Perbandingan Algoritma Runut-balik (*Backtracking*) dengan Algoritma Brute Force dalam Penyelesaian Permainan Kanoodle

Diana Tri Handayani – 13522104
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522104@std.stei.itb.ac.id

***Abstract*—Penelitian ini membahas perbandingan antara algoritma runut-balik (*backtracking*) dan algoritma brute force dalam penyelesaian permainan Kanoodle. Kanoodle adalah permainan teka-teki yang menantang pemain untuk menyusun berbagai bentuk dari serangkaian potongan puzzle ke dalam sebuah grid. Algoritma brute force memecahkan teka-teki dengan mencoba setiap kemungkinan kombinasi hingga menemukan solusi yang benar. Sementara itu, algoritma runut-balik menggunakan pendekatan yang lebih cerdas dengan mengeliminasi sebagian besar kombinasi yang tidak mungkin lebih awal, sehingga meningkatkan efisiensi. Penelitian ini akan mengevaluasi kedua algoritma berdasarkan beberapa parameter, seperti waktu komputasi dan jumlah langkah yang dibutuhkan. Hasil penelitian menunjukkan bahwa algoritma runut-balik lebih efisien dalam hal waktu dan langkah yang dibutuhkan dibandingkan dengan algoritma brute force. Temuan ini dapat menjadi acuan dalam memilih algoritma yang lebih tepat untuk menyelesaikan permainan Kanoodle dan teka-teki serupa lainnya.**

Keywords—kanoodle; puzzle; algoritma; backtracking; brute force.

I. PENDAHULUAN

Dalam dunia komputasi, pemecahan masalah merupakan salah satu aspek penting yang telah menjadi fokus utama pengembangan berbagai algoritma. Dalam konteks permainan teka-teki, seperti permainan Kanoodle, penggunaan algoritma untuk menemukan solusi yang tepat menjadi tantangan yang menarik. Kanoodle adalah permainan teka-teki di mana pemain harus menempatkan sejumlah bentuk geometris yang berbeda ke dalam papan berbentuk persegi panjang dengan sel-sel tertentu.

Pada dasarnya, pemecahan permainan Kanoodle melibatkan pencarian solusi yang memenuhi aturan tertentu, seperti posisi dan orientasi setiap bentuk yang harus sesuai dengan ruang yang tersedia di papan permainan. Dalam hal ini, pendekatan untuk permasalahan kombinasi yang umum digunakan adalah algoritma runut-balik (*backtracking*) dengan algoritma brute force sebagai pembandingnya.

Dalam makalah ini, akan dibandingkan penyelesaian berdasarkan pendekatan algoritma runut-balik dan algoritma brute force. Faktor-faktor yang akan diperbandingkan antara lain kecepatan eksekusi dan kompleksitas dari kedua pendekatan tersebut. Dengan pemahaman yang lebih dalam tentang kelebihan dan kelemahan masing-masing pendekatan, pembaca diharapkan dapat memperoleh wawasan yang lebih baik tentang strategi yang optimal dalam menyelesaikan permainan Kanoodle dan penerapannya dalam pemecahan masalah serupa lainnya.

II. DASAR TEORI

A. Algoritma Runut-balik (Backtracking)

Algoritma runut-balik (*backtracking*) merupakan salah satu teknik pencarian yang digunakan untuk menemukan solusi dari masalah komputasional, khususnya dalam masalah yang melibatkan pemilihan dari sejumlah pilihan yang tersedia. Algoritma ini telah dianggap sebagai sebuah metode pemecahan masalah yang sangkil.Algoritma runut-balik pertama kali diperkenalkan oleh D. H. Lehmer tahun 1950. Kemudian R.J Walker, Golomb, dan Baumert menyajikan uraian umum tentang algoritma runut-balik.

