Penerapan Algoritma *Brute Force* untuk Menyelesaikan Permainan Kartu 24

Muhamad Rafli Rasyiidin - 13522088
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522088@std.stei.itb.ac.id

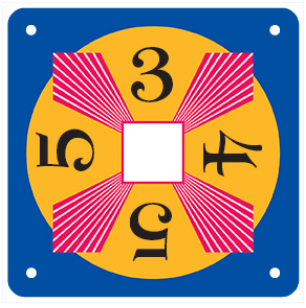
***Abstract*—Permainan kartu 24 adalah sebuah permainan yang mengandalkan kemampuan berpikir dan kemampuan berhitung. Permainan ini akan membuat pemainnya melakukan analisis dan perhitungan untuk memperoleh hasil 24 dari 4 kartu yang diberikan. Namun, ada beberapa kombinasi kartu yang cukup sulit untuk ditemukan solusinya jika dihitung secara manual. Strategi Algoritma dapat dimanfaatkan untuk menemukan solusi dari kombinasi kartu tersebut. Salah satu algoritma yang dapat digunakan adalah algoritma *Brute Force*. Makalah ini akan menjelaskan prinsip dasar dan implementasi penyelesaian permainan kartu 24 dengan menggunakan algoritma *Brute Force*.**

Keywords—Brute Force; 24 Game Cards; Catalan Number;

I. Pendahuluan

Permainan kartu 24 merupakan salah satu jenis permainan kartu yang ada di dunia. Permainan ini termasuk ke dalam salah satu permainan yang dapat mengasah kemampuan berpikir seseorang karena memerlukan analisis dan perhitungan untuk memainkannya. Permainan ini memanfaatkan kemampuan berhitung dasar, yaitu pertambahan, pengurangan, perkalian, pembagian, dan sifat aljabar, untuk mendapatkan solusinya yang bernilai 24.

Permainan kartu 24 cukup banyak dimainkan di kalangan masyarakat karena cara bermainnya yang mudah dan sederhana, serta tidak memerlukan alat-alat yang rumit. Permainan ini biasanya dimainkan menggunakan kartu remi. Namun, seiring berkembangnya zaman, permainan ini mulai dapat dimainkan di aplikasi *mobile* atau pun melalui *website*.



Gambar 1.1. Permainan kartu 24 pada *website*

Sumber: https://www.24game.com/t-about-howtoplay.aspx

Permainan ini dapat memiliki banyak solusi yang berbeda. Setiap pemain diizinkan untuk memiliki jawaban yang berbeda-beda selama hasil operasinya bernilai 24. Namun, terkadang terdapat beberapa kombinasi kartu yang tidak dapat mencapai nilai 24 sehingga membuat pemain kebingungan karena mengira kombinasi tersebut memiliki jawaban dan terus berusaha mencarinya, padahal tidak ada jawaban yang dapat menyelesaikannya. Salah satu cara untuk mengecek apakah kombinasi kartu pada permainan dapat mencapai nilai 24 adalah dengan menggunakan algoritma *Brute Force*. Algoritma *Brute Force* dapat menghasilkan seluruh kemungkinan operasi yang dapat mencapai nilai 24.

II. Teori Dasar

A. Algoritma Brute Force

Algoritma *Brute Force*, terkadang disebut juga algoritma naif, merupakan algoritma untuk memecahkan suatu persoalan dengan menggunakan pendekatan yang lempang (*straightforward*). Algoritma *Brute Force* biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Algoritma *Brute Force* memecahkan masalah dengan sangat sederhana, langsung dengan cara yang jelas (*obvious way*).

Algoritma *Brute Force* bekerja dengan cara mengecek semua kemungkinan yang dapat terjadi pada suatu persoalan. Algoritma ini tidak melakukan teknik optimasi apa pun sehingga waktu yang diperlukan untuk menemukan solusi cukup lama karena harus memeriksa seluruh kombinasi yang ada. Namun, algoritma ini juga dapat menyelesaikan hampir seluruh jenis persoalan selama solusi dapat dipastikan ada.

