

# **TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA**

**SEMESTER 2 TAHUN 2022/2023**

**PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE**



**Disusun Oleh:**

**Jericho Russel Sebastian**

**NIM 13521107**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2023**

## DAFTAR ISI

<b>I. Latar Belakang.....</b>	<b>3</b>
<b>II. Pembahasan.....</b>	<b>3</b>
2.1 Enumerasi .....	4
2.2 Evaluasi.....	5
2.3 Akhir Pencarian .....	5
<b>III. Implementasi .....</b>	<b>5</b>
3.1 Operator.java.....	6
3.2 Solution.java .....	6
3.3 Perm.java .....	7
3.4 Solver.java .....	8
3.5 Convert.java .....	9
3.6 Rand.java .....	10
3.7 Game24.java .....	10
<b>IV. Hasil Eksekusi .....</b>	<b>13</b>
4.1 Tampilan Awal.....	13
4.2 Tampilan Solusi Input Valid.....	14
4.3 Tampilan Solusi Input Acak .....	14
4.4 Tampilan Input Tidak Valid.....	14
4.5 Tampilan Menyimpan ke File .....	15
4.6 Tampilan Gagal Menyimpan File .....	16
4.7 Tampilan Tidak Menyimpan ke File.....	16
<b>V. Lampiran .....</b>	<b>17</b>
5.1 Repository GitHub .....	17
5.2 Tabel Penilaian.....	17
<b>VI. Daftar Pustaka .....</b>	<b>17</b>

## I. Latar Belakang

Permainan kartu 24 (*24 card game*) merupakan sebuah permainan yang melibatkan kartu remi dan operasi aritmatika terhadap nilai kartu tersebut. Permainan ini berlangsung sebagai berikut: diberikan 4 buah kartu remi acak boleh berulang, pemain berusaha menemukan kombinasi operasi dasar aritmatika (penjumlahan, pengurangan, perkalian, dan/atau pembagian) dan tanda kurung yang menyebabkan nilai operasi terhadap keempat kartu tersebut adalah 24, dengan syarat setiap kartu digunakan tepat satu kali dalam operasi tersebut dalam urutan bebas. Nilai dari sebuah kartu merupakan nilai angka yang tertera pada kartu tersebut, dengan pengecualian A bernilai 1, J bernilai 11, Q bernilai 12, dan K bernilai 13; *suit* kartu diabaikan. Sebagai contoh: diberikan 4 buah kartu dengan nilai berturut-turut 4, 6, Q, dan A, maka dua solusi yang mungkin adalah  $(4 - (Q - 6)) \times A$  dan  $(Q - A) \times (6 - 4)$ . Tidak semua kombinasi 4 kartu memiliki solusi dalam permainan kartu 24; J, Q, K, dan 3 merupakan salah satu kombinasi 4 kartu yang tidak dapat dioperasikan dengan operasi dasar sehingga bernilai 24.

Algoritma *brute force* merupakan teknik pemecahan persoalan yang didasari pada pendekatan lempang (*straightforward*) terhadap persoalan yang dihadapi. Pemecahan persoalan secara *brute force* dilakukan secara langsung dan sederhana untuk diimplementasikan. Algoritma *brute force* juga dapat digunakan untuk memecahkan hampir semua persoalan. Namun, kekurangan terbesar dari pendekatan secara *brute force* adalah ketidaksangkilan algoritma yang dihasilkan.

Dengan menggunakan pendekatan secara *brute force*, penulis menyusun Game24, sebuah program sederhana untuk menemukan semua solusi dari suatu permainan kartu 24. Game24 dapat menerima masukan 4 buah kartu atau membangkitkan sebuah susunan 4 kartu secara acak, dan mengembalikan semua solusi permainan kartu 24 serta menyimpan solusi tersebut dalam sebuah file.

