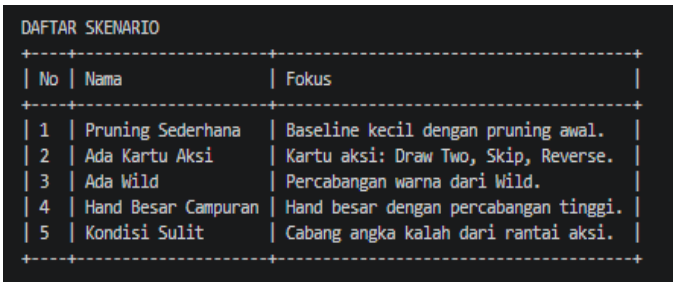
Gambar 3. Tampilan menu utama program

(Sumber: Dokumen Penulis)

B. *Skenario Pengujian*

Pengujian dilakukan menggunakan lima skenario default. Setiap skenario dirancang untuk merepresentasikan kondisi permainan yang berbeda. Skenario pertama digunakan untuk memperlihatkan pruning sederhana pada ruang pencarian kecil. Skenario kedua menekankan penggunaan kartu aksi seperti Skip, Reverse, dan Draw Two. Skenario ketiga menambahkan kartu Wild sehingga terdapat percabangan tambahan berupa pemilihan warna aktif.

Skenario keempat menggunakan jumlah kartu yang lebih besar dan komposisi kartu yang lebih beragam. Skenario ini digunakan untuk mengamati pengaruh Branch and Bound ketika jumlah kemungkinan langkah meningkat. Skenario kelima dibuat sebagai kondisi sulit, yaitu ketika pilihan awal tidak selalu langsung menghasilkan lintasan terbaik. Seluruh skenario default menggunakan batas kedalaman pencarian yang sama, yaitu D = 4. Batas kedalaman tersebut digunakan agar perbandingan antaralgoritma dilakukan pada ruang pencarian yang setara.



Gambar 4. Daftar Skenario Default Program

(Sumber: Dokumen Penulis)

Metrik utama yang digunakan dalam pengujian adalah jumlah sisa kartu pemain dan jumlah node yang dikunjungi. Jumlah sisa kartu pemain menunjukkan kualitas solusi yang dihasilkan oleh algoritma. Jumlah node yang dikunjungi menunjukkan banyaknya state yang benar-benar diperiksa selama pencarian. Pada Branch and Bound, metrik tambahan berupa node yang dipangkas dan persentase pruning juga

dicatat. Persentase pruning dihitung dari perbandingan antara node yang dipangkas dan total node pada ruang pencarian yang sepadan.

C. Hasil Pengujian

Hasil pengujian menunjukkan bahwa Branch and Bound menghasilkan jumlah sisa kartu yang sama dengan brute force pada seluruh skenario default. Hal ini menunjukkan bahwa Branch and Bound tetap mempertahankan kualitas solusi pada ruang pencarian yang dibatasi. Perbedaan utama terlihat pada jumlah node yang dikunjungi. Pada setiap skenario, Branch and Bound mengunjungi node lebih sedikit dibandingkan brute force karena sebagian cabang dipangkas berdasarkan nilai bound. Hasil rekap eksperimen dapat dilihat pada Gambar 5

REKAP HASIL EKSPERIMEN

Skenario	Depth	Algoritma	Sisa	Node	Dipangkas	Pruning	Waktu(ms)
1	4	Brute Force	2	33	0	-	209.1003
1	4	Branch and Bound	2	14	19	57.58%	103.2989
1	4	Greedy	2	-	-	-	16.6537
2	4	Brute Force	1	71	0	-	7.9811
2	4	Branch and Bound	1	16	55	77.46%	4.8990
2	4	Greedy	1	-	-	-	1.2989
3	4	Brute Force	0	43	0	-	4.1274
3	4	Branch and Bound	0	12	31	72.09%	3.3369
3	4	Greedy	0	-	-	-	0.4515
4	4	Brute Force	7	106	0	-	19.4225
4	4	Branch and Bound	7	24	82	77.36%	11.9472
4	4	Greedy	8	-	-	-	1.6587
5	4	Brute Force	2	47	0	-	6.9287
5	4	Branch and Bound	2	16	31	65.96%	3.6426
5	4	Greedy	2	-	-	-	0.8797

Gambar 5. Rekap Hasil Eksperimen Skenario Default

(Sumber: Dokumen Penulis)

Gambar 5 memperlihatkan bahwa Branch and Bound selalu menghasilkan nilai sisa kartu yang sama dengan brute force. Pada skenario 1, brute force mengunjungi 33 node, sedangkan Branch and Bound hanya mengunjungi 14 node. Pada skenario 4, selisih jumlah node menjadi lebih besar karena brute force mengunjungi 106 node, sedangkan Branch and Bound hanya mengunjungi 24 node. Hal ini menunjukkan bahwa pruning menjadi lebih terasa ketika ruang pencarian memiliki percabangan yang lebih besar. Meskipun waktu eksekusi dapat dipengaruhi oleh kondisi perangkat dan overhead program, jumlah node tetap menjadi metrik utama untuk menilai efisiensi pencarian.

