# LAPORAN TUGAS KECIL I

IF2211 STRATEGI ALGORITMA

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force


Disusun oleh:

Kartini Copa 13521026

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

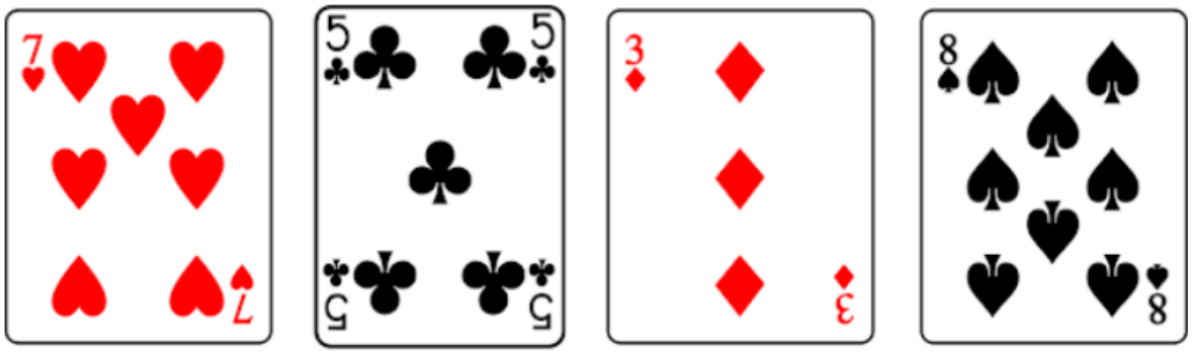
2022

# Daftar Isi

# BAB I

## Deskripsi Masalah

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. Permainan ini menarik cukup banyak peminat dikarenakan dapat meningkatkan kemampuan berhitung serta mengasah otak agar dapat berpikir dengan cepat dan akurat. Permainan Kartu 24 biasa dimainkan dengan menggunakan kartu remi. Kartu remi terdiri dari 52 kartu yang terbagi menjadi empat suit (sekop, hati, keriting, dan wajik) yang masing-masing terdiri dari 13 kartu (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). Yang perlu diperhatikan hanyalah nilai kartu yang didapat (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King). As bernilai 1, Jack bernilai 11, Queen bernilai 12, King bernilai 13, sedangkan kartu bilangan memiliki nilai dari bilangan itu sendiri. Pada awal permainan moderator atau salah satu pemain mengambil 4 kartu dari dek yang sudah dikocok secara random. Permainan berakhir ketika pemain berhasil menemukan solusi untuk membuat kumpulan nilainya menjadi 24. Pengubahan nilai tersebut dapat dilakukan menggunakan operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (×), divisi (/) dan tanda kurung ( () ). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas. (Paragraf di atas dikutip dari sini: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah2016/MakalahStima-2016-038.pdf).




Gambar 1. Permainan Kartu 24

# BAB II

## Algoritma Brute Force

Algoritma *brute force* adalah algoritma mencari semua kemungkinan solusi untuk memecahkan suatu masalah. Namun, algoritma ini biasanya bergantung pada kekuatan komputasi yang tinggi untuk mendapatkan semua solusi yang tepat daripada menggunakan teknik yang canggih sehingga sangat lambat dan tidak efisien untuk masalah atau kasus yang besar. Walaupun algoritma *brute force* memiliki kelemahan dalam efisiensi, namun algoritma ini memiliki keunggulan dapat menyelesaikan hampir semua permasalahan dengan tepat.

Berikut merupakan implementasi algoritma *brute force* dalam menyelesaikan permainan kartu 24.

- Mencari permutasi dari empat kartu.
- Mencari permutasi berulang dengan tiga elemen dari semua operator yang mungkin yaitu +, -, *, dan /.
- Membuat bentuk ekspresi matematika dari semua hasil permutasi bilangan dan operator serta menempatkan tanda kurung di posisi yang mungkin. Misalkan empat bilangan a, b, c, d, dan operator +, maka pada algoritma yang digunakan, tanda kurung disisipkan di tempat sebagai berikut.
    - (a+b)+(c+d)
    - ((a+b)+c)+d
    - (a+(b+c))+d
    - a+((b+c)+d)
    - a+(b+(c+d))
- Mengevaluasi hasil ekspresi matematika yang dibentuk dari bilangan, operator, dan tanda kurung.
- Jika hasil evaluasi adalah 24, simpan ekspresi matematika pada suatu *array*. Jika hasil evaluasi 24 maupun bukan 24, lanjutkan iterasi ke kemungkinan ekspresi matematika berikutnya hingga habis.
- Terakhir, setelah semua kemungkinan ekspresi matematika dievaluasi, tuliskan semua isi dari *array* beserta banyak isi *array* untuk menyatakan solusi yang ditemukan dan banyaknya.