Pada algoritma *backtracking* hanya pilihan yang mengarah ke solusi yang dieksplorasi, pilihan yang tidak mengarah ke solusi tidak dipertimbangkan lagi. Hal tersebut dilakukan dengan memangkas (*pruning*) simpul-simpul yang tidak mengarah ke solusi. Berikut adalah properti umum dari algoritma runut-balik:

1. Solusi persoalan

   Solusi persoalan dapat dinyatakan sebagai vektor dengan n-tuple: X = (x1, x2, …, xn) dengan xi ∈ Si. Pada umumnya, S1 = S2 = … = Sn.

2. Fungsi pembangkit nilai xk

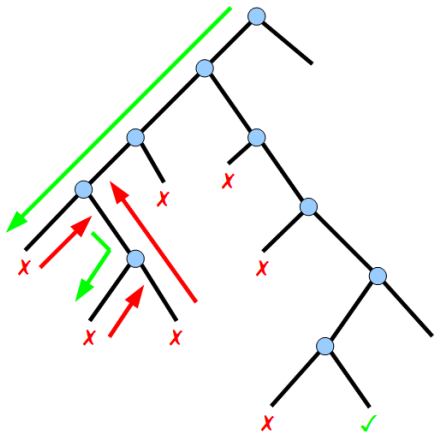
   Fungsi pembangkit ini dapat dinyatakan dengan predikat T() dengan T(x1, x2, …, xk-1) membangkitkan nilai untuk xk, yang merupakan komponen vektor solusi

3. Fungsi pembatas (bounding function)

   Properti ini dinyatakan sebagai predikat B(x1, x2, …, xk) dengan B akan bernilai true apabila (x1, x2, …, xk) mengarah ke solusi permasalahan. Jika nilai dari B bernilai false, maka cabang rekursif tersebut akan dibuang dan akan runut balik (backtrack).

Terdapat beberapa prinsip dalam pencarian solusi menggunakan algoritma backtracking yaitu:

1. Solusi dicari dengan membuat simpul-simpul status yang akan menghasilkan lintasan hingga ke daun.
2. Aturan pembangkitan simpul akan mengikuti aturan DFS (*depth first search*).
3. Simpul-simpul yang sudah dibangkitkan dapat disebut sebagai simpul hidup (*live node*).
4. Simpul yang sedang diperluas dapat disebut sebagai simpul-E (*expand node*).
5. Setiap simpul-E diperluas, lintasan yang dibangun akan semakin panjang.
6. Jika lintasan yang dibentuk tidak mengarah ke solusi, maka simpul-E yang bersangkutan dapat “dimatikan” sehingga menjadi simpul mati (*dead node*).
7. Jika pembentukan lintasan berakhir pada simpul mati, maka proses dari pencarian solusi akan *backtrack* ke simpul parent-nya.
8. Pencarian akan dihentikan apabila telah ditemukan goal node (apabila hanya satu solusi yang dicari) atau hingga tidak ada lagi simpul-E yang masih hidup (apabila dicari semua solusi yang mungkin).



Gambar 1. Ilustrasi Algoritma Runut-balik

(Sumber: https://www.w3.org/2011/Talks/01-14-steven-phenotype/)

B. *Algoritma Brute Force*

Algoritma brute force adalah pendekatan pemecahan masalah yang mendasar di bidang ilmu komputer dan matematika, yang melibatkan eksplorasi semua kemungkinan solusi hingga menemukan solusi yang benar. Pendekatan ini sering disebut juga sebagai "metode coba-coba" (trial and error). Pendekatan ini tidak memperhitungkan efisiensi dan biasanya digunakan sebagai langkah terakhir atau untuk membandingkan dengan algoritma lainnya.

Secara umum algoritma brute force melalui langkah-langkah berikut:

- Enumerasi: daftar semua kemungkinan solusi yang valid untuk masalah yang diberikan.
- Evaluasi: cek setiap solusi untuk melihat apakah memenuhi kondisi masalah.
- Pemilihan: pilih solusi yang memenuhi kondisi atau memiliki nilai terbaik (misalnya, solusi optimal dalam masalah optimasi).

