IF2211 Strategi Algoritma

24 Game Solver Menggunakan Algoritma *Brute Forice*

Laporan Tugas Kecil I

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma pada Semester 2 (dua) Tahun Akademik 2022/2023

Disusun oleh:

Ariel Jovananda K02 / 13521086

PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG

BANDUNG 2023

DAFTAR ISI

BAB 1

PENDAHULUAN

Algoritma brute-force adalah teknik yang digunakan untuk mencari solusi dengan mempertimbangkan semua kemungkinan. Algoritma ini merupakan metode yang kurang efektif dan lebih rumit, tetapi sering digunakan karena mudah dibuat dan dapat diterapkan pada berbagai masalah. Secara umum, proses algoritma brute force dapat diubah menjadi beberapa tugas berbeda:

1. Perhatikan jenis masalah yang akan dipecahkan.
2. Tuliskan semua kemungkinan penyebab masalah saat ini.
3. Mulailah dengan mempertimbangkan setiap kemungkinan satu per satu hingga ditemukan solusi yang sesuai.
4. Jika solusi tidak ditemukan setelah semua asumsi yang masuk akal dibuat, algoritme akan menghentikan proses saat ini.

Efektivitas brute force dalam memecahkan masalah sangat dibatasi oleh jumlah kemungkinan yang ada. Ada lebih banyak kemungkinan, sehingga butuh waktu lebih lama untuk menyelidiki solusinya. solusi. Akibatnya, algoritma *brute force* sering digunakan untuk menyelesaikan masalah yang relatif mudah dan tidak melibatkan banyak data atau proses rumit karena berisiko menyebabkan ledakan kombinatorial selama proses pencarian solusi.

Dalam tugas kecil ini, algoritma brute force digunakan untuk mencari solusi dalam permainan kart 24 jam. Game 24 adalah game yang menantang kemampuan pemain dalam matematika dan logika. Dalam permainan ini, pemain menerima delapan kartu remi dan diperintahkan untuk mempelajari operasi matematika dasar termasuk menghitung, menambah, mengurangi, dan mengalikan untuk mencapai angka 24. Satu-satunya cara yang dapat digunakan untuk menyelesaikan soal di Kartu 24 permainan adalah algoritma *brute force*.

BAB 2

ALGORITMA BRUTE FORCE

Algoritma brute-force adalah teknik yang digunakan untuk menemukan solusi dengan mempertimbangkan semua hasil yang layak. Dalam situasi ini, algoritme brute force akan dijalankan melalui setiap operasi matematis yang secara teoritis dapat dilakukan pada kartu buah keempat yang diberikan, hingga menemukan solusi yang menghasilkan total 24 solusi. Secara umum, berikut ini dapat dilakukan saat menerapkan algoritma brute force:

1. Menginisialisasi empat angka dari masukan *user* atau *generate* secara acak
2. Lakukan permutasi kepada 4 angka tersebut
3. Lakukan permutasi kepada lokasi 4 operasi (+, -, *, /) diantara 4 angka tersebut
4. Periksa kombinasi antara semua permutasi angka dan operator, jika hasilnya 24, simpan kombinasi tersebut dalam bentuk *string* di dalam suatu larik.
5. Jika semua kombinasi sudah di periksa, periksa isi larik. Jika kosong maka tidak ada solusi, dan sebaliknya.
6. Mengluarkan total isi larik, dan mengeluarkan semua *string* yang berada di dalam larik tersebut.

Penggunaan kekuatan belaka untuk mencari setiap solusi yang mungkin untuk permainan persisten 24 jam tidak disarankan, namun ada kekurangan dan risikonya. Deskripsi algoritma per bagian lebih dalam disertai kode sumber akan dilengkapi pada bagian berikut, diikuti dengan analisis waktu yang komprehensif untuk mengevaluasi kinerja algoritma.