# BAB III

## Source Program

Program penyelesaian kartu 24 ini ditulis dalam Bahasa C++ dengan menggunakan library sebagai berikut.

```
#include <iostream>
#include <string>
#include <time.h>
#include <vector>
#include <algorithm>
#include <random>
#include <chrono>
#include <functional>
#include <fstream>

using namespace std;
```

(Library yang digunakan)

```
int num[4];
int numPosition[4];
char sign[3], myOp[] = {'+','-','*','/'};
bool check;

bool isValid(string card) {
```

(Inisiasi awal)

```
bool isValid(string card) {
    // validate the input is A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K
    vector<string> cards = {"A", "J", "Q", "K", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
    for (int i = 0; i < cards.size(); i++) {
        if (card == cards[i]) {
            return true;
        }
    }
    return false;
}
```

(Validasi *input user*)

```
int translateInput(string card) {
    vector<string> cards = {"A", "J", "Q", "K", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
    vector<int> num = {1, 11, 12, 13, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    for (int i = 0; i < cards.size(); i++) {
        if (card == cards[i]) {
            return num[i];
        }
    }
}
```

(Konversi *string card to integer*)

```
int calculate(int x, int y, char z) {
    // x + y
    if(z == '+'){
        return x + y;
    }
    // x - y
    else if(z == '-'){
        return x - y;
    }
    // x * y
    else if(z == '*'){
        return x * y;
    }
    // x / y
    else if(z == '/'){
        if(y == 0)
            return 999;
        else if(x % y != 0)
            return 999;
        else return x/y;
    }

}
```

(Kalkulasi operator)

```
void operation(int* countAll) {
    if(calculate(calculate(calculate(numPosition[0], numPosition[1], sign[0]), numPosition[2], sign[1]), numPosition[3], sign[2]) == 24.0){
        check = true;
        cout<<"(("<<numPosition[0]<<sign[0]<<numPosition[1]<<')'<<sign[1]<<numPosition[2]<<')'<<sign[2]<<numPosition[3]<<" = 24"<<endl;
        (*countAll)++;
    }
    if(calculate(calculate(numPosition[0], numPosition[1], sign[0]), calculate(numPosition[2], numPosition[3], sign[2]), sign[1]) == 24.0){
        check = true;
        cout<<"("<<numPosition[0]<<sign[0]<<numPosition[1]<<')'<<sign[1]<<'('<<numPosition[2]<<sign[2]<<numPosition[3]<<") = 24"<<endl;
        (*countAll)++;
    }
    if(calculate(calculate(numPosition[0], calculate(numPosition[1], numPosition[2], sign[1]), sign[0]), numPosition[3], sign[2]) == 24.0){
        check = true;
        cout<<"("<<numPosition[0]<<sign[0]<<'('<<numPosition[1]<<sign[1]<<numPosition[2]<<"))"<<sign[2]<<numPosition[3]<<" = 24"<<endl;
        (*countAll)++;
    }
    if(calculate(numPosition[0], calculate(calculate(numPosition[1], numPosition[2], sign[1]), numPosition[3], sign[2]), sign[0]) == 24.0){
        // if check = true then countAll += 1
        check = true;
        cout<<numPosition[0]<<sign[0]<<"(("<<numPosition[1]<<sign[1]<<numPosition[2]<<')'<<sign[2]<<numPosition[3]<<") = 24"<<endl;
        (*countAll)++;
    }
    if(calculate(numPosition[0], calculate(numPosition[1], calculate(numPosition[2], numPosition[3], sign[2]), sign[1]), sign[0]) == 24.0){
        check = true;
        cout<<numPosition[0]<<sign[0]<<'('<<numPosition[1]<<sign[1]<<'('<<numPosition[2]<<sign[2]<<numPosition[3]<<")) = 24"<<endl;
        (*countAll)++;
    }
}
```