Algoritma brute force juga sering dikenal sebagai *exhaustive search*, atau pencarian secara menyeluruh. Ini merujuk pada pendekatan pencarian di mana semua kemungkinan solusi dieksplorasi untuk menemukan yang terbaik. Namun, secara heuristik, exhaustive search dapat dimodifikasi dengan menggunakan teknik-teknik cerdas atau aturan-aturan berdasarkan intuisi atau pengalaman untuk mengurangi jumlah solusi yang dieksplorasi.

Misalnya, dalam kasus pencarian jalur terpendek dalam graf, alih-alih mencoba semua kemungkinan jalur, pendekatan heuristik dapat mengurangi jumlah jalur yang diperiksa dengan menggunakan aturan-aturan seperti "hanya mengeksplorasi jalur yang memiliki panjang kurang dari batas tertentu" atau "prioritaskan jalur-jalur yang memiliki kemungkinan kecil untuk menjadi jalur terpendek berdasarkan pengetahuan sebelumnya".

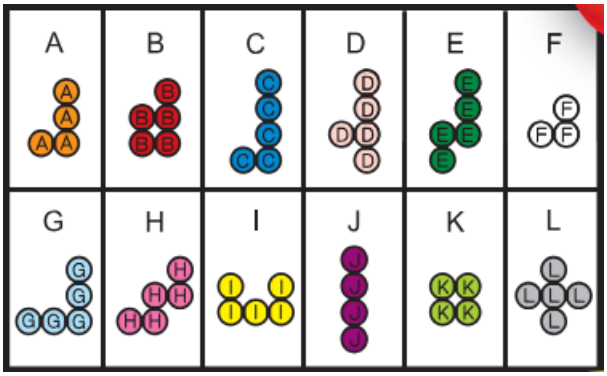
Dengan memanfaatkan pengetahuan atau aturan-aturan semacam ini, *exhaustive search* bisa menjadi lebih efisien tanpa kehilangan jaminan solusi optimal. Meskipun demikian, perlu diingat bahwa pada dasarnya exhaustive search tetap mencoba semua kemungkinan solusi yang memungkinkan, meskipun dengan strategi yang lebih cerdas.

C. *Permainan Kanoodle*

Kanoodle adalah sebuah permainan teka-teki yang melibatkan menyusun bentuk geometris menggunakan potongan-potongan kecil yang memiliki berbagai bentuk dan ukuran. Tujuan dalam Kanoodle adalah menyusun potongan-potongan tersebut sehingga sesuai dengan pola yang diberikan. Pada kanoodle original, papan berukuran 5x11 dan potongan geometris berjumlah 12 dengan bentuk berbeda beda.

Gambar 2. Bentuk Fisik Permainan Kanoodle

(Sumber: https://www.educationalinsights.com/shop/collections/kanoodle)



Gambar 3. Variasi Potongan Geometris
(Sumber: https://www.educationalinsights.com/amfile/file/download/file/193/product/940/)

Meskipun secara umum, pemain ditugaskan untuk memenuhi papan dengan potongan geometris, tetapi biasanya terdapat tantangan tambahan dengan menaruh beberapa potongan geometris ke papan secara acak. Dari posisi potongan yang acak itulah selanjutnya pemain akan ditantang untuk melanjutkan hingga papan penuh. Berdasarkan website www.educationalinsights.com semakin banyak potongan yang disisakan, maka level dianggap semakin tinggi atau susah. Dalam beberapa permainan, waktu penyelesaian biasanya akan dicatat dan dibandingkan dengan pemain lain.