Program juga menampilkan pohon pencarian untuk skenario tertentu. Pohon brute force memperlihatkan seluruh cabang legal yang ditelusuri sampai batas kedalaman atau kondisi berhenti. Pohon Branch and Bound menampilkan cabang yang benar-benar diekspansi setelah pemilihan berdasarkan nilai bound. Cabang yang tidak menjanjikan tidak diteruskan karena nilai bound-nya tidak lebih baik daripada solusi terbaik sementara. Contoh tampilan pohon pencarian dapat dilihat pada Gambar 6 dan Gambar 7.

```
BRUTE FORCE

Solusi
  PLAYER R_SKIP - PLAYER R_D2 - PLAYER R7

Pohon pencarian
  ROOT
  ├── PLAYER R_SKIP
  │   ├── PLAYER R_D2
  │   │   └── PLAYER R7
  │   │       ├── OPPONENT R1    [LIMIT]
  │   │       └── OPPONENT G7    [LIMIT]
  │   └── PLAYER R7
  │       ├── OPPONENT R1
  │       │   └── PLAYER R_D2    [LIMIT]
  │       └── OPPONENT G7
  │           ├── PLAYER G7      [LIMIT]
  │           └── PLAYER B7      [LIMIT]
  ├── PLAYER R_D2
  │   ├── PLAYER R_SKIP
  │   │   └── PLAYER R7
  │   │       ├── OPPONENT R1    [LIMIT]
  │   │       └── OPPONENT G7    [LIMIT]
  │   └── PLAYER R7
  │       ├── OPPONENT R1
  │       │   └── PLAYER R_SKIP  [LIMIT]
  │       └── OPPONENT G7
  │           ├── PLAYER G7      [LIMIT]
  │           └── PLAYER B7      [LIMIT]
  └── PLAYER R7
      ├── OPPONENT R1
      │   ├── PLAYER R_SKIP
      │   │   └── PLAYER R_D2    [LIMIT]
      │   └── PLAYER R_D2
      │       └── PLAYER R_SKIP  [LIMIT]
      └── OPPONENT G7
          ├── PLAYER G7          [STOP]
          └── PLAYER B7
              └── OPPONENT B2    [LIMIT]
```

Gambar 6. Pohon Pencarian Algoritma Brute Force Skenario 1

(Sumber: Dokumen Penulis)

```
BRANCH AND BOUND

Solusi
  PLAYER R_D2 - PLAYER R_SKIP - PLAYER R7

Pohon pencarian
  ROOT                          c=1  [EXPAND]
  ├── PLAYER R_D2               c=1  [EXPAND]
  │   ├── PLAYER R_SKIP         c=1  [EXPAND]
  │   │   └── PLAYER R7         c=1  [EXPAND]
  │   │       ├── OPPONENT G7   c=2  [LIMIT]
  │   │       └── OPPONENT R1   c=2  [LIMIT]
  │   └── PLAYER R7             c=1  [EXPAND]
  ├── PLAYER R_SKIP             c=1  [EXPAND]
  │   ├── PLAYER R_D2           c=1  [EXPAND]
  │   │   └── PLAYER R7         c=1  [EXPAND]
  │   │       ├── OPPONENT G7   c=2  [LIMIT]
  │   │       └── OPPONENT R1   c=2  [LIMIT]
  │   └── PLAYER R7             c=1  [EXPAND]
  └── PLAYER R7                 c=1  [EXPAND]
```

Gambar 7. Pohon Pencarian Algoritma Branch and Bound Skenario 1

(Sumber: Dokumen Penulis)

D. Analisis Hasil

Berdasarkan hasil pengujian, brute force dan Branch and Bound menghasilkan jumlah sisa kartu yang sama pada seluruh skenario default. Kesamaan hasil tersebut menunjukkan bahwa Branch and Bound tetap mampu menemukan lintasan terbaik dalam ruang pencarian yang dibatasi. Perbedaan utama kedua algoritma terletak pada cara menelusuri pohon ruang status. Brute force memeriksa seluruh cabang legal sampai batas kedalaman, sedangkan Branch and Bound memilih simpul hidup berdasarkan nilai bound dan menghentikan cabang yang tidak lagi berpotensi memperbaiki solusi. Dengan demikian, Branch

and Bound dapat mempertahankan kualitas solusi sambil mengurangi jumlah node yang perlu diperiksa.