## II. Pembahasan

Teknik utama yang digunakan oleh Game24 adalah *exhaustive search*, suatu teknik pencarian solusi secara *brute force* untuk persoalan-persoalan kombinatorik. Teknik ini terdiri dari 3 tahap:

1. Enumerasi setiap kemungkinan solusi secara sistematis.

2. Evaluasi setiap kemungkinan solusi satu per satu, dan simpan solusi terbaik yang ditemukan sejauh ini.
3. Pada akhir pencarian, umumkan solusi terbaik.

Dengan menggunakan teknik ini, program akan selalu menemukan semua solusi untuk sembarang kombinasi 4 buah kartu dalam permainan kartu 24, jika solusi ada. Program juga dipastikan berhenti karena banyak kemungkinan solusi terhingga. Berikut merupakan deskripsi dari setiap tahap *exhaustive search* yang digunakan dalam program.

## 2.1 Enumerasi

Tahap enumerasi membangkitkan sebuah *list* berisi semua kemungkinan solusi dari instansi permainan kartu 24 yang diberikan. Setiap kemungkinan solusi ini berupa suatu kombinasi 4 buah kartu dan 3 buah operator dasar aritmatika dengan suatu konfigurasi tanda kurung. Untuk membangkitkan semua kemungkinan solusi secara sistematis, digunakan algoritma berikut:

1. Enumerasi setiap permutasi dari 4 buah kartu yang diberikan. Terdapat maksimum  $4! = 24$  permutasi unik dari sembarang 4 buah kartu.
2. Enumerasi setiap kumpulan 3 operator dasar aritmatika. Terdapat  $4^3 = 64$  kumpulan unik dari 3 operator yang dapat digunakan dalam permainan.
3. Enumerasi setiap konfigurasi tanda kurung. Terdapat 5 konfigurasi tanda kurung yang dapat menghasilkan nilai unik, yaitu:

$$(a * b) * (c * d)$$

$$((a * b) * c) * d$$

$$(a * (b * c)) * d$$

$$a * ((b * c) * d)$$

$$a * (b * (c * d))$$

4. Sebuah kemungkinan solusi dibentuk dari sebuah permutasi 4 buah kartu, sebuah kumpulan 3 operator, dan satu konfigurasi tanda kurung. Terdapat maksimum  $24 \times 64 \times 5 = 7680$  kemungkinan solusi yang akan dienumerasi oleh program.

## 2.2 Evaluasi

Pada tahap evaluasi, setiap kemungkinan solusi diperiksa satu per satu. Jika suatu kemungkinan solusi bernilai 24, program menyimpan solusi tersebut dalam suatu *list* hasil. Sebaliknya, solusi diabaikan.

## 2.3 Akhir Pencarian

Ketika program mencapai tahap ini, maka *list* hasil berisi semua solusi dari instansi permainan kartu 24 yang diberikan. Isi dari *list* hasil ditampilkan kepada pengguna sebagai solusi-solusi yang dapat digunakan untuk menyelesaikan permainan.

## III. Implementasi

Game24 ditulis dalam bahasa Java, dengan sistematika file sebagai berikut:

```
Tucil1_13521107
|-- bin
|   |-- classes
|   |   |-- ...
|   |-- libs
|       |-- Game24.jar
|-- doc
|   |-- Tucil1_K1_13521107_Jericho Russel Sebastian.pdf
|-- src
|   |-- game24
|       |-- struct
|       |   |-- Operator.java
|       |   |-- Solution.java
|       |-- util
|       |   |-- Convert.java
|       |   |-- Perm.java
|       |   |-- Rand.java
|       |   |-- Solver.java
|       |-- Game24.java
|-- test
|   |-- hasil_1.txt
|-- build.bat
```

```
|-- run.bat
|-- README.md
```

Penulis tidak akan menampilkan keseluruhan isi *source code* program, melainkan hanya potongan-potongan yang relevan dengan alur kerja program dan algoritma *exhaustive search*.