Algoritma *Brute Force* memiliki beberapa karakteristik:

1. Algoritma *Brute Force* umumnya tidak efektif dan tidak efisien karena harus memeriksa seluruh kemungkinan yang ada sehingga memerlukan *resource* dan volume komputasi yang besar, serta waktu penyelesaian yang lama.
2. Algoritma *Brute Force* lebih cocok untuk persoalan yang ukurannya kecil. Hal tersebut dikarenakan algoritma *Brute Force* dapat diimplementasi dengan mudah dan lebih sederhana daripada algoritma lainnya. Selain itu, algoritma *Brute Force* sering

digunakan sebagai basis untuk membandingkan algoritma lain yang lebih efektif dan efisien.
3. Algoritma *Brute Force* dapat menyelesaikan hampir seluruh persoalan yang ada. Bahkan, beberapa persoalan hanya dapat diselesaikan dengan menggunakan algoritma *Brute Force*. Sangat sulit untuk menunjukkan persoalan yang tidak dapat diselesaikan dengan algoritma *Brute Force*.

Algoritma *Brute Force* memiliki beberapa kelebihan:

1. Dapat digunakan untuk menyelesaikan hampir sebagian besar persoalan yang ada
2. Sederhana dan mudah dipahami
3. Dapat menghasilkan algoritma yang layak untuk beberapa persoalan
4. Dapat menghasilkan algoritma standard untuk beberapa tugas komputasi

Algoritma *Brute Force* memiliki beberapa kekurangan:

1. Jarang menghasilkan algoritma yang efektif dan efisien
2. Lambat ketika persoalan berukuran besar
3. Tidak sekontruktif atau sekreatif algoritma penyelesaian masalah lainnya

B. *Permainan Kartu 24*

Permainan kartu 24 merupakan permainan matematika yang melibatkan manipulasi angka dengan menggunakan operasi aritmatika dasar, yaitu pertambahan, pengurangan, perkalian, dan pembagian, untuk mencapai nilai 24. Permainan ini biasanya menggunakan kartu remi untuk membentuk kombinasi 4 angka. Pada permainan ini, jenis kartu (*heart, diamond, club, spade*) diabaikan dan fokus pada angka yang terdapat pada kartu. Kartu-kartu yang tidak memiliki angka di dalamnya, akan diubah menjadi angka tertentu sesuai dengan kesepakatan yang berlaku antar pemain. Umumnya, kartu As akan bernilai 1, kartu Jack akan bernilai 11, kartu Queen akan bernilai 12, kartu King akan bernilai 13, dan kartu Joker tidak digunakan. Terkadang terdapat peraturan yang menyatakan bahwa kartu As dapat bernilai 1 atau 11 sehingga para pemain dapat memilih untuk menggunakan 1 atau 11 ketika mencari solusi permainan.

Permainan kartu 24 dapat memiliki satu atau lebih solusi yang berbeda, serta tidak memiliki solusi sama sekali. Sebagai contoh, kombinasi kartu 6, 4, 3, As (1) hanya memiliki satu buah solusi yaitu 6 / (1 - (3 / 4)). Kombinasi kartu 8, 3, 3, 3 memiliki banyak solusi, seperti 8 * ((3 - 3) + 3), (8 * 3) - (3 - 3), dan 8 * (3 - (3 - 3)). Kombinasi kartu 10, 10, 10, 10 tidak memiliki solusi sama sekali.

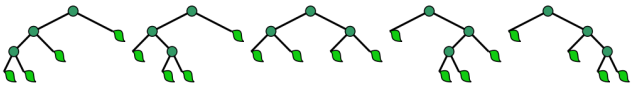
C. *Catalan Number*

Catalan Number atau bilangan *Catalan* merupakan deret angka yang muncul dalam berbagai persoalan kombinatorik dan sering melibatkan objek yang ditentukan secara rekursif. Nama *Catalan* berasal dari ahli matematika dari Belgia yang bernama Eugène Charles Catalan. Deret bilangan *Catalan* mampu mengidentifikasi jumlah struktur tertentu yang dapat dibentuk dari sejumlah elemen. Secara umum, bilangan Catalan ke-n dapat dinyatakan dengan

