# Perbandingan Algoritma Brute Force dan Algoritma Branch and Bound pada Penyelesain Hacking Minigame di Game Fallout

Venantius Sean Ardi Nugroho - 13522078
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522078@std.stei.itb.ac.id

***Abstrak*—Pada Minigame Hacking pada game Fallout pemain diminta untuk memilih kata yang tepat dari himpunan kata. Suksesnya minigame ini ditentukan oleh kecerdikan pemain dalam memilih kata dengan langkah sesedikit mungkin dengan manfaatkan clue yang diberikan. Makalah ini menyelidiki perbedaan penggunaan algoritma Branch and Bound dan Brute Force dalam menyelesaikan permainan ini berdasarkan kemungkinan menang dan rata - rata langkah yang diperlukan untuk memilih solusinya .**

***Keywords—Fallout; hacking minigame;branc and bound; brute force;***

## I. PENDAHULUAN

Fallout merupakan seri video game *role playing game* yang bertemakan tentang kehidupan paska perang nuklir antar negara. Seri ini awalnya diciptakan oleh Interplay Entertainment pada tahun 1997. Semenjak tahun 2008, seri ini dikembangkan dan diterbitkan oleh Bethesda Softworks. Seri ini memiliki 6 game utama sejak pertama kali dirilis, yaitu: Fallout, Fallout 2, Fallout 3, Fallout: New Vegas, Fallout 4, serta Fallout 76. Selain keenam game tersebut, seri ini juga memiliki 2 game *spin-off* yaitu Fallout Tactics serta Fallout Shelter.

Pada game Fallout 3, Fallout: New Vegas, Fallout 4, dan juga Fallouot 76 terdapat *minigame* (game di dalam game) yaitu hacking. Dengan menyelesaikan *minigame* tersebut, pemain dapat mendapatkan barang – barang yang berguna.


Gambar 1.1 Tampilah *minigame* Hacking

Masalah yang dijajakan oleh *minigame* tersebut adalah untuk memilih kata yang tepat dari sebuah pilihan kata – kata. Seperti yang bisa dilihat di Gambar 1.1, terdapat beberapa kata dalam bahasa Inggris seperti TERMS,FRIES, dan TRIED (semuanya memiliki jumlah karakter samaa). Karakter – karakter yang berada di antara kata – kata tersebut tidak relevan karena pada *minigame* tersebut hanya kata – kata yang valid sebagai jawabanlah yang dapat dipilih sebagai kata (ditandakan dengan di-*highlight*-nya kata tersebut, selain kata – kata tersbut hanya meng-*highlight* karakter).

Dalam penyelesaian permainan ini, pemain hanya diberikan 4 kali kesempatan untuk memilih kata sebelum permainan gagal. Untuk membantu pemain dalam mencari kata yang benar dalam daftar kata tersebut, *minigame* akan memberikan *clue* berupa angka setelah sebuah pemilihan kata dilakukan. Angka tersebut merepresentasikan kemiripan dari kata yang dipilih dengan kata yang benar. Kemiripan pada kasus ini dideskripsikan sebagai berapa jumlah karakter yang memiliki karakter yang sama pada posisi yang sama. Sebagai contoh, bila kata yang benar adalah TERMS dan pemain memilih TRIED, maka permainan akan memberi keluaran *clue* yaitu 1 karena karakter ”T” pada kedua kata tersebut ada pada posisi yang sama (indeks ke – 0).

Tentu saja, permainan ini bisa saja diselesaikan dengan cara *brute force* dengan cara memilih karakter secara random, namun mengingat kesempatan pemilihan kata kita, strategi ini mungkin tidak begitu efektif, untuk itu dibutuhkan Algoritma yang lebih baik untuk menyelesaikan persoalan ini. Karya tulis ini membahas tentang penggunaan Algoritma Branch and Bound untuk meningkatkan kesempatan menang dalam permainan Hacking pada game Fallout.