(Solusi persamaan)

```
void randomNumPosition()
{
    for(int a=0; a<4; a++)
        for(int b=0; b<4; b++)
        {
            if(b==a) continue;
            for(int c=0; c<4; c++)
            {
                if(c==a or c==b) continue;
                for(int d=0; d<4; d++)
                {
                    if(d==a or d==b or d==c) continue;

                    numPosition[0] = num[a];
                    numPosition[1] = num[b];
                    numPosition[2] = num[c];
                    numPosition[3] = num[d];

                    operation();
                }
            }
        }
}
```

(Permutasi angka)

```
void signOperation() {
    for(int a=0; a<4; a++)
        for(int b=0; b<4; b++)
            for(int c=0; c<4; c++)
            {
                sign[0] = myOp[a];
                sign[1] = myOp[b];
                sign[2] = myOp[c];

                randomNumPosition();
            }
}
```

(Permutasi operator)

```
int main()
{
    // 3 types keyboard, file, random
    int total;
    char choose;
    cout << "Keyboard (k), File (f), Random (r): ";
    cin >> choose;
    if (choose == 'k') { // Menerima string A, 2, 3, 4, 5, 6, 7, 8, 9, 10
        string card1, card2, card3, card4;
        cout << "Card 1: ";
        cin >> card1;
        cout << "Card 2: ";
        cin >> card2;
        cout << "Card 3: ";
        cin >> card3;
        cout << "Card 4: ";
        cin >> card4;
        // validasi input with boolean isValid
        if (isValid(card1) && isValid(card2) && isValid(card3) && isValid(card4)) {
            // translate card to integer
            num[0] = translateInput(card1);
            num[1] = translateInput(card2);
            num[2] = translateInput(card3);
            num[3] = translateInput(card4);

        } else {
            cout << "Invalid Input" << endl;
            return 0;
        }
    }
```

(Fungsi *main* bagian *input keyboard*)

```
else if (choose == 'f') { // Menerima file
    string filename;
    cout << "Filename: ";
    cin >> filename;
    ifstream file(filename);
    string card1, card2, card3, card4;
    file >> card1 >> card2 >> card3 >> card4;
    num[0] = translateInput(card1);
    num[1] = translateInput(card2);
    num[2] = translateInput(card3);
    num[3] = translateInput(card4);

}
else if (choose == 'r') { // Random A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K using vector
    int i;
    vector<string> cards = {"A", "J", "Q", "K", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
    unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
    std::mt19937_64 rng(seed);
    shuffle(cards.begin(), cards.end(), rng);
    cout << "Random cards: " << cards[0] << " " << cards[1] << " " << cards[2] << " " << cards[3] << endl;
    num[0] = translateInput(cards[0]);
    num[1] = translateInput(cards[1]);
    num[2] = translateInput(cards[2]);
    num[3] = translateInput(cards[3]);
}
else {
    cout << "Invalid Input" << endl;
    return 0;
}
```

(Fungsi *main* bagian *input from file*)

```
else if (choose == 'r') { // Random A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K using vector
    int i;
    vector<string> cards = {"A", "J", "Q", "K", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
    unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
    std::mt19937_64 rng(seed);
    shuffle(cards.begin(), cards.end(), rng);
    cout << "Random cards: " << cards[0] << " " << cards[1] << " " << cards[2] << " " << cards[3] << endl;
    num[0] = translateInput(cards[0]);
    num[1] = translateInput(cards[1]);
    num[2] = translateInput(cards[2]);
    num[3] = translateInput(cards[3]);
}
else {
    cout << "Invalid Input" << endl;
    return 0;
}
```

(Fungsi *main* bagian *generate random*)

```
    // start timer
    auto start = chrono::high_resolution_clock::now();
   // count all solution
    int countAll = 0;
    signOperation(&countAll);
    // if countAll == 0
    if (countAll == 0) {
        cout << "No solution" << endl;
    }
    else {
        cout << "Total solution: " << countAll << endl;
    }

    auto finish = chrono::high_resolution_clock::now();
    chrono::duration<double> elapsed = finish - start;
    cout << "Elapsed time: " << elapsed.count() << endl;
    // want to save output?
    char save;
    cout << "Save output? (y/n): ";
    cin >> save;
    if (save == 'y') {
        string filename;
        cout << "Filename: ";
        cin >> filename;
        ofstream file(filename);
        file << "Card 1: " << num[0] << endl;
        file << "Card 2: " << num[1] << endl;
        file << "Card 3: " << num[2] << endl;
        file << "Card 4: " << num[3] << endl;
        // save operaation
        file << "Total solution: " << countAll << endl;
        file << "Elapsed time: " << elapsed.count() << endl;
    }
    return 0;
}
```