BAB 3

IMPLEMENTASI PROGRAM

Bahasa yand dipakai adalah *C++*, berikut adalah *source code* dari program ini. Program ini menggunakan dua *file* yaitu *functions.cpp* yang memiliki semua fungsi-fungsi yang dipakai di dalam program ini dan *main.cpp* untuk menjalankan program.

functions.cpp

Fungsi *permuteNumbers*

```
//Fungsi untuk melakukan permutasi pada suatu array integer berjumlah 4
void permuteNumbers (int array[4], int (*arrOfArr)[4]){
    int line = 0;
    int count = 0;

    for(int i = 0; i < 4; i++){
        for(int j = 0; j < 4; j++){
            for (int k = 0; k < 4; k++){
                for(int l = 0; l < 4; l++){
                    if (i != j && i != k && i != l && j != k && j != l && k != l){
                        arrOfArr[line][0] = array[i];
                        arrOfArr[line][1] = array[j];
                        arrOfArr[line][2] = array[k];
                        arrOfArr[line][3] = array[l];
                        line++;
                    }
                }
            }
        }
    }
}
```

Fungsi *permuteOp*

```
//Fungsi untuk melakukan permutasi kepada array yang berisi operasi matematika
void permuteOp (char arrOfArr [64][3]){
    char array[4] = {'+', '-', '*', '/'};
    int line = 0;
    int count = 0;

    for(int i = 0; i < 4; i++){
        for(int j = 0; j < 4; j++){
            for (int k = 0; k < 4; k++){
                arrOfArr[line][0] = array[i];
                arrOfArr[line][1] = array[j];
                arrOfArr[line][2] = array[k];
                line++;
            }
        }
    }
}
```

Funsi *performOperation*

```c
//Fungsi untuk melakukan operasi
double performOperation(double num1, double num2, char op) {
    if (op == '+') return num1 + num2;
    if (op == '-') return num1 - num2;
    if (op == '*') return num1 * num2;
    if (op == '/') {
        if (num2 == 0) {
            return false;
        }
        else{
            return num1 / num2;
        }
    }
    return -1; // in case of invalid operator
}
```

Fungsi *checkArr*

```c
//Fungsi untuk mencocokkan array dan mengembalikkan nomor urutan array yang sama
int checkArr(int arrBin[7]){
    int arr1[7] = {0, 0, 0, 0, 1, 1, 1};
    int arr2[7] = {0, 0, 0, 1, 0, 1, 1};
    int arr3[7] = {0, 0, 0, 1, 1, 0, 1};
    int arr4[7] = {0, 0, 1, 0, 0, 1 ,1};
    int arr5[7] = {0, 0, 1, 0, 1, 0, 1};
    bool valid = true;
    int i;

    for(i = 2; i < 7; i++){
        if(arrBin[i] != arr1[i]){
            valid = false;
            break;
        }
    }
    if(valid){
        return 1;
    }
    valid = true;

    for( i = 2; i < 7; i++){
        if(arrBin[i] != arr2[i]){
            valid = false;
            break;
        }
    }
    if(valid){
        return 2;
    }
    valid = true;

    for( i = 2; i < 7; i++){
        if(arrBin[i] != arr3[i]){
            valid = false;
            break;
        }
    }
    if(valid){
        return 3;
    }
    valid = true;

    for( i = 2; i < 7; i++){
        if(arrBin[i] != arr4[i]){
            valid = false;
            break;
        }
    }
    if(valid){
        return 4;
    }
    valid = true;

    for( i = 2; i < 7; i++){
        if(arrBin[i] != arr5[i]){
            valid = false;
            break;
        }
    }
    if(valid){
        return 5;
    }
    return -1;
}
```

Fungsi *genPostfix*

```cpp
//Fungsi untuk menghasilkan string yang sudah diubah dari postfix ke infix
string genPostFix (int arrNum[4], char arrOp[3], int arrBin[7]){
    int ret = checkArr(arrBin);
    string res = "";
    switch(ret){
        case 1:
            res += std::to_string(arrNum[0]) + " " + arrOp[2] + " " + "(" + std::to_string(arrNum[1]) + " " + arrOp[1] + " " + "(" +
            std::to_string(arrNum[2]) + " " + arrOp[0] + " " + std::to_string(arrNum[3]) + ")" + ")";
            break;
        case 2:     namespace std
            res += std::to_string(arrNum[0]) + " " + arrOp[2] + " " + "(" + "(" + std::to_string(arrNum[1]) + " " + arrOp[0] + " " +
            std::to_string(arrNum[2]) + ")" + " " + arrOp[1] + " " + std::to_string(arrNum[3]) + ")";
            break;
            //a+((b+c)+d)
        case 3:
            res += "(" + std::to_string(arrNum[0]) + " " + arrOp[1] + " " + "(" + std::to_string(arrNum[1]) + " " + arrOp[0] + " " +
            std::to_string(arrNum[2]) + ")" + ")" + " " + arrOp[2] + " " + std::to_string(arrNum[3]);
            break;

        case 4:
            res += "(" + std::to_string(arrNum[0]) + " " + arrOp[0] + " " + std::to_string(arrNum[1]) + ")" + " " + arrOp[2] + " " + "(" +
            std::to_string(arrNum[2]) + " " + arrOp[1] + " " + std::to_string(arrNum[3]) + ")";
            break;
        case 5:
            res += "(";
            res += "(" + std::to_string(arrNum[0]) + " " + arrOp[0] + " " + std::to_string(arrNum[1]) + ")" + " " + arrOp[1] + " " +
            std::to_string(arrNum[2]) + ")" + " "+ arrOp[2] + " " + std::to_string(arrNum[3]);
            break;
    }
    return res;
}
```