## II. LANDASAN TEORI

### A. Pohon Ruang Status

Ruang status , atau yang dalam bahasa Inggris dikenal sebagai *state space* merupakan himpunan seluruh status dari sebuah sistem. Tiap koordinat merupakan variabel status dan tiap nilai dari status bisa mereprersentasikan status dari sistem.

Tiap simpul dalam ruang status berkorespondensi dengan status yang berbeda dalam sistem.

Ruang status dapat direpresentasikan sebagai tuple [N,, A, S, G] dengan :

- N adalah himpunan status
- A adalah himpunan sisi yang menghubungkan status – status
- S adalah himpunan tak kosong yang berisi *start states*
- G adalah himpunan tak kosong yang berisi *goal states*

Ruang status bisa direpresentasikan dengan beberapa struktur, salah satunya adalah pohon. Ruang status yang memiliki struktur pohon dinamakan pohon ruang status. Pada makalah ini didefinisikan simpul ekspan sebagai simpul yang sedang diperiksa dan di-"ekspan" untuk mendapatkan anak – anaknya, sedangkan simpul hidup adalah simpu – simpul yang mungkin menjadi pilihan dalam persoalan.

### B. *Algortima Branch and Bound*

Algortima Branch and Bound adalah salah satu strategi dalam strategi algoritma yang bekerja dengan membagi masalah menjadi lebih kecil dalam bentuk *branch* pada *state space tree* serta mematikan beberapa cabang yang tidak mengarah ke solusinya. Algoritma ini umumnya digunakan untuk permasalahan optimisasi.

Sifat dari algoritma Branch and Bound ada beberapa, di antaranya adalah :

- Menggunakan fungsi pembatas untuk melakukan pemangkasan jalur yang tidak mengarah ke solusi.
- Memilih berdasarkan cost simpul yanng paling kecil
- Bila semua cost node pada simpul hidup sama, maka algoritma akan memilih berdasarkan urutan pembangkitan.

Cost pada tiap simpul yang biasanya direpresentasikan dengan lambang $\hat{c}(i)$ merupakan nilai taksiran lintasan termuran ke simpul status tujuan dengan melaui status i.

Permasalahan yang dapat diselesaikan oleh algoritma ini dibagi menjadi 2 kategori yaitu permasalahan yang letak simpul solusinya sudah diketahui serta permasalahan yang letak simpul solusinya tidak diketahui. Contoh permasalahan yang letak simpulnya diketahui adalah permasalahan N-Queens, sedangakn permasalahan yang tidak dikethui lletak simpul soluinya adalah 15-puzzle. Untuk permasalahan yang letak simpul solusinya tidak diketahui, nilai costnya perlu dihitung secara heuristik.

### C. *Algortima Brute Force*

Algoritma Brute Force adalah algoritma yang sederhana dalam menyelesaikan suatu permasalahan, lebih spesifiknya adalah dengan cara mencari semua opsi yang mungkin sampai suatu solusi ditemukan. Ciri – ciri dari algoritma ini adalah algorima Brute Force sangat bergantung pada kekuatan komputasi. Algoritma ini berguna sebagai *benchmark* dari strategi lain yang ingin diuji menyelesaikan suatu permasalahan.

## III. IMPLEMENTASI SOLUSI

### A. *Pemetaan Solusi pada Elemen Algoritma Branch and Bouund*

Seperti yang ditulis pada Bab II, algoritma Branch and Bound biasanya digunakan untuk menyelesaikan persoalan optimasi. Pada persoalan *minigame* Hacking, hal yang ingin dioptimalisasi adalah langkah yang diperlukan untuk menyelesaikan permainan Hacking. Meskipun begitu, perlu diingat bahwa pada game ini terdapat elemen keberuntungan, sehingga menebak kata secara optimal (hanya dalam 1 langkah) tidak dapat dipastikan, sasaran dari algoritma ini hanyalah untuk mencoba mendapatkan solusi dengan langkah terkecil menggunakan clue yang diberikan. Akibat dari elemen keberntungan selain itu adalah algoritma ini tidak tahu letak simpul solusinya. Oleh Sebab itu perlu ditemukan cost heuristik untuk menyelesaikan permasalahan ini.