### 3.1 Operator.java

Operator.java merupakan file yang mendefinisikan tipe data Operator, yaitu tipe data yang mewakili keempat operator dasar aritmatika yang dapat digunakan dalam permainan. Operator merupakan tipe data enumerasi dengan 4 nilai, yaitu `plus`, `min`, `mult`, dan `div`, disertai dengan sebuah fungsi `eval(float, float)` untuk menghitung nilai sebuah operasi aritmatika.

```
public enum Operator{
    plus, min, mult, div;

    public static Operator[] all = {plus, min, mult, div};

    public float eval(float a, float b){
        // Mengembalikan hasil operasi a dengan b
        ...
    }
}
```

### 3.2 Solution.java

Solution.java mengandung definisi tipe data Solution, yaitu tipe data yang mewakili kemungkinan solusi dari permainan kartu 24. Sebuah instansi Solution memuat 4 buah *float* mewakili nilai keempat kartu dalam urutan tertentu, 3 buah Operator, dan sebuah *byte* yang memuat informasi tentang konfigurasi tanda kurung dari solusi tersebut, disertai dengan sebuah fungsi `eval()` untuk mengevaluasi nilai solusi ini.

```
public class Solution{
    float a, b, c, d;
    Operator op1, op2, op3;
    byte parentheses; // Bernilai 1-5 mewakili konfigurasi kurung
```

```

/* KONSTRUKTOR */
public Solution(float[] vals, Operator[] ops, byte p){...}

public float eval(){
    // Mengembalikan nilai evaluasi solusi ini
    ...
}
}

```

### 3.3 Perm.java

Perm.java memuat kelas Perm, sebuah kelas pembantu yang menyediakan fungsionalitas pembangkitan susunan dan permutasi.

```

public class Perm{
    static ArrayList<float[]> cards = new ArrayList<>(24);
    static Operator[][] ops = null;

    public static ArrayList<float> permCard4(float[] vals){
        // Membangkitkan semua permutasi 4 buah kartu

        cards.clear();
        for(int i = 1; i <= 4; i++){
            for(int j = 1; j <= 4; j++){
                if(i == j)continue;
                for(int k = 1; k <= 4; k++){
                    if(i == k || j == k)continue;

                    // Buat sebuah permutasi 4 kartu
                    float temp4[] = new float[]{
                        vals[i-1],
                        vals[j-1],
                        vals[k-1],
                        vals[24/(i*j*k)-1]
                    };

                    // Pastikan tidak ada permutasi duplikat
                    boolean unique = true;
                    for(float[] curr : cards){

```

```

        if(curr[0] == temp4[0] && curr[1] == temp4[1]
&& curr[2] == temp4[2] && curr[3] == temp4[3]){
            unique = false;
            break;
        }
    }
    if(unique) cards.add(temp4);
}
}
return cards;
}

public static Operator[][] permOp3(){
    // Membangkitkan semua urutan 3 operator

    if(ops == null){
        ops = new Operator[64][];

        int i = 0;
        for(Operator op1 : Operator.all)
            for(Operator op2 : Operator.all)
                for(Operator op3 : Operator.all)
                    ops[i++] = new Operator[]{op1, op2, op3};
    }
    return ops;
}
}

```

### 3.4 Solver.java

Solver.java memuat kelas Solver, sebuah kelas pembantu yang menjadi penggerak utama algoritma *exhaustive search*. Solver berisi sebuah fungsi yang mengenumerasi semua kemungkinan solusi, melakukan evaluasi satu per satu, dan menyimpan semua solusi yang ditemukan.

```

public class Solver{
    static ArrayList<Solution> solutions = new ArrayList<>();

```



```

public static ArrayList<Solution> solve(float[] inps){
    // Mencari semua solusi menggunakan exhaustive search

    solutions.clear();

    // ENUMERASI
    for(float[] vals : Perm.permCard4(inps)){
        for(Operator[] ops : Perm.permOp3()){
            for(byte p = 1; p <= 5; p++){
                Solution current = new Solution(vals, ops, p);

                // EVALUASI
                if(current.eval() == 24f)solutions.add(current);
            }
        }
    }
    // AKHIR PENCARIAN
    return solutions;
}
}