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Bilangan *Catalan* dapat digambarkan menjadi pohon biner. Sebagai contoh, perhatikan Gambar 2.1., gambar tersebut merupakan pohon biner untuk bilangan *Catalan* dengan n = 3. Banyaknya variasi pada pohon biner tersebut berjumlah 5 yang didapat dari nilai C_3. Variasi tersebut telah mencakup seluruh kemungkinan yang dapat dibuat.

$$C_3 = \frac{1}{3+1}\binom{6}{3} = 5$$



Gambar 2.1. Pohon biner untuk C_3

Sumber: https://en.wikipedia.org/wiki/Catalan_number

Salah satu kegunaan bilangan *Catalan* adalah untuk menghitung jumlah ekspresi yang mengandung n pasang tanda kurung. Ekspresi dan penempatan tanda kurung tersebut dapat direpresentasikan dalam sebuah pohon biner. Setiap daun pada pohon biner tersebut merepresentasikan sebuah elemen tertentu (dalam kasus permainan kartu 24, daun akan merepresentasikan angka pada kartu), sedangkan setiap cabangnya akan merepresentasikan operator yang digunakan untuk memproses kedua elemen yang ada pada daunnya. Tanda kurung akan diletakkan pada setiap operasi yang ada (satu operasi ditandai dengan satu cabang dan kedua daunnya). Sebagai contoh, perhatikan Gambar 2.2., pohon biner tersebut dapat ditulis menjadi a * (b + (c * d)). Tanda kurung tidak ditulis pada cabang paling atas atau *root* karena ekspresi tersebut akan melakukan operasi yang sama, baik ada tanda kurung maupun tidak.

Gambar 2.2. Ekspresi berbentuk pohon biner

Sumber: Dokumen penulis

III. Implementasi

A. *Algoritma Brute Force*

Pada makalah ini, aturan permainan kartu 24 yang digunakan adalah aturan umum dengan kartu As bernilai 1, kartu Jack bernilai 11, kartu Queen bernilai 12, dan kartu King bernilai 13.

a. Jumlah Kemungkinan Solusi

Algoritma *Brute Force* akan diimplementasikan dengan menggunakan *looping* yang melakukan iterasi kepada seluruh kombinasi kartu yang ada. Jumlah kombinasi kartu

tersebut dapat ditentukan dengan menggunakan permutasi. Banyaknya kartu yang digunakan pada permainan ini adalah 4. Namun, tidak semua kartu tersebut berbeda. Kombinasi kartu tersebut memiliki kemungkinan 2 kartu bernilai sama, 3 kartu bernilai sama, dan 4 kartu bernilai sama sehingga

- Jumlah kombinasi kartu jika semua nilainya berbeda

$$P(4; 1, 1, 1, 1) = \frac{4!}{1!1!1!1!} = 24$$

- Jumlah kombinasi kartu jika 2 kartu bernilai sama

$$P(4; 1, 1, 2) = \frac{4!}{1!1!2!} = 12$$

$$P(4; 2, 2) = \frac{4!}{2!2!} = 6$$

- Jumlah kombinasi kartu jika 3 kartu bernilai sama

$$P(4; 1, 3) = \frac{4!}{1!3!} = 4$$

- Jumlah kombinasi kartu jika 4 kartu bernilai sama

$$P(4; 4) = \frac{4!}{4!} = 1$$

Selain kartu, ada juga kombinasi operator yang digunakan. Operator aritmatika terdiri dari penjumlahan (+), pengurangan (-), perkalian (*), dan pembagian (/). Namun, banyaknya operator yang dapat digunakan dalam 1 operasi adalah 3. Hal tersebut dibuktikan oleh persamaan berikut (dengan a, b, c, d sebagai angka dan # sebagai operator)

$$a \# b \# c \# d$$

Setiap # memiliki 4 kemungkinan operator yang dapat digunakan sehingga

4	4	4

$$Jumlah\ kombinasi\ operator = 4 * 4 * 4 = 64$$

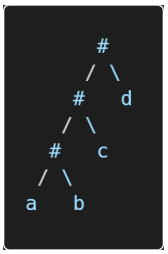
Banyaknya kemungkinan operasi yang dilakukan oleh algoritma *Brute Force* bergantung pada nilai pada setiap kartu sehingga jumlah operasi yang dilakukannya berjumlah