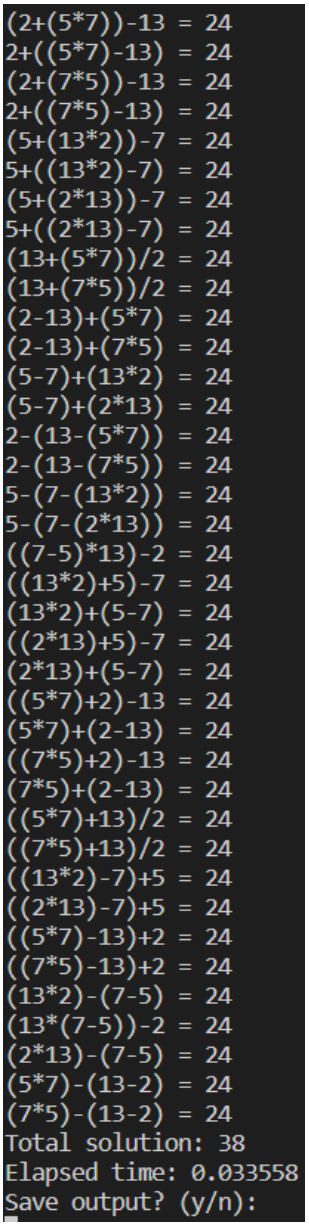
(Fungsi *main*)