Efektivitas pruning terlihat dari jumlah node yang berhasil dipangkas oleh Branch and Bound. Pada skenario 1, Branch and Bound memangkas 19 dari 33 node, sedangkan pada skenario 2 dan 4 persentase pruning masing-masing mencapai 77,46% dan 77,36%. Hasil tersebut menunjukkan bahwa Branch and Bound cenderung lebih efektif pada skenario yang memiliki percabangan lebih besar. Semakin banyak kemungkinan langkah yang tersedia, semakin besar pula peluang algoritma untuk memangkas cabang yang tidak menjanjikan.

Greedy memiliki waktu eksekusi paling kecil karena hanya memilih satu langkah pada setiap giliran. Namun, pendekatan ini tidak selalu menghasilkan solusi dengan kualitas yang sama seperti brute force dan Branch and Bound. Pada skenario 4, greedy menyisakan 8 kartu, sedangkan brute force dan Branch and Bound menyisakan 7 kartu. Perbedaan tersebut menunjukkan bahwa keputusan terbaik pada satu giliran belum tentu menghasilkan kondisi terbaik setelah beberapa langkah berikutnya.

Hasil pengujian menunjukkan bahwa Branch and Bound sesuai digunakan untuk mencari strategi pada model satu ronde UNO yang dibatasi. Algoritma ini menghasilkan solusi yang sama dengan brute force, tetapi dengan jumlah node yang lebih sedikit. Namun, hasil tersebut tetap berlaku dalam batasan model yang digunakan, yaitu permainan dua pemain, informasi kartu diketahui, tidak ada pengambilan kartu dari deck, dan pencarian dilakukan sampai depth 4.

V. KESIMPULAN

Berdasarkan implementasi dan pengujian yang dilakukan, algoritma Branch and Bound dapat diterapkan untuk menentukan strategi pada satu ronde permainan UNO dalam model yang dibatasi. Pada seluruh skenario default, Branch and Bound menghasilkan jumlah sisa kartu yang sama dengan brute force, tetapi dengan jumlah node yang lebih sedikit karena adanya pruning. Hasil tersebut menunjukkan bahwa Branch and Bound dapat mengurangi proses pencarian tanpa menurunkan kualitas solusi pada ruang pencarian yang digunakan. Sementara itu, greedy memiliki waktu eksekusi lebih kecil, tetapi tidak selalu menghasilkan solusi sebaik brute force dan Branch and Bound. Untuk pengembangan berikutnya, model permainan dapat diperluas dengan menambahkan mekanisme draw pile, jumlah pemain yang lebih banyak, serta evaluasi pada beberapa ronde permainan.

LINK REPOSITORY GITHUB

https://github.com/fawwazazam/Makalah-IF2211-Strategi-Algoritma

UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan makalah yang berjudul "Penerapan Branch and Bound untuk Menentukan Strategi Optimal pada Satu Ronde Permainan UNO" dengan baik. Penulis juga mengucapkan terima kasih kepada pihak-pihak yang telah mendukung dalam penulisan makalah ini, yaitu:

1. Bapak Dr. Ir. Rinaldi Munir, M.T., dan Bapak Dr. Eng. Ir. Rila Mandala, M.Eng selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma atas ilmu, pengajaran, dan arahan yang telah diberikan selama perkuliahan.
2. Orang tua penulis yang senantiasa memberikan doa, dukungan, dan semangat selama proses penyusunan makalah ini.
3. Teman-teman penulis yang telah memberikan bantuan, masukan, dan dukungan dalam menyelesaikan makalah ini.

Akhir kata, penulis berharap makalah ini dapat memberikan wawasan mengenai penerapan algoritma Branch and Bound pada persoalan strategi permainan kartu.

REFERENCES

[1] Uno Rules, "Original Uno Rules." [Online]. Available: https://www.unorules.com/. Accessed: Jun. 19, 2026.

[2] E. D. Demaine, M. L. Demaine, R. Uehara, T. Uno, and Y. Uno, "UNO is hard, even for a single player," 2010.

[3] V. P. Atmadja, "Aplikasi Pohon Keputusan dalam Strategi Permainan Kartu Uno," Makalah IF2120 Matematika Diskrit, Institut Teknologi Bandung, 2021.

[4] R. Munir, N. U. Maulidevi, and M. L. Khodra, "Algoritma Branch & Bound," Bahan Kuliah IF2211 Strategi Algoritma, Institut Teknologi Bandung, 2026.

[5] Levitin, Introduction to the Design and Analysis of Algorithms, 3rd ed. Boston, MA, USA: Pearson, 2012.

[6] D. Pintara, "Strategi Penyelesaian Permainan Kartu Tri-Peaks dengan Pendekatan Branch and Bound," Makalah IF2211 Strategi Algoritma, Institut Teknologi Bandung, 2017.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

A. Fawwaz Azam Wicaksono - 13524070