- Semua nilainya berbeda: $24 * 64 = 1536$
- Dua kartu bernilai sama:
  $12 * 64 = 768$
  $6 * 64 = 384$
- Tiga kartu bernilai sama: $4 * 64 = 256$
- Empat kartu bernilai sama: $1 * 64 = 64$

b. Implementasi *Brute Force*

Algoritma *Brute Force*, yang diimplementasi pada makalah ini, menggabungkan *for looping* dengan suatu pola operasi tertentu. Pola tersebut diperoleh dari *Catalan Number*. Pada bagian sebelumnya, telah dijelaskan bahwa jumlah operator yang dapat digunakan dalam 1 operasi adalah 3 sehingga jumlah pola yang akan digunakan berjumlah 5. Jumlah tersebut diperoleh dari *Catalan Number* untuk n = 3 (C_3). Berikut merupakan pohon biner untuk C_3 yang menghasilkan pola-pola operasi

Gambar 3.1. Pola operasi 1
Sumber: Dokumen penulis



Gambar 3.2. Pola operasi 2
Sumber: Dokumen penulis

Gambar 3.3. Pola operasi 3
Sumber: Dokumen penulis

Gambar 3.4. Pola operasi 4
Sumber: Dokumen penulis

Gambar 3.5. Pola operasi 5
Sumber: Dokumen penulis

Dari pohon biner tersebut, diperoleh pola-pola sebagai berikut

- Gambar 3.1. : (a # b) # (c # d)
- Gambar 3.2. : ((a # b) # c) # d
- Gambar 3.3. : (a # (b # c)) # d
- Gambar 3.4. : a # ((b # c) # d)
- Gambar 3.5. : a # (b # (c # d)

Pola-pola tersebut akan digunakan pada algoritma *Brute Force* untuk mencari solusi dari permainan kartu 24.

B. Program

Penyelesaian permainan kartu 24 dengan algoritma *Brute Force* diimplementasikan menggunakan bahasa pemrograman Java dengan paradigma pemrograman berorientasi objek (PBO/OOP). Berikut merupakan cuplikan kodenya

```
1  public void solve24BruteForce() {
2          for (int i = 0; i < N; i++) {
3              for (int op1 = 0; op1 < 4; op1++) {
4                  for (int j = 0; j < N; j++) {
5                      if (j != i) {
6                          for (int op2 = 0; op2 < 4; op2++) {
7                              for (int k = 0; k < N; k++) {
8                                  if (k != i && k != j) {
9                                      for (int op3 = 0; op3 < 4; op3++) {
10                                         for (int l = 0; l < N; l++) {
11                                             if (l != k && l != i && l != j) {
12                                                 // (a # b) # (c # d)
13                                                 if (count(count((double)cardList.get(i),
   (double)cardList.get(j), op1), count((double)cardList.get(k), (double)cardList.get(l), op2), op3) == 24)
   {
14                                                     pathList.add(String.format("(%d %s %d) %s (%d %s
   %d)", cardList.get(i), operator.get(op1), cardList.get(j), operator.get(op3), cardList.get(k),
   operator.get(op2), cardList.get(l)));
15                                                 }
16
17                                                 // (a # (b # c)) # d
18                                                 if (count(count((double)cardList.get(i),
   count((double)cardList.get(j), (double)cardList.get(k), op2), op1), (double)cardList.get(l), op3) == 24)
   {
19                                                     pathList.add(String.format("(%d %s (%d %s %d)) %s
   %d", cardList.get(i), operator.get(op1), cardList.get(j), operator.get(op2), cardList.get(k),
   operator.get(op3), cardList.get(l)));
20                                                 }
21
22                                                 // ((a # b) # c) # d
23                                                 if (count(count(count((double)cardList.get(i),
   (double)cardList.get(j), op1), (double)cardList.get(k), op2), (double)cardList.get(l), op3) == 24) {
24                                                     pathList.add(String.format("((%d %s %d) %s %d) %s
   %d", cardList.get(i), operator.get(op1), cardList.get(j), operator.get(op2), cardList.get(k),
   operator.get(op3), cardList.get(l)));
25                                                 }
26
27                                                 // a # ((b # c) # d)
28                                                 if (count((double)cardList.get(i),
   count(count((double)cardList.get(j), (double)cardList.get(k), op2), (double)cardList.get(l), op3), op1)
   == 24) {
29                                                     pathList.add(String.format("%d %s ((%d %s %d) %s
   %d)", cardList.get(i), operator.get(op1), cardList.get(j), operator.get(op2), cardList.get(k),
   operator.get(op3), cardList.get(l)));
30                                                 }
31
32                                                 // a # (b # (c # d))
33                                                 if (count((double)cardList.get(i),
   count((double)cardList.get(j), count((double)cardList.get(k), (double)cardList.get(l), op3), op2), op1)
   == 24) {
34                                                     pathList.add(String.format("%d %s (%d %s (%d %s
   %d))", cardList.get(i), operator.get(op1), cardList.get(j), operator.get(op2), cardList.get(k),
   operator.get(op3), cardList.get(l)));
35                                                 }
36                                             }
37                                         }
38                                     }
39
40                                 }
41                             }
42                         }
43
44                     }
45                 }
46             }
47         }
48     }
```