# BAB IV

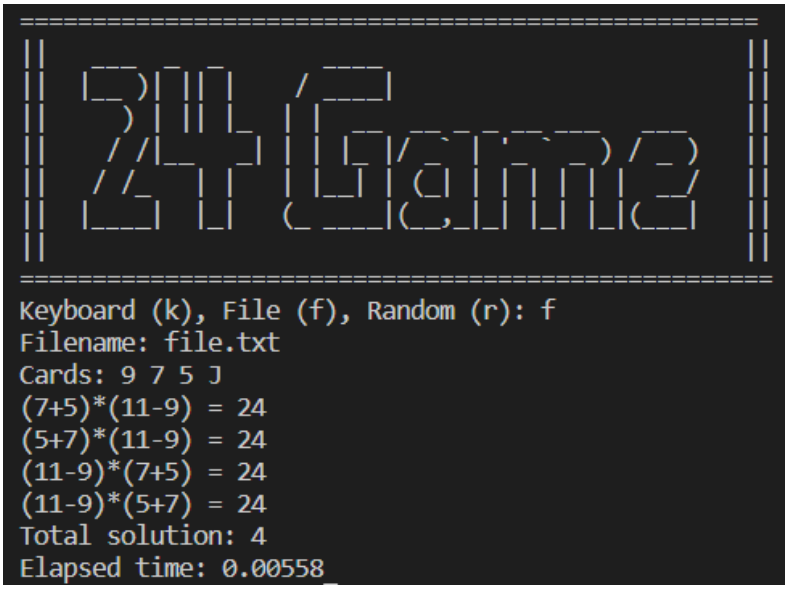
## Testing

### *Input* dan *output*

*Input from keyboard*: K, 2, 5, 7

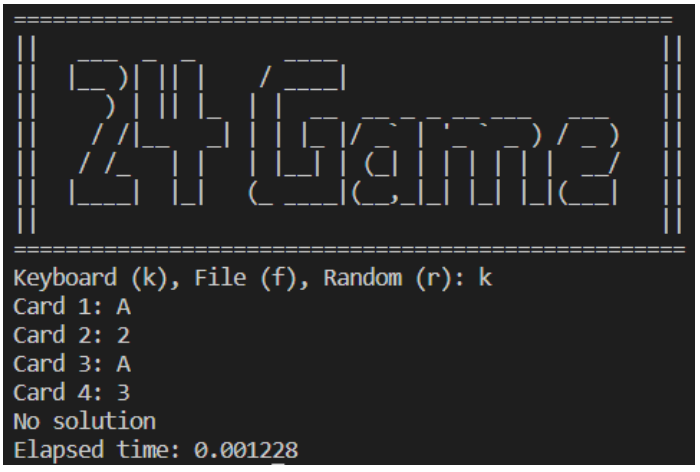
*Input from file* : 9 7 5 J




*Input from generate random*: 10, Q, 8, 7

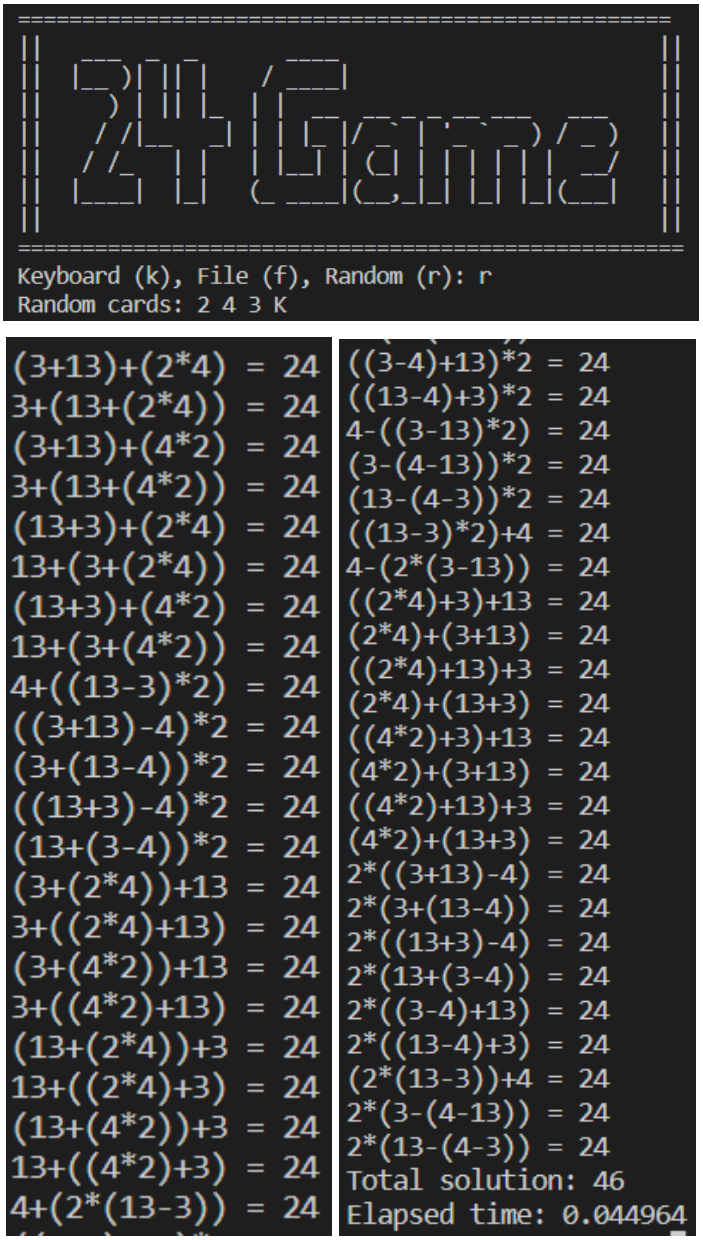

(Solusi tidak ditemukan)