Fungsi *postFix*

```cpp
//Fungsi untuk melakukan perhitungan postfix
int postFix(int arrOfNum[24][4], std:: vector<string>& arrStr){
    int arrOfBin[5][7] = {{0, 0, 0, 0, 1, 1, 1},
                          {0, 0, 0, 1, 0, 1, 1},
                          {0, 0, 0, 1, 1, 0, 1},
                          {0, 0, 1, 0, 0, 1 ,1},
                          {0, 0, 1, 0, 1, 0, 1}};
    char arrOp[64][3];
    int resultSum = 0;
    bool valid;

    permuteOp(arrOp);
    for(int i = 0; i < 5; i++){
        for(int j = 0; j < 24; j++){
            for(int k = 0; k < 64; k++){
                int currNum = 0;
                int currOp = 0;
                std:: stack<double> stack;
                for(int l = 0; l < 7; l++){
                    if (arrOfBin[i][l] == 0){
                        stack.push(arrOfNum[j][currNum]);
                        currNum++;
                    }
                    else{
                        double num2 = stack.top(); stack.pop();
                        double num1 = stack.top(); stack.pop();
                        double result = performOperation(num1, num2, arrOp[k][currOp]);
                        stack.push(result);
                        currOp++;
                        if (l == 6) {
                            result = stack.top(); stack.pop();
                            if (result == 24){
                                arrStr.push_back(genPostFix(arrOfNum[j], arrOp[k], arrOfBin[i]));
                                resultSum++;
                            }
                        }
                    }
                }
```

```
            }
        }
    }
    if (resultSum > 0) {
        cout << resultSum << " " << "solutions found" << '\n';
    }
    else{
        cout << "Tidak ada solusi" << '\n';
    }
    for(int i = 0; i < arrStr.size(); i++){
        cout << arrStr[i] << "\n";
    }
    return resultSum;
}
```

Fungsi *converter*

```
//Fungsi untuk mengubah array of strings menjadi array of int
void converter(string arr[4], int arrNum[4]){
    for(int i = 0; i < 4; i++){
        if (arr[i] == "A"){
            arrNum[i] = 1;
        }
        if (arr[i] == "2"){
            arrNum[i] = 2;
        }
        if (arr[i] == "3"){
            arrNum[i] = 3;
        }
        if (arr[i] == "4"){
            arrNum[i] = 4;
        }
        if (arr[i] == "5"){
            arrNum[i] = 5;
        }
        if (arr[i] == "6"){
            arrNum[i] = 6;
        }
        if (arr[i] == "7"){
            arrNum[i] = 7;
        }
        if (arr[i] == "8"){
            arrNum[i] = 8;
        }
        if (arr[i] == "9"){
            arrNum[i] = 9;
        }
        if (arr[i] == "10"){
            arrNum[i] = 10;
        }
        if (arr[i] == "J"){
            arrNum[i] = 11;
        }
        if (arr[i] == "Q"){
            arrNum[i] = 12;
```

```
        }
        if (arr[i] == "K"){
            arrNum[i] = 13;
        }
    }

}
```

Fungsi *converterKartu*

```
//Fungsi untuk mengubah array of int menjadi array of string
void converterKartu(int arrNum[4], string arr[4]){
    for(int i = 0; i < 4; i++){
        if (arrNum[i] == 1){
            arr[i] = "A";
        }
        if (arrNum[i] == 2){
            arr[i] = "2";
        }
        if (arrNum[i] == 3){
            arr[i] = "3";
        }
        if (arrNum[i] == 4){
            arr[i] = "4";
        }
        if (arrNum[i] == 5){
            arr[i] = "5";
        }
        if (arrNum[i] == 6){
            arr[i] = "6";
        }
        if (arrNum[i] == 7){
            arr[i] = "7";
        }
        if (arrNum[i] == 8){
            arr[i] = "8";
        }
        if (arrNum[i] == 9){
            arr[i] = "9";
        }
        if (arrNum[i] == 10){
            arr[i] = "10";
        }
        if (arrNum[i] == 11){
            arr[i] = "J";
        }
        if (arrNum[i] == 12){
            arr[i] = "Q";
```

```
        }
        if (arrNum[i] == 13){
            arr[i] = "K";
        }
    }

}
```