Gambar 3.6. Kode program

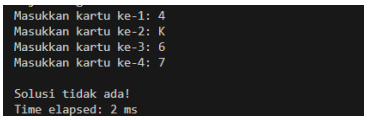
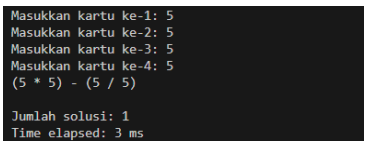
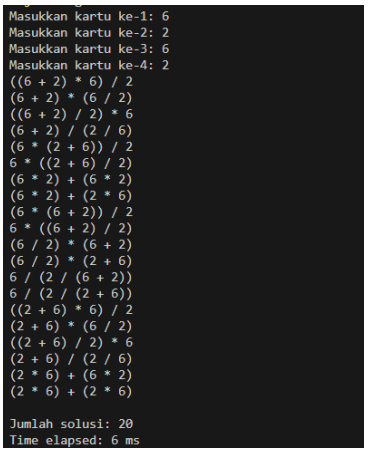
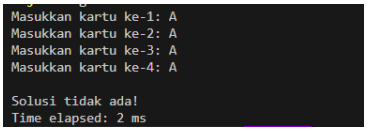
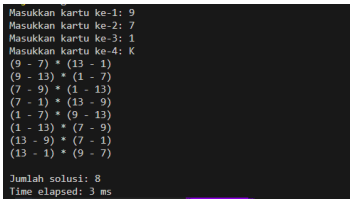
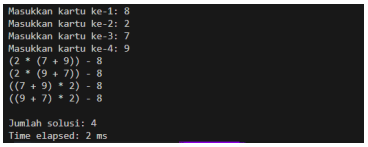
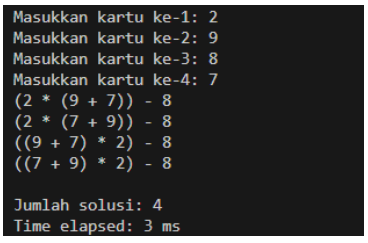
Sumber: Dokumen Penulis

Kode di atas merupakan bagian utama dari algoritma *Brute Force*. Kode tersebut akan melakukan iterasi kepada seluruh kombinasi kartu yang ada dan mengecek hasil operasinya. Jika hasil operasi bernilai 24, program akan menambahkan operasi tersebut ke dalam suatu list bernama pathList yang nantinya akan dijadikan *output* yang ditampilkan pada layar. Fungsi count() pada kode di atas, bertugas untuk melakukan operasi pertambahan, pengurangan, perkalian, dan pembagian. Pola-pola yang telah disebutkan sebelumnya, digunakan pada kode di atas yang dapat dilihat pada line 12, 17, 22, 27, dan 32. Hasil dari program ini tidak dipengaruhi oleh urutan bilangan pada *input*. Program akan menghasilkan *output* yang sama jika bilangan *input* sama, tetapi diacak.