Pohon ruang status pada permasalahan ini memiliki simpul – simpul dengan cost yang merepresentasikan seberapa mirip sebuah kata dengan kata solusinya. Root node pada pohon ruang status diberi cost 100 dan digunakan untuk membangkitkan anak – anaknya. Anak – anak dari root node, yaitu simpul pada level 1, memiliki cost sebanyak jumlah karakter kata tersebut karena pada kondisi sekarang, kita tidak tahu berapa mirip tiap kata yang kita pilih. Simpul – simpul selain kasus khusus tersebut memiliki cost yang yang bisa dihitung sebagai :

$$\hat{c}(i) = f(i) + \hat{g}(i) \qquad (1)$$

Dengan :

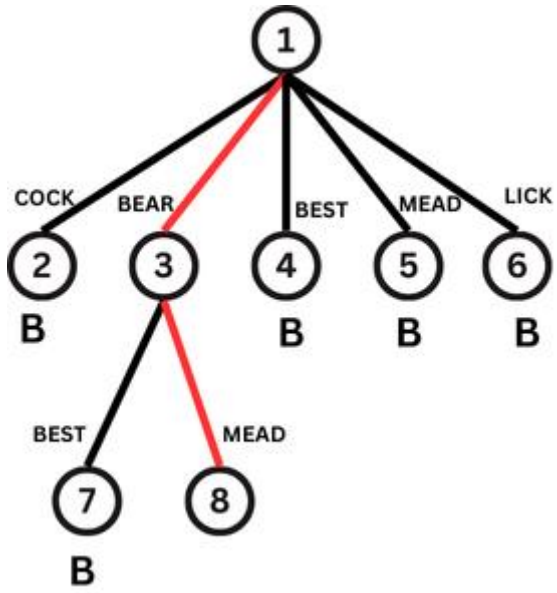
1. $\hat{c}(i)$ : Cost simpul i
2. $f(i)$ : Cost parent dari simpul i
3. $\hat{g}(i)$ : Kemiripan * -1

Kemiripan adalah nilai yang ditunjukkan oleh clue dari permainan. Karena nilai kemiripan disediakan oleh permainan, kemiripan harus diinput secara manual tiap kali pemilihan kata

Setelah pemain mendapatkan nilai kemiripan, program dapat melakukan pembangkitan simpul – simpul hidup. Fungsi pembangkit dari program ini menerima parent node, tingkat kemiripan , dan set kata dari permasalahan. Simpul - simput yang dibangkitkan oleh Fungsi tersebut hanyalah simpul yang mungkin memiliki tingkat kemiripan yang diberikan. Sebagai contoh , misal terdapat set kata sebagai berikut (BEAR,BEST,COCK,MEAD,LICK) ,dipilih kata BEAR dengan tingkat kemiripan 2. Maka simpul – simpul yang akan dibangkitkan adalah BEST dengan cost 2 dan MEAD dengan cost 2. Terlihat di sini bahwa dengan clue tingkat kemiripan, kita bisa memangkas cukup banyak opsi karena jawabannya pasti antara BEST dan MEAD , opsi selain itu bisa dipangkas. Pada ilustrasi ini juga tergambarkan bahwa dengan tingkat kemiripan, belum bisa ditentukan karakter apa yang sama , pada kasus ini ada dua opsi, BE pada BEAR dan BEST yang sama atau EA pada MEAD dan BEAR yang sama. Kriteria pemangkasan simpul yang lainya adalah memangkas simpul dengan nama yang sebelumnya pernah ditebak.

Setelah simpul – simpul dibangkitkan, perlu dilakukan pemilihan simpul. Simpul dipilih berdasarkan cost dengan nilai terkecil. Untuk membantu pemilihan ini, digunakan *priority queue* sebagai simpul hidupnya. Simpul expannya dipilih sebagai anggota terdepan dari simpul hidup.