Fungsi *writeToFile*

```cpp
//Fungsi untuk menulis output dalam suatu file yang akan dibuat
void writeToFile(string kartu[4], int solusi, vector<string>& arrStr, double time){
    string namaFile;
    cout << "Nama file yang diinginkan: ";
    cin >> namaFile;

    ofstream outFile;
    outFile.open("./test/" + namaFile);

    outFile << "Kartu: " << kartu[0] << " " << kartu[1] << " " << kartu[2] << " " << kartu[3] << std::endl;
    if (solusi > 0){
        outFile << solusi << " solutions found" << std::endl;
    }
    else{
        outFile << "Tidak ada solusi" << std::endl;
    }
    for(int i = 0; i < arrStr.size(); i++){
        outFile << arrStr[i] << "\n";
    }
    outFile << "Runtime: " << time << " detik" << std::endl;

    // Close the file
    outFile.close();

}
```

main.cpp

```cpp
#include <iostream>
#include <stack>
#include <sstream>
#include <string>
#include <time.h>
#include <vector>
#include <ctime>
#include <fstream>
#include "functions.cpp"
using namespace std;

int main (){
    int pil;
    cout << "Ketik 1 untuk memasukkan kartu\n" << "Ketik 2 untuk mengacak kartu\n";
    cin >> pil;
    if (pil == 1){
        bool valid = false;
        int arrNum[4];
        int arrOfNum[24][4];
        string arr[4];
        std:: vector<string> arrStr;
        while(!valid){
            cout << "Kartu (contoh input: A B C D): ";
            string input;
            string input1;
            string input2;
            string input3;
            int count = 0;
            cin >> input >> input1 >> input2 >> input3;
            if (input == "2"|| input == "3" || input == "4" || input == "5" || input == "6" || input == "7" || input == "8" || input == "9" |
            | input == "10" || input == "A" || input == "J" || input == "Q" || input == "K"){
                count++;
                arr[0] = input;
            }
            if (input1 == "2"|| input1 == "3" || input1 == "4" || input1 == "5" || input1 == "6" || input1 == "7" || input1 == "8" || input1
            == "9" || input1 == "10" || input1 == "A" || input1 == "J" || input1 == "Q" || input1 == "K"){
                count++;
                arr[1] = input1;
```