*Input from keyboard*: A, 2, A, 3

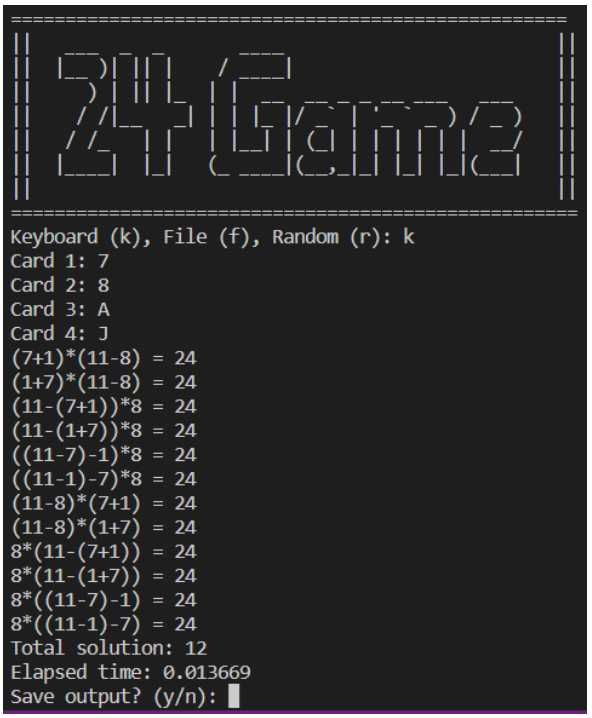


*Input from file*: K, 7, 4, 2

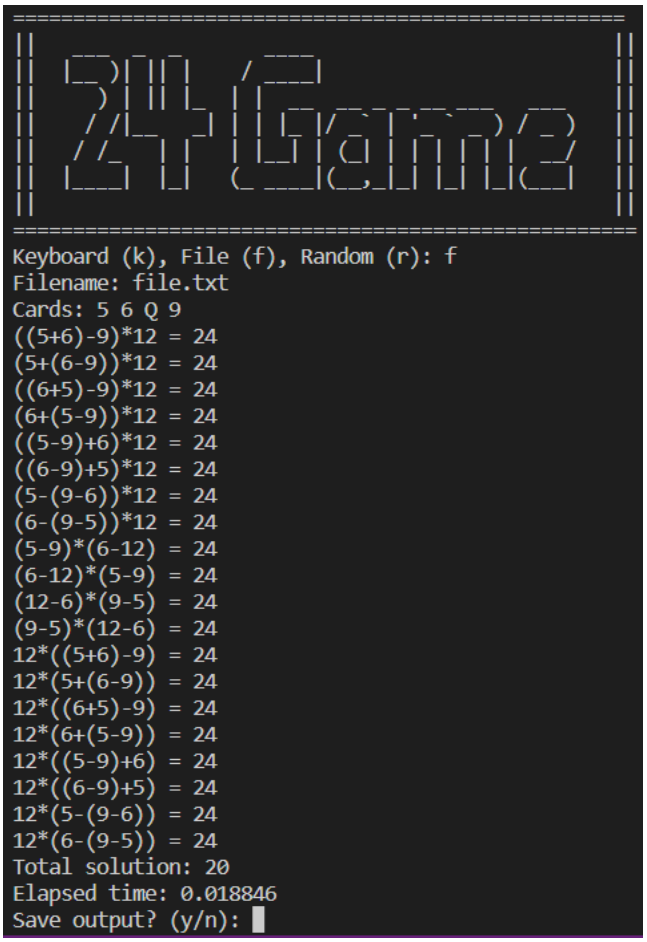

*Input from generate random*: 2, 4, 3, K



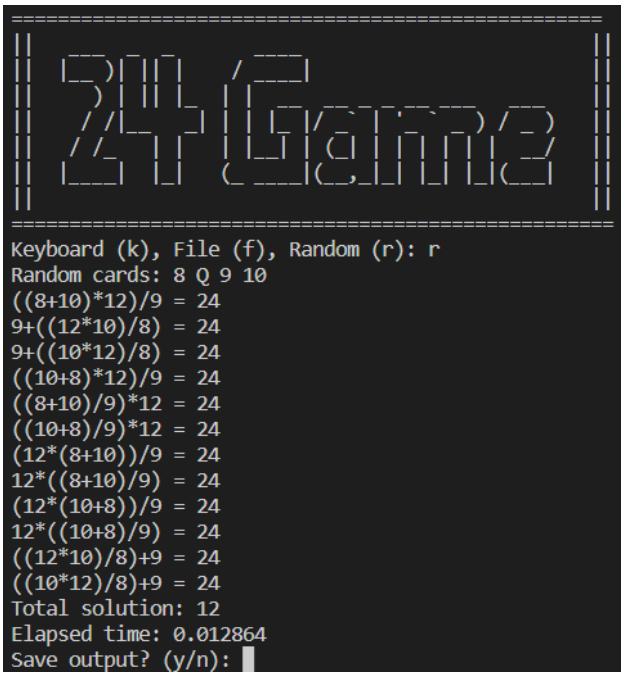
*Input from keyboard*: 7, 8, A, J




*Input from file*: 5, 6, Q, 9



*Input from generate random*



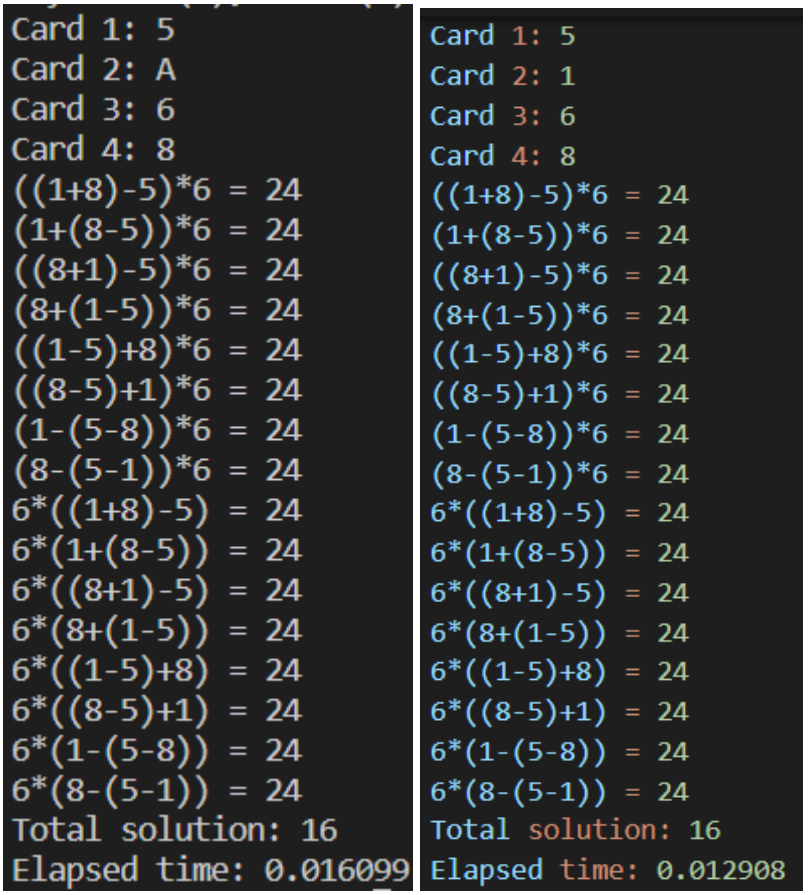

**Validasi *input***

Validasi input


Validasi *input* kartu

***SAVE***

Menyimpan solusi ke dalam file



```
Card 1: 5
Card 2: A
Card 3: 6
Card 4: 8
((1+8)-5)*6 = 24
(1+(8-5))*6 = 24
((8+1)-5)*6 = 24
(8+(1-5))*6 = 24
((1-5)+8)*6 = 24
((8-5)+1)*6 = 24
(1-(5-8))*6 = 24
(8-(5-1))*6 = 24
6*((1+8)-5) = 24
6*(1+(8-5)) = 24
6*((8+1)-5) = 24
6*(8+(1-5)) = 24
6*((1-5)+8) = 24
6*((8-5)+1) = 24
6*(1-(5-8)) = 24
6*(8-(5-1)) = 24
Total solution: 16
Elapsed time: 0.016099
```

```
Card 1: 5
Card 2: 1
Card 3: 6
Card 4: 8
((1+8)-5)*6 = 24
(1+(8-5))*6 = 24
((8+1)-5)*6 = 24
(8+(1-5))*6 = 24
((1-5)+8)*6 = 24
((8-5)+1)*6 = 24
(1-(5-8))*6 = 24
(8-(5-1))*6 = 24
6*((1+8)-5) = 24
6*(1+(8-5)) = 24
6*((8+1)-5) = 24
6*(8+(1-5)) = 24