Ilustrasi dari contoh persoalan game Hacking dengan set kata (COCK,BEAR,BEST, MEAD,LICK) dan jawaban adalah MEAD adalah sebagai berikut :



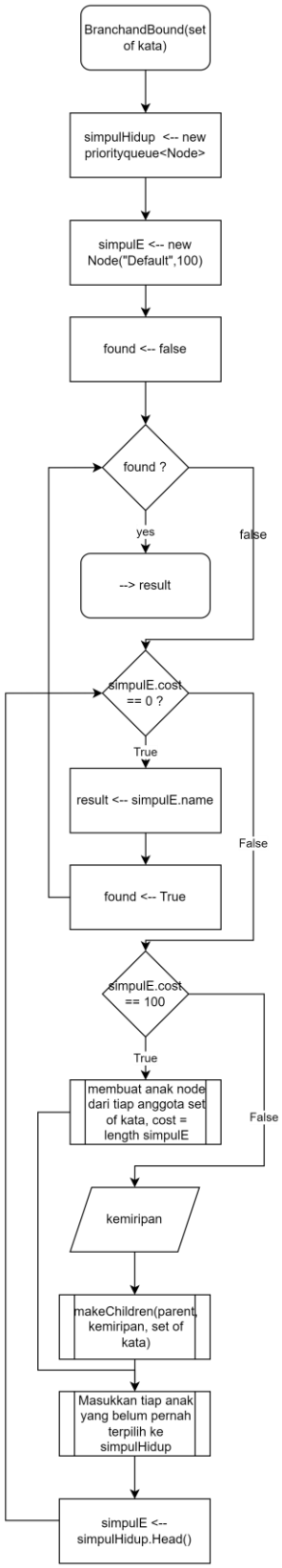

Gambar 3.1 Ilustrasi Contoh Penyelesaian Masalah

Dengan angka pada tiap simpul merepresentasikan urutan pembangkitan. Tabel keadaan simpul ekspan dan simpul bisa direpresentasikan sebagai berikut :

Tabel 3.1 Keadaan Simpul Ekspan dan Simpul Hidup

| Iterasi ke - | Simpul E | Simpul Hidup |
|---|---|---|
| 1 | 1 | 2,3,4,5,6 |
| 2 | 2 | 3,4,5,6 |
| 3 | 3 | 7,8,3,4,5,6 |
| 4 | 7 | 8,3,4,5,6 |
| 5 | 8 | Jawaban ditemukan |

Secara garis besar, alur dari program ini adalah sebagai berikut :



Gambar 3.2 Diagram Alur Algoritma Branch and Bound

*B. Implementasi Solusi dengan Brute Force*

Menyelesaikan permasalahan ini dengan strategei Brute Force dapat dilakukan dengan cukup sederhana. Caranya adalah dengan membandingkan tiap kata satu per satu dengan kata jawabannya. Algoritma akan berhenti bila kata yang dibandingkan sama dengan jawabannya.

## IV. HASIL DAN PEMBAHASAN

Pada eksperimen, terdapat dua parameter yang diuji yaitu *win rate* serta rata – rata steps yang diperlukan untuk menyelesaikan permainan. *Win rate* didefinisikan sebagai banyak permainan yang diselesaikan dengan langkah kurang dari sama dengan 4 dibandingkan dengan banyak eksperimen yang dilakukan.