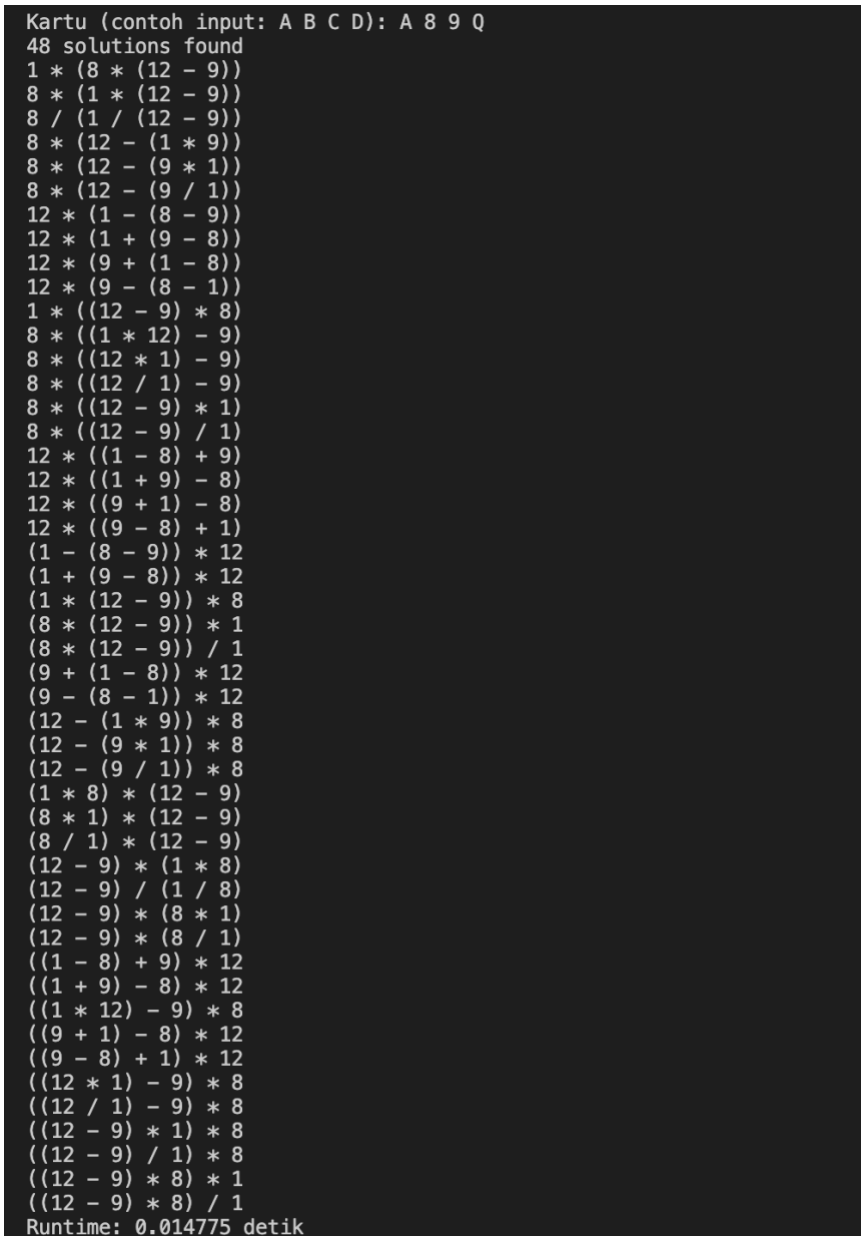
```
        }
        if (input2 == "2"|| input2 == "3" || input2 == "4" || input2 == "5" || input2 == "6" || input2 == "7" || input2 == "8" || input2
        == "9" || input2 == "10" || input2 == "A" || input2 == "J" || input2 == "Q" || input2 == "K"){
            count++;
            arr[2] = input2;
        }
        if (input3 == "2"|| input3 == "3" || input3 == "4" || input3 == "5" || input3 == "6" || input3 == "7" || input3 == "8" || input3
        == "9" || input3 == "10" || input3 == "A" || input3 == "J" || input3 == "Q" || input3 == "K"){
            count++;
            arr[3] = input3;
        }
        if(count == 4){
            valid = true;
        }
        else{
            cout << "Masukkan salah\n";
        }
    }
    clock_t start = clock();
    converter(arr, arrNum);
    permuteNumbers(arrNum, arrOfNum);
    int solusi = postFix(arrOfNum, arrStr);
    clock_t end = clock();
    double runtime = (double)(end - start) / CLOCKS_PER_SEC;
    std::cout << "Runtime: " << runtime << " detik" << std::endl;
    cout << "Apakah ingin menyimpan output di dalam file (y/n)\n";
    char pil;
    cin >> pil;
    while(pil != 'y' && pil != 'n'){
        cout << "Masukkan tidak valid, silahkan ulangi\n";
        cout << "Apakah ingin menyimpan output di dalam file (y/n)\n";
        cin >> pil;
    }
    if(pil == 'y'){
        writeToFile(arr, solusi, arrStr, runtime);
        return 0;
      else{
        return 0;
      }
  }
  if (pil == 2){
      int arrNum[4];
      int arrOfNum[24][4];
      string kartu[4];
      std:: vector<string> arrStr;
      srand(time(0));
      arrNum[0] = rand() % 13 + 1;
      arrNum[1] = rand() % 13 + 1;
      arrNum[2] = rand() % 13 + 1;
      arrNum[3] = rand() % 13 + 1;
      converterKartu(arrNum, kartu);
      cout << "Hasil acak kartu: ";
      cout << kartu[0] << " " << kartu[1] << " " << kartu[2] << " " << kartu[3] << "\n";
      clock_t start = clock();
      permuteNumbers(arrNum, arrOfNum);
      int solusi = postFix(arrOfNum, arrStr);
      clock_t end = clock();
      double runtime = (double)(end - start) / CLOCKS_PER_SEC;
      std::cout << "Runtime: " << runtime << " detik" << std::endl;
      cout << "Apakah ingin menyimpan output di dalam file (y/n)\n";
      char pil;
      cin >> pil;
      while(pil != 'y' && pil != 'n'){
          cout << "Masukkan tidak valid, silahkan ulangi\n";
          cout << "Apakah ingin menyimpan output di dalam file (y/n)\n";
          cin >> pil;
      }
      if(pil == 'y'){
          writeToFile(kartu, solusi, arrStr, runtime);
          return 0;
      }
      else{
          return 0;
```