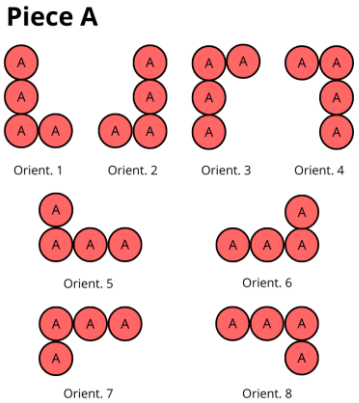
III. Implementasi Solusi

A. *Solusi Umum*

Langkah awal yang digunakan untuk menyelesaikan permainan kanoodle dalam bentuk algoritma adalah dengan menentukan objek-objek yang terlibat. Dengan kata lain, penyelesaian kali ini dilakukan secara berorientasi objek karena kanoodle dapat dipandang permainan yang terdiri dari objek papan atau *board* dan berbagai objek geometri yang akan mengisi papan hingga penuh. Sehingga langkah pertama yang dilakukan adalah membuat objek dengan nama *board* dan objek bernama *piece*.

Objek *board* memiliki atribut *row*, *col*, dan *board*, yang sesuai namanya masing-masing merepresentasikan banyak baris, banyak kolom, dan sebuah array dua dimensi untuk menyimpan isian dari tiap baris dan kolomnya. Untuk permainan kanoodle dasar 2D memiliki 5 baris dan 11 kolom. Kemudian, objek *piece* memiliki atribut kategori dan list dari berisi setiap kemungkinan orientasi dari sebuah bentuk geometri. Orientasi di sini dalam artian bentuk lain hasil dari rotasi, pencerminan, maupun kombinasi keduanya.

Berikut adalah ilustrasi salah satu jenis *piece* beserta orientasinya:



Gambar 4. Ilustrasi Piece A
(Sumber: dokumen penulis)

B. *Algoritma Runut-balik (Backtracking)*

Dalam membuat algoritma runut-balik, langkah pertama adalah mengidentifikasi setiap properti umum *backtracking*:

- Solusi persoalan:

  Permainan akan selesai ketika papan atau *board* telah penuh berisi seluruh *piece* berbeda sebanyak 12 buah tanpa tumpang tindih. Sehingga:

  X = (A, B, C, D, E, F, G, H, I , J, K, L)

  dengan setiap abjad berbeda mewakili jenis *piece* yang berbeda beserta orientasi yang terpilih dan posisi dalam *board* yang memenuhi.

- Fungsi pembangkit:

  Simpul baru dapat dibangkitkan dengan mengambil sebuah orientasi dari jenis *piece* yang belum terletakkan, beserta posisi dalam *board* yang memungkinkan.

- Fungsi pembatas:

  Setelah melakukan pengamatan terhadap setiap bentuk *piece* yang ada, ditemukan batasan-batasan berikut yang akan menyebabkan *backtrack*:

  - Terbentuk pola kosong dengan ukuran 1x1 yang dikelilingi oleh *piece*

  - Terbentuk pola kosong dengan ukuran 1x2 yang dikelilingi oleh *piece* baik secara vertikal maupun horizontal

- Terbentuk pola kosong dengan ukuran 1x3 yang dikelilingi oleh *piece* baik secara vertikal maupun horizontal

- Terbentuk pola kosong dengan ukuran 1x5 yang dikelilingi oleh *piece* baik secara vertikal maupun horizontal

- Terbentuk pola kosong dengan ukuran 2x3 yang dikelilingi oleh *piece* baik secara vertikal maupun horizontal

Ruang-ruang dengan ukuran tersebut tidak akan dapat terisi dengan penuh tanpa tumpang tindih menggunakan *piece* yang ada.

Permasalahan kanoodle memiliki ruang solusi kombinasi antara *piece*, orientasi, dan lokasi pada *board*. Dari properti umum yang sudah didefinisikan, dibuat algoritma runut-balik yang sesuai dengan memanfaatkan obejk *board* dan *piece* secara rekursif. Untuk algoritma secara mendetail dapat dilihat pada repository yang tertera pada bagian 'Link Repository Github' dan terletak pada file backtracking.py. Algoritma tersebut ditulis dalam bahasa python.