6*((1-5)+8) = 24
6*((8-5)+1) = 24
6*(1-(5-8)) = 24
6*(8-(5-1)) = 24
Total solution: 16
Elapsed time: 0.012908
```

# BAB V

## Kesimpulan

Algoritma *brute force* dapat digunakan untuk menyelesaikan permainan kartu 24. Algoritma *brute force* dapat mencari operasi permutasi yang diperlukan dalam menyelesaikan perhitungan.

## Saran

Solusi dari program permainan kartu 24 ini  mungkin berbeda tetapi algoritma *brute force* yang digunakan pada permainan ini dapat diimplementasikan pada program lain yang serupa.

# REFERENSI

[1] R. Munir (2022). Algoritma Brute Force Bagian 1 [Powerpoint Slides]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf

# LAMPIRAN

## Link Github

https://github.com/kartinicopa/Tucil1_13521026.git

## Checklist

| Poin | Ya | Tidak |
|---|---|---|
| Program dapat dikompilasi tanpa kesalahan | ✓ | |
| Program berhasil *running* | ✓ | |
| Program dapat membaca *input* | ✓ | |
| Solusi yang diberikan program memenuhi (berhasil mencapai 24) | ✓ | |
| Program dapat membaca *input* / *generate* sendiri dan memberikan luaran dalam file teks | ✓ | |