```

### 3.5 Convert.java

Convert.java memuat kelas Convert yang berisi berbagai fungsi konversi antara kartu (yang direpresentasikan dengan *string*) dengan nilai kartu (yang direpresentasikan dengan bilangan *float*).

```

public class Convert{
    public static float[] card4ToValue4(String[] cards){
        // Mengembalikan array berisi nilai keempat kartu
        ...
    }

    public static float cardToValue(String card){
        // Mengembalikan nilai dari satu kartu
        ...
    }
}

```

```

    public static String valueToCard(float val){
        // Mengembalikan nama kartu yang bernilai val
        ...
    }
}

```

### 3.6 Rand.java

Rand.java memuat kelas Rand yang menyediakan fungsionalitas pembangkitan 4 kartu acak.

```

public class Rand{
    static Random rand = new Random();
    static String[] tempCard4 = new String[4];

    public static String[] generateCard4(){
        // Membangkitkan array 4 kartu acak

        for(int i = 0; i < 4; i++){
            int val = rand.nextInt(13) + 1;
            tempCard4[i] = Convert.valueToCard(val);
        }
        return tempCard4;
    }
}

```

### 3.7 Game24.java

Game24.java memuat kelas Game24, yaitu kelas yang berfungsi sebagai program utama dan menjadi *entry point* ketika program dijalankan. Selain memuat prosedur `main`, kelas Game24 juga memuat beberapa fungsi pembantu untuk berkomunikasi dengan kelas Solver, menampilkan hasil pencarian, dan menyimpan solusi ke file.

```

public class Game24{
    static Scanner in;
    static ArrayList<Solution> solutions = null;
    static long elapsedNano;

    public static void main(String[] args){

```

```

// PROGRAM UTAMA

in = new Scanner(System.in);
...

boolean valid;
do{
    // Menerima dan memvalidasi input
    valid = true;
    ...
    String inpRaw = in.nextLine().trim().toUpperCase();

    String[] inps;
    float[] inpVal;
    if(!inpRaw.equals("R")){
        // Split input dan ubah menjadi array nilai kartu
        inps = inpRaw.split("\\s+");
        inpVal = Convert.card4ToValue4();

        if(inpVal != null){
            // Input valid, proses
            startSolve(inpVal);
        }else{
            // Input tidak valid, ulangi
            valid = false;
            ...
        }
    }else{
        // Membangkitkan input random
        inps = Rand.generateCard4();
        ...
        inpVal = Convert.card4ToValue4();
        startSolve(inpVal);
    }
}while(!valid);

// Tampilkan hasil dan simpan ke file
displaySolutions();

```

```

        promptSaveResults();
        ...
        // Tampilkan waktu pencarian dan akhiri program
    }

static void startSolve(float[] inpVal){
    // Berkomunikasi dengan Solver dan mengukur waktu pencarian

    long timeBefore = System.nanoTime();
    solutions = Solver.solve(inpVal);
    long timeAfter = System.nanoTime();

    elapsedNano = timeAfter - timeBefore;
}

static void displaySolutions(){
    // Menampilkan solusi ke layar terminal
    ...
}

static void promptSaveResults(){
    // Menyimpan solusi ke file .txt dalam folder test
    // Nama file diinputkan oleh pengguna
    ...
}
}

```

Alur kerja program utama:

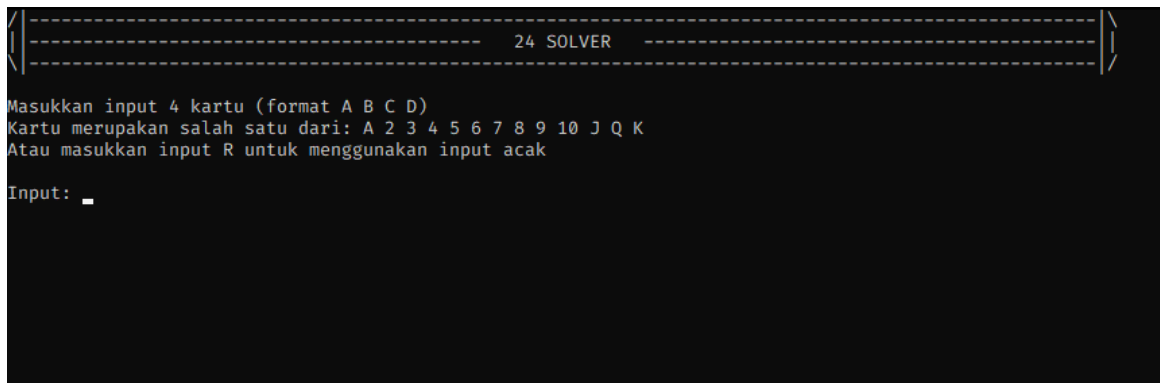
1. Program menerima input dari pengguna.
2. Jika input adalah 4 buah kartu yang valid, ubah menjadi 4 buah *float* senilai dengan nilai kartu. Sedangkan, jika input adalah R, bangkitkan 4 buah kartu acak dan gunakan sebagai input. Selain dari kedua kasus di atas, maka input tidak valid dan program meminta input baru.
3. Nilai input dikirim ke Solver dan program mencatat waktu sebelum pencarian.

4. Solver mengenumerasi semua kemungkinan solusi yang dapat dibentuk dari nilai input. Setiap kemungkinan solusi dievaluasi, dan solusi yang bernilai 24 disimpan dalam sebuah *list* dan dikembalikan ke program utama.
5. Program mencatat waktu setelah pencarian dan menghitung selisihnya dengan waktu sebelum eksekusi. Hasil perhitungan merupakan total waktu pencarian.
6. Program menampilkan isi *list* hasil pencarian ke layar terminal dan menyimpan isi *list* ke suatu file teks (jika pengguna memilih untuk menyimpan hasil).
7. Program menampilkan total waktu pencarian dan mengakhiri eksekusi.

## IV. Hasil Eksekusi

Berikut merupakan hasil eksekusi program Game24. Semua file solusi yang dibuat oleh program dapat ditemukan dalam folder *test*.

### 4.1 Tampilan Awal



```
<|----- 24 SOLVER -----|>
Masukkan input 4 kartu (format A B C D)
Kartu merupakan salah satu dari: A 2 3 4 5 6 7 8 9 10 J Q K
Atau masukkan input R untuk menggunakan input acak
Input: _
```

Gambar 4.1.1 Tampilan Awal Game24

## 4.2 Tampilan Solusi Input Valid

```
|----- 24 SOLVER -----|
|-----|
Masukkan input 4 kartu (format A B C D)
Kartu merupakan salah satu dari: A 2 3 4 5 6 7 8 9 10 J Q K
Atau masukkan input R untuk menggunakan input acak

Input: j k 7 8
Ditemukan 26 solusi sebagai berikut:
  J - (K * (7 - 8))      J - (K / (7 - 8))      (J + K) * (8 - 7)      J + (K * (8 - 7))
  (J + K) / (8 - 7)      J + (K / (8 - 7))      J - ((7 - 8) * K)      J + ((8 - 7) * K)
  (J * (8 - 7)) + K      (J / (8 - 7)) + K      K - (J * (7 - 8))      K - (J / (7 - 8))
  (K + J) * (8 - 7)      K + (J * (8 - 7))      (K + J) / (8 - 7)      K + (J / (8 - 7))
  K - ((7 - 8) * J)      K + ((8 - 7) * J)      ((K - 8) * 7) - J      (K * (8 - 7)) + J
  (K / (8 - 7)) + J      (7 * (K - 8)) - J      (8 - 7) * (J + K)      ((8 - 7) * J) + K
  (8 - 7) * (K + J)      ((8 - 7) * K) + J
Simpan solusi? (Y/N)
```