Terdapat beberapa kasus uji yang diujikan pada tiap algoritma. Kategori kasus uji yang dibuat terdiri dari :

- Jumlah kata 7 dengan panjang kata 4 karakter
- Jumlah kata 10 dengan panjang kata 4 karakter
- Jumlah kata 13 dengan panjang kata 4 karakter
- Jumlah kata 7 dengan panjang kata 5 karakter
- Jumlah kata 10 dengan panjang kata 5 karakter
- Jumlah kata 13 dengan panjang kata 5 karakter
- Jumlah kata 7 dengan panjang kata 6 karakter
- Jumlah kata 10 dengan panjang kata 6 karakter
- Jumlah kata 13 dengan panjang kata 6 karakter

Untuk setiap kategori akan disediakan 3 kasus uji supaya data bisa lebih beragam dan untuk setiap kasus uji, dilakukan repetisi sebanyak 1000 kali supaya data yang didapat akurat. Untuk mengakomodasi ekperimen ini yang banyak repetisinya, dibuat versi dari algoritma branch and bound yang bisa ”bermain dengan dirinya sendiri” , dengan kata lain memilih kata sendiri dan menentukan kemiripan sendiri. Test case yang diuji bisa dilihat pada folder test di link repositori.Berikut adalah hasil eksperimen dengan kriteria seperti yang telah dijelaskan di atas:

Tabel 4.1 Data Hasil Eksperimen Branch and Bound

| Jumlah Kata di Set | Jumlah Karakter pada Kata | Test Case ke - | Branch and Bound | |
|---|---|---|---|---|
| | | | Win Rate (%) | Average Steps |
| 7 | 4 | 1 | 72.1 | 3 |
| | | 2 | 86.4 | 2 |
| | | 3 | 83.8 | 3 |
| | 5 | 1 | 72.4 | 3 |
| | | 2 | 100 | 2 |
| | | 3 | 86.9 | 2 |
| | 6 | 1 | 87.9 | 2 |
| | | 2 | 71.1 | 3 |
| | | 3 | 100 | 2 |
| 10 | 4 | 1 | 70 | 3 |
| | | 2 | 70.5 | 3 |
| | | 3 | 80.1 | 3 |
| | 5 | 1 | 71.7 | 3 |
| | | 2 | 51.8 | 4 |
| | | 3 | 71.7 | 3 |
| | 6 | 1 | 90.2 | 3 |
| | | 2 | 100 | 2 |
| | | 3 | 77 | 3 |
| 13 | 4 | 1 | 76.8 | 3 |
| | | 2 | 74.9 | 3 |
| | | 3 | 85.1 | 3 |
| | 5 | 1 | 61.3 | 4 |
| | | 2 | 69.2 | 3 |
| | | 3 | 54.4 | 4 |
| | 6 | 1 | 76.3 | 3 |
| | | 2 | 77.4 | 3 |
| | | 3 | 61.4 | 4 |
| | | **Average** | 77.05 | 2.92 |

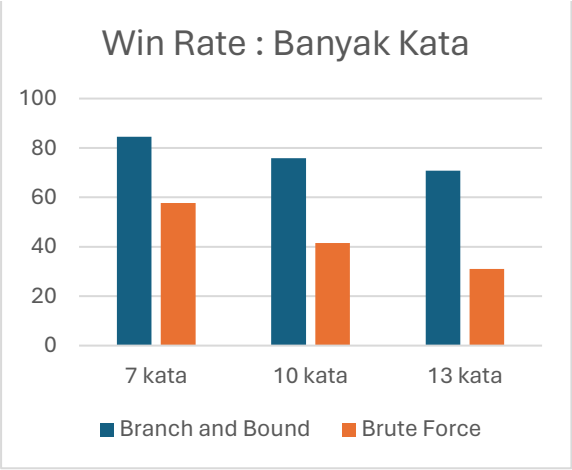
Tabel 4.2 Data Hasil Eksperimen Brute Force

