# LAPORAN TUGAS KECIL I

# IF2211 STRATEGI ALGORITMA

# PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE

**Dosen Pengajar: Dr. Nur Ulfa Maulidevi, S.T, M.Sc.**


**Disusun oleh:**

**Kevin John Wesley Hutabarat (13521042)**


## SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

## INSTITUT TEKNOLOGI BANDUNG

## BANDUNG

## JANUARI 2022

# DAFTAR ISI

# DAFTAR GAMBAR

# BAB I

# TEORI DASAR

## 1. Brute Force Algorithm

Algoritma brute force (naive algorithm) adalah algoritma yang dilakukan dengan pendekatan yang lempang (straightforward) untuk menyelesaikan suatu persoalan. Algoritma ini digunakan untuk memecahkan persoalan secara langsung, sederhana, dan dengan cara yang jelas. Algoritma ini sering digunakan karena dapat digunakan untuk memecahkan hampir semua persoalan yang ada.

Algoritma brute force biasanya tidak cerdas karena membutuhkan langkah yang banyak untuk menyelesaikan sebuah persoalan, sehingga kurang mangkus dibanding dengan algoritma-algoritma yang lain. Dengan langkah yang banyak, membuat algoritma ini membutuhkan waktu dan memori yang lebih banyak. Untuk persoalan dengan luas cakupan yang kecil, biasanya brute force lebih disenangi karena lebih sederhana dan lebih mudah diimplementasikan. Meskipun begitu, terdapat juga persoalan yang lebih efisien, bahkan hanya dapat diselesaikan dengan algoritma brute force. Contohnya adalah persoalan mengurutkan elemen array, mencari elemen terbesar di array tidak terurut, dan sebagainya.

## 2. Permainan Kartu 24

Permainan kartu 24 adalah permainan yang dilakukan dengan cara mengambil empat kartu, kemudian dari empat kartu tersebut mencari kemungkinan ekspresi yang mungkin agar dapat menghasilkan angka 24. Kartu-kartu yang tersedia adalah 52 kartu remi dengan angka 2 sampai 10, dan kartu A, J, Q, K yang ekuivalen dengan angka 1, 11, 12, 13 secara berurutan dengan banyak kartu untuk masing-masing angka ada sebanyak empat kartu. Operator-operator yang dapat digunakan adalah +, -, *, dan /. Dalam permainan juga dapat menggunakan tanda kurung untuk menentukan angka apa yang akan terlebih dahulu dioperasikan.

## 3. Mencari Solusi Permainan Kartu 24 dengan Algoritma Brute Force

Solusi dari permainan ini dapat diperoleh dengan memanfaatkan algoritma brute force. Langkah-langkah mencari solusinya adalah:

i. Input 4 kartu yang ingin dicari solusinya dalam bentuk string 1 baris (bisa input pengguna langsung ataupun random).
ii. Mengambil kartu dengan cara split string, kemudian disimpan ke dalam vector of float.
iii. Mengubah tipe kartu dari string menjadi float agar dapat dihitung.
iv. Membuat permutasi yang mungkin dari keempat angka tersebut, kemudian semua kemungkinan disimpan di dalam vector of vector of float.
v. Membuat kombinasi 3 operator yang mungkin dari keempat operator yang tersedia, kemudian semua kemungkinan disimpan di dalam vector of vector of string.

vi. Mengevaluasi setiap angka dalam masing-masing hasil permutasi dengan operator hasil kombinasi, apabila hasilnya sama dengan 24, permutasi angka dan operator tersebut disimpan ke dalam vector of solution.
vii. Semua kemungkinan permutasi angka dan operator yang memenuhi sudah tersimpan di list of solution, kemudian program akan menampilkan semua solusi yang terdapat di list tersebut.

# BAB II

# IMPLEMENTASI

## 1. Fungsi dan Prosedur

Fungsi dan prosedur yang terdapat di dalam program, antara lain:

a. Fungsi self(): mendapatkan list kartu dari input pengguna.
b. Fungsi random(): mendapatkan list kartu secara random oleh program.
c. Fungsi validation(vector<string> kartu): menghasilkan boolean untuk mengecek kevalidan input kartu.
d. Fungsi split(string kartuawal): mendapatkan kartu dari satu line string untuk dimasukkan ke dalam list.
e. Fungsi convertToFloat(vector<string> kartu): mengubah tipe kartu dari string menjadi float agar bisa dihitung.
f. Fungsi acakOp (vector<string> listOp): menghasilkan kemungkinan permutasi operator yang mungkin.
g. Fungsi permutation (vector<float> listangka): menghasilkan kemungkinan permutasi angka kartu yang mungkin.
h. Fungsi isIn (vector<vector<float>> listbesar, vector<float> listangka): mengecek apakah listangka sudah terdapat di listbesar agar tidak ada permutasi dengan susunan angka sama.
i. Fungsi evaluate (vector<float> angka, vector<string> operasi): Mengevaluasi apakah dari angka dan operator dapat dibentuk angka 24. Jika memenuhi, akan disimpan ke dalam list.
j. Fungsi calculate(float bil1, float bil2, string op): menghitung nilai dengan angka dan operator tertentu.
k. Prosedur displaySolution (vector<solution> listSolusi): menampilkan solusi ke terminal.
l. Fungsi toString(solution listSolusi): mengubah solusi dari tipe solution menjadi string.
m. Fungsi toCard (int kartuint): mengubah kartu dalam bentuk integer menjadi string.

## 2. Source Code

Program ditulis dalam Bahasa C++. Source code program dapat diakses di link repository GitHub yang terdapat pada lampiran. Source code terdapat dalam file makeit24.cpp dalam folder src.

# BAB III

# PENGUJIAN

## 1. Tampilan awal program

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bin/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
```

Gambar 3.1. 1 Tampilan awal saat program dijalankan

## 2. Tampilan saat pengguna memilih kartu

```
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
1
7 3 6 Q
```

Gambar 3.2. 1 Program memilih kartu secara random

```
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
2
Masukkan kartu anda: K 4 5 6
```

Gambar 3.2. 2 kartu dipilih sendiri oleh pengguna

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bins/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
4
Masukan anda salah!
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
```

Gambar 3.2. 3 pengguna salah memasukkan input untuk memilih metode pengambilan kartu

```
Masukkan kartu anda: w r 4 3
Masukan salah! Silakan ulangi lagi
Masukkan kartu anda: 3 4 5 6 7
Masukan salah! Silakan ulangi lagi
Masukkan kartu anda:
```

Gambar 3.2. 4 Pengguna salah memasukkan input kartu

### 3. Contoh solusi yang ditampilkan program

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bin/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
2
Masukkan kartu anda: K 4 5 6
8 solutions found!
( K - ( 4 + 5 ) ) * 6
( ( K - 4 ) - 5 ) * 6
( K - ( 5 + 4 ) ) * 6
( ( K - 5 ) - 4 ) * 6
6 * ( K - ( 4 + 5 ) )
6 * ( ( K - 4 ) - 5 )
6 * ( K - ( 5 + 4 ) )
6 * ( ( K - 5 ) - 4 )

Execution time: 0.046605600 detik
```

Gambar 3.3. 1 Contoh solusi 1

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bin/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
2
Masukkan kartu anda: 5 5 5 5
1 solution found!
( 5 * 5 ) - ( 5 / 5 )

Execution time: 0.003436600 detik
```

Gambar 3.3. 2 Contoh solusi 2

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bin/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
2
Masukkan kartu anda: Q 7 7 4
0 solution found!

Execution time: 0.013127400 detik
```

Gambar 3.3. 3 Contoh solusi 3

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bin/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
1
8 9 2 3
12 solutions found!
8 * ( 9 - ( 2 * 3 ) )
8 * ( 9 - ( 3 * 2 ) )
( 8 * ( 9 - 3 ) ) / 2
8 * ( ( 9 - 3 ) / 2 )
( 8 / 2 ) * ( 9 - 3 )
8 / ( 2 / ( 9 - 3 ) )
( 9 - ( 2 * 3 ) ) * 8
( ( 9 - 3 ) * 8 ) / 2
( 9 - 3 ) * ( 8 / 2 )
( 9 - ( 3 * 2 ) ) * 8
( ( 9 - 3 ) / 2 ) * 8
( 9 - 3 ) / ( 2 / 8 )

Execution time: 0.046969400 detik
```

Gambar 3.3. 4 Contoh solusi 4

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bin/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
1
J J 6 9
0 solution found!