C. Algoritma Brute Force

Algoritma brute force memiliki 3 tahapan utama yaitu enumerasi, evaluasi, dan pemilihan. Berikut adalah implementasi setiap tahapan dari permasalahan kanoodle:

- Enumerasi:

  Untuk mendaftar semua kemungkinan solusi, langkah pertama adalah mendata posisi kosong yang tersedia dari *board*. Selanjutnya, dari setiap orientasi dalam setiap *piece* yang tersisa didata untuk setiap penempatan pada posisi kosong *board* tanpa menyebabkan tumpang tindih dengan *piece* yang sudah ada atau posisi valid. Dari data orientasi beserta posisi valid tersebut akan dikombinasikan antar *piece* yang tersisa sebagai hasil enumerasi.

- Evaluasi:

  Setiap hasil enumerasi akan dicoba ditempatkan ke dalam *board* satu per satu. Untuk mengoptimalkan *exhaustive search* dilakukan pendekatan heuristik. Heuristik di sini sama seperti fungsi pembatas pada algoritma runut-balik, yaitu ketika terbentuk pola-pola yang mustahil untuk diisi oleh *piece* secara valid. Sehingga ketika dalam mencoba menempatkan suatu *piece* menyebabkan terbentuknya pola yang mustahil, akan langsung dilanjutkan ke evaluasi untuk enumerasi selanjutnya.

- Pemilihan:

  Pemilihan solusi dilakukan ketika suatu evaluasi dapat memenuhi *board* secara sempurna untuk pertama kalinya. Karena dalam permasalahan ini tidak terdapat bobot apapun dalam setiap langkahnya, sehingga solusi optimal ketika percobaan langkah paling sedikit dan waktu eksekusi paling singkat.

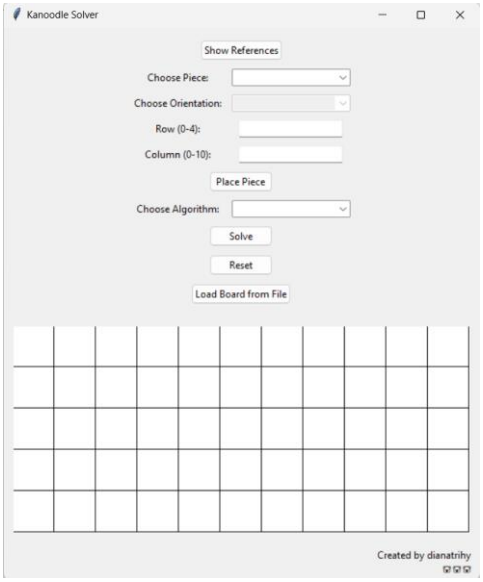
Dari ketiga tahapan tersebut dibuat algoritma brute force yang sesuai dengan memanfaatkan objek *board* dan *piece* pula. Untuk algoritma secara mendetail dapat dilihat pada repository yang tertera pada bagian 'Link Repository Github' dan terletak pada file bruteforce.py. Algoritma tersebut ditulis dalam bahasa python.

D. Visualisasi

Untuk memudahkan pengguna ketika melakukan penyelesaian kanoodle menggunakan algoritma, telah dibuat visualisasi solusi dalam bentuk antarmuka pengguna grafis atau *Graphical User Interface* (GUI). Program ini menyediakan berbagai fungsionalitas atau fitur berikut:

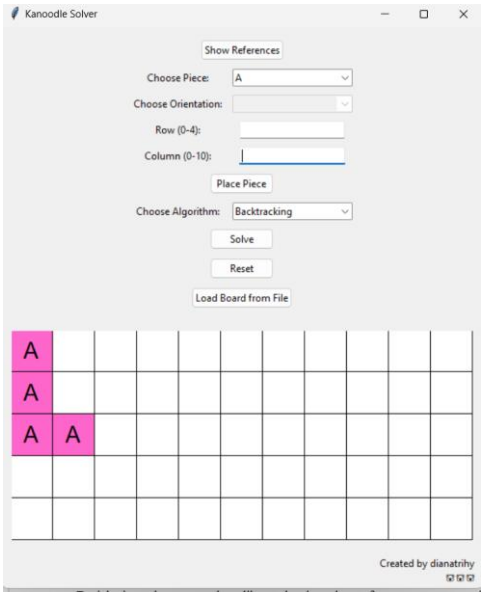
- Melihat kamus atau referensi setiap piece dan orientasinya melalui tombol yang memunculkan jendela (*window*) lain berisi gambar terkait.
- Memilih *piece* yaitu antara huruf A hingga L yang akan ditempatkan ke *board*.
- Memilih orientasi dari *piece* yang sudah dipilih sebelumnya.
- Memilih baris dan kolom untuk menentukan posisi peletakkan *piece* pada *board*.
- Meletakkan *piece* terpilih berdasarkan orientasi dan posisi yang sudah dimasukkan.
- Memilih jenis algoritma antara runut-balik atau brute force untuk menyelesaikannya.
- Menyelesaikan kanoodle dengan *board* sesuai visualisi menggunakan algoritma terpilih.
- Tombol reset untuk mengembalikan kondisi program ke kondisi semula.
- Memuat konfigurasi papan dari sebuah file text yang valid.
- Program dapat menampilkan solusi kanoodle berdasarkan kondisi-kondisi yang telah dipilih sebelumnya, beserta jumlah percobaan peletakkan *piece* pada board dan waktu eksekusinya.

Selain fungsionalitas tersebut, program juga sudah mengantisipasi setiap kemungkinan yang terjadi seperti salah input, terjadi error tidak terduga, maupun ketika solusi kanoodle tidak ditemukan. Berikut adalah beberapa tangkapan layar dari GUI yang telah dibuat;



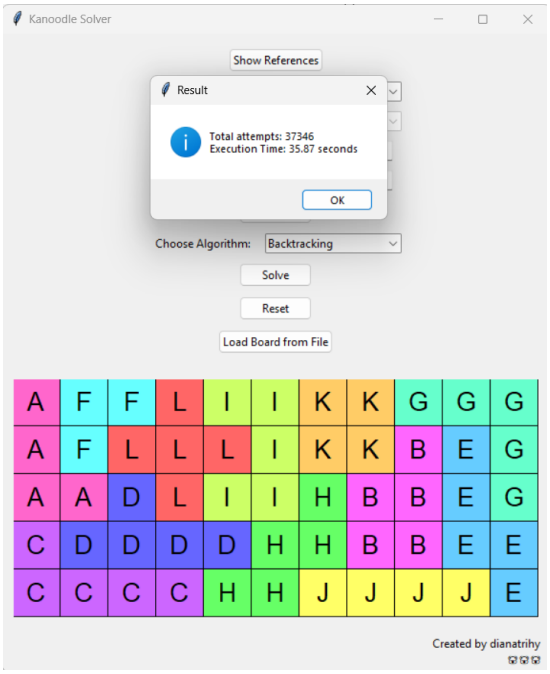
Gambar 5. Tampilan Awal Program

(Sumber: dokumen penulis)



Gambar 6. Tampilan Setelah Peletakkan Sebuah *Piece*

(Sumber: dokumen penulis)



Gambar 7. Tampilan Program Ketika Mendapatkan Solusi

(Sumber: dokumen penulis)

GUI dibuat dalam bahasa python dengan memanfaatkan toolkit TkinterGUI dibuat dalam bahasa python dengan memanfaatkan *toolkit* Tkinter.

IV. Percobaan dan Analisis

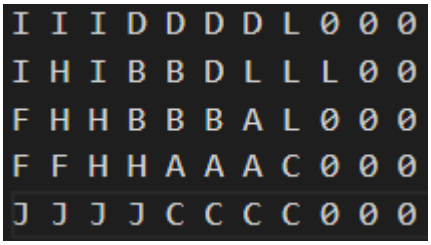
A. *Bahan Uji Coba*

Percobaan dilakukan dengan memberikan beberapa *state* atau kondisi *board* untuk dapat diselesaikan menggunakan algoritma runut-balik dan algoritma brute-force. Dikarenakan kompleksitas algoritma brute-force dalam penyelesaian kanoodle dengan sangatlah besar, kondisi awal *board* yang akan menjadi uji coba adalah ketika 3-5 *piece* tersisa yang belum dimasukkan. Berikut adalah enam *state board* yang menjadi bahan uji coba:

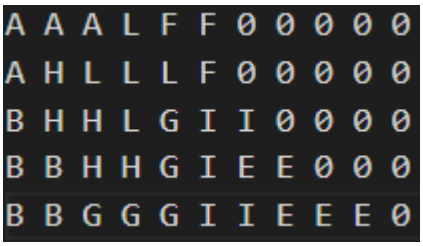
- Uji Coba 1:

```
J J J J B B D K K 0 0
I I I E B B D K K 0 0
I H I E B G D D 0 0 0
F H H E E G D 0 0 0 0
F F H H E G G G 0 0 0
```

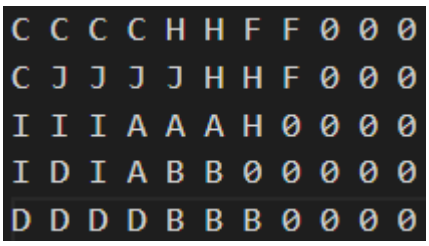
- Uji Coba 2:



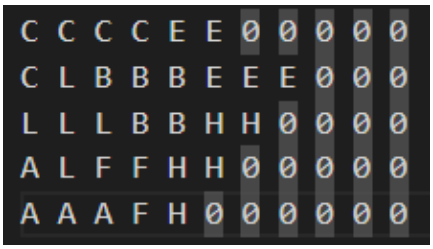
- Uji Coba 3:



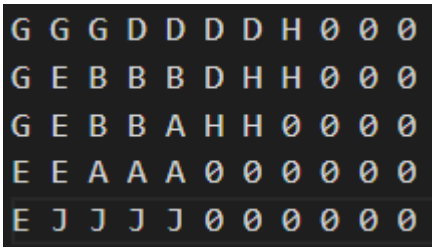
- Uji Coba 4:



- Uji Coba 5:



- Uji Coba 6:



B. Analisis Waktu Eksekusi

Berikut adalah hasil uji coba masing-masing algoritma berdasarkan lama waktu eksekusi dalam detik:

Tabel 1. Hasil Uji Coba dengan Catatan Waktu Eksekusi

Uji Coba	Runut-balik	Brute Force
1	0.01	0.29
2	0.03	0.18
3	0.01	0.03
4	0.03	7.11
5	0.04	145.89
6	0.03	469.13
Jumlah	**0.15**	**622.63**
Rata-rata	**0.025**	**103.77**

Terlihat dari hasil uji coba, setiap percobaan algoritma runut-balik selalu relatif lebih cepat. Terlihat juga melalui rata-rata waktu eksekusi yang terpaut hingga 103.745 detik. Hal ini menunjukkan kesesuaian antara teori dengan hasil uji coba. Algoritma runut-balik berdasarkan teori memang seharusnya lebih cepat daripada algoritma brute force. Hal tersebut terjadi karena adanya sistem *backtrack* dalam algoritma runut balik yang dapat menyebabkan pengecekan tidak perlu dari titik awal kembali. Sedangkan pada algoritma brute force yang tidak mengenal sistem *backtrack*, harus mengecek dari titik awal kembali ketika ditemukan suatu kondisi yang menyebabkan tidak menuju solusi.