IV. Analisis dan Pembahasan

Hasil pengujian dapat dilihat pada tabel berikut

Test Case	Input	Output
1	(J, 3, 8, 5)	Masukkan kartu ke-1: J Masukkan kartu ke-2: 8 Masukkan kartu ke-3: 3 Masukkan kartu ke-4: 5 (11 + (5 - 8)) * 3 ((11 + 5) - 8) * 3 (11 - 8) * (3 + 5) (11 - 8) * (5 + 3) ((11 - 8) + 5) * 3 (11 - (8 - 5)) * 3 (11 - (3 + 5)) * 8 (11 - 3) * (8 - 5) ((11 - 3) - 5) * 8 (11 - (5 + 3)) * 8 ((11 - 5) - 3) * 8 (8 - 5) * (11 - 3) 8 * (11 - (3 + 5)) 8 * ((11 - 3) - 5) 8 * (11 - (5 + 3)) 8 * ((11 - 5) - 3) (3 + 5) * (11 - 8) (3 - 11) * (5 - 8) 3 * ((11 + 5) - 8) 3 * (11 + (5 - 8)) 3 * ((11 - 8) + 5) 3 * (11 - (8 - 5)) 3 * ((5 + 11) - 8) 3 * (5 + (11 - 8)) 3 * ((5 - 8) + 11) 3 * (5 - (8 - 11)) (5 + (11 - 8)) * 3 ((5 + 11) - 8) * 3 (5 + 3) * (11 - 8) ((5 - 8) + 11) * 3 (5 - (8 - 11)) * 3 (5 - 8) * (3 - 11) Jumlah solusi: 32 Time elapsed: 12 ms
2	(3, 9, 3, 7)	Masukkan kartu ke-1: 3 Masukkan kartu ke-2: 9 Masukkan kartu ke-3: 3 Masukkan kartu ke-4: 7 3 + ((9 * 7) / 3) 3 + (9 * (7 / 3)) 3 + ((9 / 3) * 7) 3 + (9 / (3 / 7)) 3 + ((7 * 9) / 3) 3 + (7 * (9 / 3)) 3 + ((7 / 3) * 9) 3 + (7 / (3 / 9)) (3 - 9) * (3 - 7) (3 - 7) * (3 - 9) (3 * 7) + (9 / 3) (9 - 3) * (7 - 3) (9 * (7 / 3)) + 3 ((9 * 7) / 3) + 3 (9 / 3) + (3 * 7) (9 / 3) + (7 * 3) ((9 / 3) * 7) + 3 (9 / (3 / 7)) + 3 (7 - 3) * (9 - 3) (7 * 3) + (9 / 3) (7 * (9 / 3)) + 3 ((7 * 9) / 3) + 3 ((7 / 3) * 9) + 3 (7 / (3 / 9)) + 3 Jumlah solusi: 24 Time elapsed: 7 ms
3	(1, 5, 5, 5)	Masukkan kartu ke-1: 1 Masukkan kartu ke-2: 5 Masukkan kartu ke-3: 5 Masukkan kartu ke-4: 5 (5 - (1 / 5)) * 5 5 * (5 - (1 / 5)) Jumlah solusi: 2 Time elapsed: 5 ms
4	(K, 4, 7, 6)	Masukkan kartu ke-1: K Masukkan kartu ke-2: 4 Masukkan kartu ke-3: 7 Masukkan kartu ke-4: 6 Solusi tidak ada! Time elapsed: 2 ms

5	(4, K, 6, 7)	
6	(5, 5, 5, 5)	
7	(6, 2, 6, 2)	
8	(A, A, A, A)	
9	(9, 7, 1, K)	
10	(8, 2, 7, 9)	
11	(2, 9, 8, 7)	