| Jumlah Kata di Set | Jumlah Karakter pada Kata | Test Case ke - | Brute Force | |
|---|---|---|---|---|
| | | | Win Rate (%) | Average Steps |
| 7 | 4 | 1 | 56.6 | 4 |
| | | 2 | 55.6 | 4 |
| | | 3 | 58.4 | 3 |
| | 5 | 1 | 58.8 | 3 |
| | | 2 | 57.2 | 3 |
| | | 3 | 57 | 3 |
| | 6 | 1 | 570 | 4 |
| | | 2 | 59.1 | 3 |
| | | 3 | 59.2 | 3 |
| 10 | 4 | 1 | 40.4 | 5 |
| | | 2 | 42.4 | 5 |
| | | 3 | 42 | 5 |
| | 5 | 1 | 39.4 | 5 |
| | | 2 | 40.5 | 5 |
| | | 3 | 41.8 | 5 |
| | 6 | 1 | 39.8 | 5 |
| | | 2 | 41.7 | 5 |
| | | 3 | 46 | 4 |
| 13 | 4 | 1 | 29.7 | 7 |
| | | 2 | 31.7 | 7 |
| | | 3 | 33.1 | 6 |

| | | | | |
|---|---|---|---|---|
| | 5 | 1 | 29.5 | 7 |
| | | 2 | 28.6 | 7 |
| | | 3 | 33 | 6 |
| | 6 | 1 | 33.4 | 6 |
| | | 2 | 29 | 7 |
| | | 3 | 31.1 | 6 |
| | | **Average** | 43.41 | 4.92 |

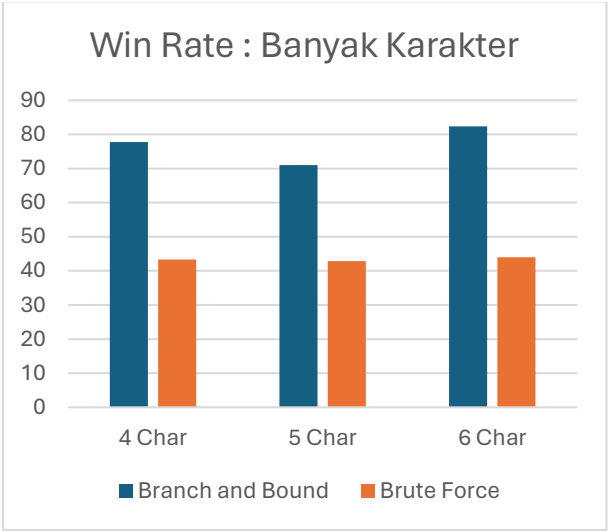
Dari data di atas, bisa diambil bahwa algoritma Branch and Bound memiliki rata – rata win rate yang lebih besar dari algortima Brute Force serta memiliki rata – rata langkah yang diperlukan untuk mencapai tujuan lebih lebih kecil dari algoritma Brute Force. Penggunaan algoritma Branch and Bound berhasil meningkatkan tingkat kesuksesan sebesar kurang lebih 75 %. Lebih pentingnya lagi, berdasarkan data tersebut, didapat bahwa pemain akan lebih sering menang dalam permainan Hacking.

Beberapa data yang *stand out* dari hasil ekseperimen terlihat pada eksperimen algoritma Branch and Bound pada test case 7-5-2,7-6-3, dan 10-6-2. Pada ketiga test case tersebut memiliki win rate 100% walaupun jumlah eksperimennya sebanyak 1000. Kejadian ini kemungkinan terjadi karena dengan algoritma yang diimplementasikan memilih simpul dengan cost yang sama secara tidak acak, melainkan sesuai urutan masuk. Hal tersebut mengakibatkan pada kondisi tertentu jawaban pasti ditemukan. Sebagai contoh, test case 7-5-2 isinya adalah sebagai berikut : plead;asset;chaos;radio;pride;shake;frame; . Bila solusinya adalah 4 pilihan kata pertama, maka jawaban pasti akan ditemukan. Bila solusinya adalah ”pride” rute yang dipakai pasti plead → pride, bila solusinya ”shake” rute yang dipakai pasti plead → asset → chaos → shake, sedangkan bila solusinya ”frame” rute yang dipakai pasti plead → asset → chaos → frame. Dari node chaos, bisa ditentukan node yang bersesuiain karena ”shake” memiliki tingkat kemiripan yang lebih dibandingkan ”frame” bila dibandingkan dengan ”chaos”. Kesimpulannya, pada skenario terburuk pun , pada beberapa persoalan, solusi pasti ditemukan dengan Algoritma Branch and Bound. Skenario tersebut tidak ada pada algoritma Brute Force, kecuali jika jumlah kata dalam permainannya hanya empat.