Gambar 4.2.1 Tampilan Solusi Game24 dengan Input Valid

## 4.3 Tampilan Solusi Input Acak

```
|----- 24 SOLVER -----|
|-----|
Masukkan input 4 kartu (format A B C D)
Kartu merupakan salah satu dari: A 2 3 4 5 6 7 8 9 10 J Q K
Atau masukkan input R untuk menggunakan input acak

Input: r
Masukan acak: A 10 4 3
Ditemukan 18 solusi sebagai berikut:
  (A + 3) * (10 - 4)      (10 - (A + 3)) * 4      ((10 - A) - 3) * 4      (10 - 4) * (A + 3)
  (10 - 4) * (3 + A)      (10 - (3 + A)) * 4      ((10 - 3) - A) * 4      (10 * (3 - A)) + 4
  4 - ((A - 3) * 10)      4 - (10 * (A - 3))      4 * (10 - (A + 3))      4 * ((10 - A) - 3)
  4 + (10 * (3 - A))      4 * (10 - (3 + A))      4 * ((10 - 3) - A)      4 + ((3 - A) * 10)
  (3 + A) * (10 - 4)      ((3 - A) * 10) + 4
Simpan solusi? (Y/N) _
```

Gambar 4.3.1 Tampilan Solusi Game24 dengan Input Acak

## 4.4 Tampilan Input Tidak Valid

```
|----- 24 SOLVER -----|
|-----|
Masukkan input 4 kartu (format A B C D)
Kartu merupakan salah satu dari: A 2 3 4 5 6 7 8 9 10 J Q K
Atau masukkan input R untuk menggunakan input acak

Input: 1 2 3 4
Masukan tidak valid!

Input: _
```

Gambar 4.4.1 Tampilan Game24 dengan Input Tidak Valid

## 4.5 Tampilan Menyimpan ke File

```
----- 24 SOLVER -----
Masukkan input 4 kartu (format A B C D)
Kartu merupakan salah satu dari: A 2 3 4 5 6 7 8 9 10 J Q K
Atau masukkan input R untuk menggunakan input acak

Input: j k 7 8
Ditemukan 26 solusi sebagai berikut:
  J - (K * (7 - 8))      J - (K / (7 - 8))      (J + K) * (8 - 7)      J + (K * (8 - 7))
  (J + K) / (8 - 7)      J + (K / (8 - 7))      J - ((7 - 8) * K)      J + ((8 - 7) * K)
  (J * (8 - 7)) + K      (J / (8 - 7)) + K      K - (J * (7 - 8))      K - (J / (7 - 8))
  (K + J) * (8 - 7)      K + (J * (8 - 7))      (K + J) / (8 - 7)      K + (J / (8 - 7))
  K - ((7 - 8) * J)      K + ((8 - 7) * J)      ((K - 8) * 7) - J      (K * (8 - 7)) + J
  (K / (8 - 7)) + J      (7 * (K - 8)) - J      (8 - 7) * (J + K)      ((8 - 7) * J) + K
  (8 - 7) * (K + J)      ((8 - 7) * K) + J

Simpan solusi? (Y/N) y
File: test/hasil_1
Solusi berhasil disimpan di test/hasil_1.txt

Program selesai dengan waktu pencarian 3ms
Tekan ENTER untuk menutup program...
```

Gambar 4.5.1 Tampilan Game24 Menyimpan ke File

```
hasil_1.txt - Notepad
File Edit Format View Help
J - (K * (7 - 8))
J - (K / (7 - 8))
(J + K) * (8 - 7)
J + (K * (8 - 7))
(J + K) / (8 - 7)
J + (K / (8 - 7))
J - ((7 - 8) * K)
J + ((8 - 7) * K)
(J * (8 - 7)) + K
(J / (8 - 7)) + K
K - (J * (7 - 8))
K - (J / (7 - 8))
(K + J) * (8 - 7)
K + (J * (8 - 7))
(K + J) / (8 - 7)
K + (J / (8 - 7))
K - ((7 - 8) * J)
K + ((8 - 7) * J)
((K - 8) * 7) - J
(K * (8 - 7)) + J
(K / (8 - 7)) + J
(7 * (K - 8)) - J
(8 - 7) * (J + K)
((8 - 7) * J) + K
(8 - 7) * (K + J)
((8 - 7) * K) + J
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Gambar 4.5.2 Tampilan Isi File Output Game24

## 4.6 Tampilan Gagal Menyimpan File