Tabel 4.1. Tabel hasil pengujian
Sumber: Dokumen penulis

Berdasarkan hasil pengujian pada Tabel 4.1., algoritma *Brute Force* terbukti dapat menyelesaikan dan menemukan semua solusi untuk permainan kartu 24. Pada *test case* 1, dilakukan pengujian untuk 4 nilai yang berbeda dan diperoleh 32 solusi dengan waktu eksekusi 12 ms. Pada *test case* 2, dilakukan pengujian untuk 2 kartu bernilai sama dan sisanya berbeda, diperoleh 24 solusi dengan waktu eksekusi 7 ms. Pada *test case* 7, dilakukan pengujian untuk 2 kartu bernilai sama dan 2 lainnya juga bernilai sama, diperoleh solusi sebanyak 20 dan waktu eksekusi 6 ms. Dari ketiga *test case* tersebut, dapat dilihat bahwa waktu eksekusi untuk 2 kartu bernilai sama lebih cepat daripada 4 kartu berbeda. Hasil ini bisa dikatakan sejalan dengan teori yang telah disebutkan sebelumnya yaitu jumlah iterasi yang dilakukan oleh 2 kartu bernilai sama lebih sedikit daripada 4 kartu berbeda. Semakin banyak iterasi, semakin besar pula waktu yang diperlukan untuk menyelesaikannya, begitu pula sebaliknya. Namun, jika *test case* 2 dan *test case* 7 dibandingkan dengan *test case* 4 dan *test case* 9, waktu eksekusinya lebih lama. Hal ini bertentangan dengan teori sebelumnya. Ada beberapa alasan yang dapat menyebabkan hal ini terjadi. Pertama, kondisi perangkat yang berbeda ketika dijalankan. Kondisi perangkat tidak dapat sama persis ketika keempat *test case* tersebut dijalankan sehingga waktu eksekusi program juga akan terpengaruh. Kedua, *test case* 2 dan *test case* 7 harus melakukan penambahan elemen ke dalam list lebih banyak daripada *test case* 4 dan *test case* 9. Hal tersebut juga bisa menjadi salah satu penyebab *test case* 2 dan 7 lebih lama karena proses yang dilakukan lebih banyak.

Test case 4, 5, 10, dan 11 dilakukan untuk membuktikan bahwa urutan nilai kartu tidak akan mempengaruhi hasil yang didapatkan. *Test case* 4, 5, dan 8 dilakukan untuk membuktikan bahwa ada kombinasi kartu yang tidak memiliki solusi sama sekali. *Test case* 3 dan 6 dilakukan untuk membandingkan waktu eksekusi jika kartu yang bernilai sama ada 3 dan 4. Hasil yang didapatkan *test case* 6 lebih cepat daripada *test case* 3 dan hasil ini masih sesuai dengan teori yang telah disebutkan sebelumnya

V. KESIMPULAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat disimpulkan bahwa algoritma *Brute Force* dapat digunakan untuk memperoleh semua solusi pada permainan kartu 24. Waktu yang diperlukan oleh algoritma ini untuk mencari solusinya pun cukup cepat, sekitar 1-20 ms. Hal tersebut dikarenakan ukuran persoalan ini masih tergolong kecil, hanya 1536 iterasi pada kasus terburuknya. Oleh karena itu, dapat disimpulkan juga bahwa algoritma *Brute Force* tergolong layak untuk persoalan permainan kartu 24.

VI. PENUTUP

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan makalah ini dengan lancar. Penulis juga mengucapkan terima kasih kepada Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc., atas bimbingannya selama perkuliahan sehingga penulis dapat menyelesaikan perkuliahan dan makalah ini dengan baik. Selain itu, penulis juga ingin mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir

atas materi dan referensi yang telah disediakan pada situs beliau. Terakhir, penulis ingin mengucapkan terima kasih kepada orang tua, keluarga, teman, dan seluruh pihak yang terlibat dalam pembuatan Makalah IF2211 Strategi Algoritma ini. Penulis berharap makalah ini dapat memberikan manfaat bagi para pembaca.

Lampiran

Tautan repository:
https://github.com/MRafliRasyiidin/Makalah_Stima

References

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

[2] "Munir, Rinaldi. "Algoritma Brute Force Bagian 2, " URL: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf diakses pada 12 Juni 2024

[3] GeeksforGeeks. "Program for Nth Catalan Number," GeeksforGeeks. [Online].
URL: https://www.geeksforgeeks.org/program-nth-catalan-number/ diakses pada 12 Juni 2024

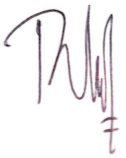
[4] Herawati, E. "Algoritma Brute Force," URL: https://repository.unikom.ac.id/37037/1/BruteForce%28bagian%201%29.pdf diakses pada 12 juni 2024

[5] CP-Algorithm. "Catalan Numbers - Algorithms for Competitive Programming," [Online].
URL: https://cp-algorithms.com/combinatorics/catalan-numbers.html 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



Muhamad Rafli Rasyiidin 13522088