C. Analisis Kompleksitas

Berikut adalah hasil uji coba masing-masing algoritma berdasarkan banyaknya percobaan menempatkan *piece* ke dalam *board*:

Tabel 2. Hasil Uji Coba dengan Catatan Kompleksitas

Uji Coba	Runut-balik	Brute Force
1	4	63
2	16	39
3	8	13
4	16	1137
5	28	32005
6	22	121545
Jumlah	94	154802
Rata-rata	15.67	25800.34

Terlihat dari hasil uji coba, setiap percobaan algoritma runut-balik selalu memiliki jumlah percobaan yang lebih sedikit. Terlihat juga melalui rata-rata percobaan yang terpaut hingga 25784.67 percobaan. Hal ini menunjukkan kesesuaian pula antara teori dengan hasil uji coba. Banyaknya jumlah percobaan di sini dapat mencerminkan tingkat kompleksitas algoritma dan banyaknya ruang memori yang diperlukan.

Algoritma runut-balik berdasarkan teori memang seharusnya memiliki kompleksitas yang lebih sederhana dibandingkan algoritma brute force. Hal tersebut terjadi karena adanya system pemangkasan (*pruning*) dalam algoritma runut balik yang dapat menyebabkan pembangkitan simpul diminimasi. Sedangkan pada algoritma brute force yang hanya mengandalkan fungsi heuristik akan membangkitkan simpul

yang lebih banyak. Algoritma brute force bahkan dapat membutuhkan memori yang lebih besar lagi apabila tanpa menggunakan pendekatan heuristik.

V. KESIMPULAN

Algoritma runut-balik maupun algoritma brute force dapat menyelesaikan permasalahan kanoodle. Meskipun waktu dan langkah yang diperlukan algoritma brute force sangat besar, terutama jika dibandingkan algoritma runut balik. Dalam uji coba papan dengan kekurangan 3-5 potongan geometri, algoritma runut-balik mampu menyelesaikan dengan rata-rata waktu kurang dari 0.1 detik, sedangkan algoritma brute force memiliki rata-rata lebih dari 1 menit. Begitu pula memori atau langkah yang dibutuhkan, rata-rata algoritma runut-balik hanya membutuhkan sekitar 15 langkah, sedangkan algoritma brute force hingga lebih dari 25 ribu langkah. Hal ini menunjukkan bahwa algoritma runut-balik sangat dapat mengoptimalkan algoritma pencarian menyeluruh, meskipun sudan dengan heuristik.

LINK REPOSITORY GITHUB

GitHub - dianatrihy/kanoodle-solver-backtracking

UCAPAN TERIMA KASIH

Penulis ingin menyampaikan apresiasi kepada berbagai pihak yang telah memberikan dukungan. Pertama-tama, penulis bersyukur kepada Allah SWT atas karunia dan rahmat-Nya, yang memungkinkan penulis menyelesaikan makalah berjudul " Perbandingan Algoritma Runut-balik (Backtracking) dengan Algoritma Brute Force dalam Penyelesaian Permainan Kanoodle" dengan baik. Selanjutnya, penulis mengucapkan terima kasih yang tulus kepada Dr. Ir. Rinaldi Munir, M.T., Dr. Nur Ulfa Maulidevi, dan Dr. Ir. Rila Mandala, sebagai dosen mata kuliah Strategi Algoritma, atas bimbingan dan arahannya selama perkuliahan. Tak lupa, penulis juga mengucapkan terima kasih kepada semua pihak yang telah memberikan referensi yang sangat membantu dalam penulisan makalah ini.

REFERENSI

[1] Educational Insight. Tidak Diketahui. "Kanoodle Guide". https://www.educationalinsights.com/amfile/file/download/file/193/product/940/ diakses pada 12 Juni 2024

[2] Munir, Rinaldi. 2021. "Algoritma Runut-balik (Backtracking) (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf diakses pada 12 Juni 2024

[3] Munir, Rinaldi. 2021. "Algoritma Runut-balik (Backtracking) (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian2.pdf diakses pada 12 Juni 2024

[4] Munir, Rinaldi. 2022. "Algoritma Brute Force (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf diakses pada 12 Juni 2024

[5] Munir, Rinaldi. 2022. "Algoritma Brute Force (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/20212022/Algoritma-Brute-Force-(2022)-Bag2.pdf diakses pada 12 Juni 2024

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, 12 Juni 2024

Diana Tri Handayani
13522104