```
|----- 24 SOLVER -----|
|-----|
Masukkan input 4 kartu (format A B C D)
Kartu merupakan salah satu dari: A 2 3 4 5 6 7 8 9 10 J Q K
Atau masukkan input R untuk menggunakan input acak

Input: j k 7 8
Ditemukan 26 solusi sebagai berikut:
J - (K * (7 - 8))      J - (K / (7 - 8))      (J + K) * (8 - 7)      J + (K * (8 - 7))
(J + K) / (8 - 7)      J + (K / (8 - 7))      J - ((7 - 8) * K)      J + ((8 - 7) * K)
(J * (8 - 7)) + K      (J / (8 - 7)) + K      K - (J * (7 - 8))      K - (J / (7 - 8))
(K + J) * (8 - 7)      K + (J * (8 - 7))      (K + J) / (8 - 7)      K + (J / (8 - 7))
K - ((7 - 8) * J)      K + ((8 - 7) * J)      ((K - 8) * 7) - J      (K * (8 - 7)) + J
(K / (8 - 7)) + J      (7 * (K - 8)) - J      (8 - 7) * (J + K)      ((8 - 7) * J) + K
(8 - 7) * (K + J)      ((8 - 7) * K) + J

Simpan solusi? (Y/N) y
File: test/<><>
ERR: Gagal menyimpan file: java.io.FileNotFoundException: test/<><>.txt (The filename, directory name, or volume label syntax is incorrect)

Program selesai dengan waktu pencarian 3ms
Tekan ENTER untuk menutup program...
```

Gambar 4.6.1 Tampilan Game24 Gagal Menyimpan File

## 4.7 Tampilan Tidak Menyimpan ke File

```
|----- 24 SOLVER -----|
|-----|
Masukkan input 4 kartu (format A B C D)
Kartu merupakan salah satu dari: A 2 3 4 5 6 7 8 9 10 J Q K
Atau masukkan input R untuk menggunakan input acak

Input: j k 7 8
Ditemukan 26 solusi sebagai berikut:
J - (K * (7 - 8))      J - (K / (7 - 8))      (J + K) * (8 - 7)      J + (K * (8 - 7))
(J + K) / (8 - 7)      J + (K / (8 - 7))      J - ((7 - 8) * K)      J + ((8 - 7) * K)
(J * (8 - 7)) + K      (J / (8 - 7)) + K      K - (J * (7 - 8))      K - (J / (7 - 8))
(K + J) * (8 - 7)      K + (J * (8 - 7))      (K + J) / (8 - 7)      K + (J / (8 - 7))
K - ((7 - 8) * J)      K + ((8 - 7) * J)      ((K - 8) * 7) - J      (K * (8 - 7)) + J
(K / (8 - 7)) + J      (7 * (K - 8)) - J      (8 - 7) * (J + K)      ((8 - 7) * J) + K
(8 - 7) * (K + J)      ((8 - 7) * K) + J

Simpan solusi? (Y/N) n

Program selesai dengan waktu pencarian 3ms
Tekan ENTER untuk menutup program...
```

Gambar 4.7.1 Tampilan Game24 Tidak Menyimpan ke File



## V. Lampiran

### 5.1 Repository GitHub

*Repository* untuk *source code* dan hasil kompilasi program ini dapat diakses lewat tautan berikut: [https://github.com/JerichoFletcher/Tucil1\\_13521107](https://github.com/JerichoFletcher/Tucil1_13521107)

### 5.2 Tabel Penilaian

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan		
2. Program berhasil <i>running</i>		
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		

## VI. Daftar Pustaka

Chandra, Evita. 2016. “Penerapan Algoritma Brute Force pada Permainan Kartu 24 (24 *game*)”. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf> diakses pada 23 Januari 2023.

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](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf) diakses pada 23 Januari 2023.