Execution time: 0.017117000 detik
Apakah anda ingin menyimpan solusi ini?
1. Ya
2. Tidak
```

Gambar 3.3. 5 Contoh solusi 5

```
PS C:\Users\User\Documents\GitHub\Tucil1_13521042\src> ./../bin/makeit24
Pilih cara anda memasukkan kartu
1. Random
2. Input sendiri
1
Q 6 6 4
22 solutions found!
Q + ( 6 * ( 6 - 4 ) )
Q + ( ( 6 - 4 ) * 6 )
Q - ( 6 * ( 4 - 6 ) )
( ( Q * 6 ) / 4 ) + 6
( Q * ( 6 / 4 ) ) + 6
Q - ( ( 4 - 6 ) * 6 )
( ( Q / 4 ) * 6 ) + 6
( Q / ( 4 / 6 ) ) + 6
6 + ( ( Q * 6 ) / 4 )
6 + ( Q * ( 6 / 4 ) )
6 + ( ( Q / 4 ) * 6 )
6 + ( Q / ( 4 / 6 ) )
( ( 6 * Q ) / 4 ) + 6
( 6 * ( Q / 4 ) ) + 6
6 + ( ( 6 * Q ) / 4 )
6 + ( 6 * ( Q / 4 ) )
6 + ( ( 6 / 4 ) * Q )
6 + ( 6 / ( 4 / Q ) )
( 6 * ( 6 - 4 ) ) + Q
( ( 6 / 4 ) * Q ) + 6
( 6 / ( 4 / Q ) ) + 6
( ( 6 - 4 ) * 6 ) + Q

Execution time: 0.1692048394 detik
Apakah anda ingin menyimpan solusi ini?
1. Ya
2. Tidak
```

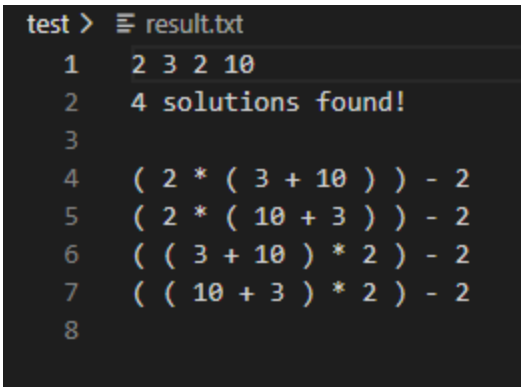
Gambar 3.3. 6 Contoh solusi 6

**4. Tampilan program ketika pengguna ingin menyimpan solusi ke dalam file**

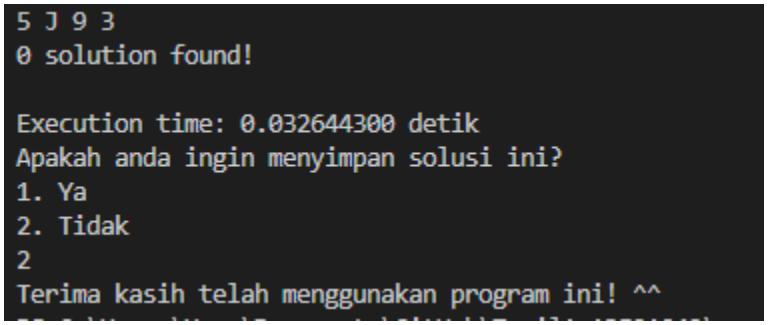
```
2 3 2 10
4 solutions found!
( 2 * ( 3 + 10 ) ) - 2
( 2 * ( 10 + 3 ) ) - 2
( ( 3 + 10 ) * 2 ) - 2
( ( 10 + 3 ) * 2 ) - 2

Execution time: 0.024774600 detik
Apakah anda ingin menyimpan solusi ini?
1. Ya
2. Tidak
1
Masukkan nama file: result.txt
Berhasil disimpan!
Terima kasih telah menggunakan program ini! ^^
```

Gambar 3.4. 1 Pengguna menyimpan solusi ke file




Gambar 3.4. 2 Contoh isi file berisi solusi




Gambar 3.4. 3 Pengguna tidak menyimpan solusi ke file

# LAMPIRAN

Link repository GitHub program:

https://github.com/kevinjohn01/Tucil1_13521042

| Poin | Ya | Tidak |
|---|---|---|
| 1. Program berhasil dikompilasi tanpa kesalahan | ✓ | |
| 2. Program berhasil running | ✓ | |
| 3. Program dapat membaca input / generate sendiri dan memberikan luaran | ✓ | |
| 4. Solusi yang diberikan program memenuhi (berhasil mencapai 24) | ✓ | |
| 5. Program dapat menyimpan solusi dalam file teks | ✓ | |