Gambar 4.1 Perbandingan Win Rate dan Jumlah Kata dalam Permainan

Diagram di atas menunjukkan bahwa algoritma Branch and Bound serta algoritma Brute Force memiliki korelasi negatif dengan banyaknya kata yang terdapat dalam permainan. Alasan dari korelasi tersebut adalah karena dengan bertambahnya kata dalam permainan, kemungkinan jawaban yang salah akan meningkat sedangkan jawabannya tetap saja satu kata. Akibatnya peluang untuk memilih kata yang tepat yang berkurang. Perlu diperhatikan pula bahwa pada algoritma Brute Force penurunan *win rate* yang dihasilkan lebih tajam daripada pada algoritma Brute Force. Hal ini membuktikan kekuatan dari pemangkasan yang terdapat pada algortima tersebut.




Gambar 4.1 Perbandingan Win Rate dan Jumlah Karakter pada Setiap Kata dalam Permainan

Diagram di atas menunjukkan bahwa algortima Branch and Bound dan algoritma Brute Force tidak memiliki korelasi yang signifikan dengan jumlah karakter pada tiap kata dalam permainan. Hal ini mungkin terjadi karena dengan bertambahnya karakter pada kata, memungkinkan lebih banyak matches yang bisa digunakan untuk pemangkasan, tapi di lain pihak pasangan kombinasi karakter yang mungkin match juga meningkat, mengakibatkan lebih banyaknya anak yang dihasilkan suatu simpul.

## V. KESIMPULAN

Telah dilaksanakan eksperimen perbandingan algoritma Brute Force dan algoritma Branch and Bound pada penyelesain Hacking Minigame di Game Fallout. Dari eksperimen, didapatkan informasi sebagai berikut :

- Algoritma Branch and Bound memiliki *win rate* yang lebih besar dibanding algoritma Brute Force sebesar 75%
- Algoritma Branch and Bound memiliki rata – rata jumlah langkah yang lebih kecil dibanding algoritma Brute Force sebesar 60%

- Peningkatan jumlah kata dalam permainan akan menurukan performa algortima Brute Force maupun algoritma Branch and Bound, namun algoritma Branch and Bound dapat menekan penurunan ini cukup signifikan.
- Tidak ditemukan korelasi yang signifikan antara jumlah karakter dalam tiap kata dalam permaian dengan algoritma Brute Force maupun Branch and Bound.

## VI. Ucapan Terima Kasih

Penulis dengan ini menyampaikan terima kasih, terutama kepada pihak – pihak berikut :

1. Tuhan Yang Maha Esa.
2. Bapak dan Ibu Dosen mata kuliah IF 2211, Strategi Algoritma.
3. Kedua orang tua penulis
4. Pembaca sekalian yang sudah memberikan waktu dan perhatiannya untuk membaca makalah ini.

## TAUTAN KODE PROGRAM

https://github.com/Leaguemen/Branch_and_Bound_Fallout_Solver.git

## REFERENSI

[1] R. Munir, “Algoritma Branch and Bound (Bagian 1).” Accessed: Oct. 06, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

[2] R. Munir, “Algoritma Brute Force (Bagian 1).” Accessed: Oct. 06, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

[3] H. Kumar, “Branch and Bound Algorithm.” Accessed: Nov. 06, 2024. [Online]. Available: https://www.geeksforgeeks.org/branch-and-bound-algorithm/

[4] F. Community, “Hacking.” Accessed: Oct. 06, 2024. [Online]. Available: https://fallout.fandom.com/wiki/Hacking

[5]D. Nykamp, “State Space Definition.” Accessed: Nov. 06, 2024. [Online]. Available: https://mathinsight.org/definition/state_space

## 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, 11 Juni 2023

Venantius Sean Ardi Nugroho, 13522078