BAB 4

INPUT DAN OUTPUT PROGRAM

Berikut adalah contoh-contoh tes input mengetes apakah program sudah berjalan dengan baik dan benar, sekaligus output yang dihasilkan dari beberapa input tersebut.

Input dan hasil output pertama



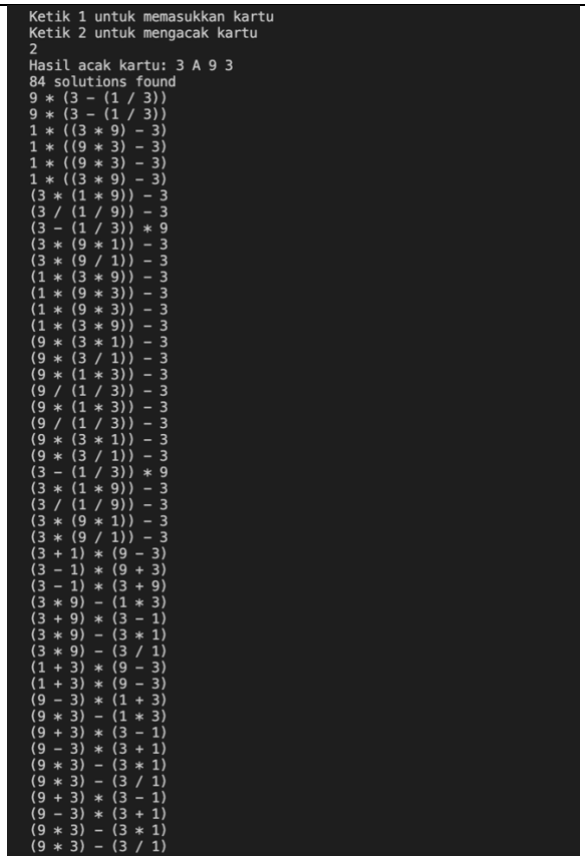
Input dan hasil output kedua

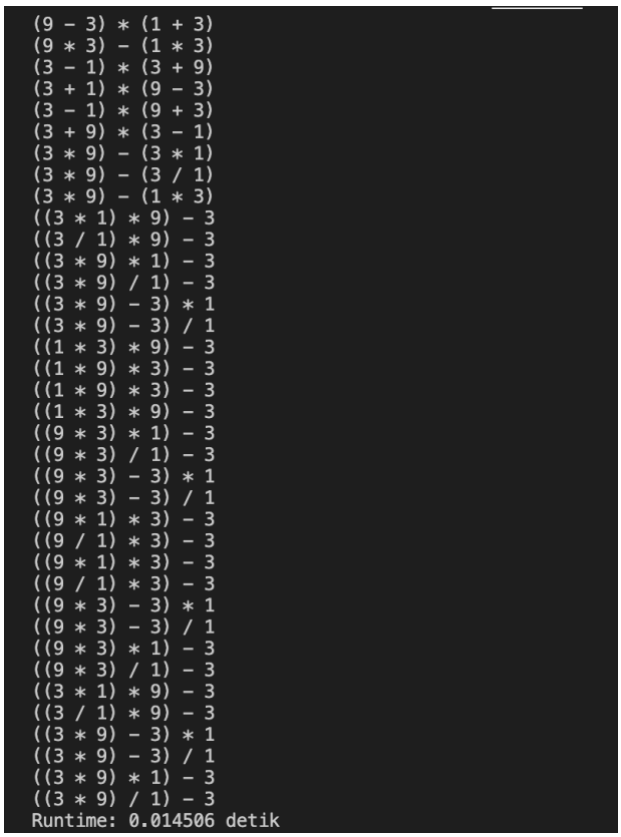
```
Ketik 1 untuk memasukkan kartu
Ketik 2 untuk mengacak kartu
1
Kartu (contoh input: A B C D): 7 7 7 7
Tidak ada solusi
Runtime: 0.015698 detik
```

Input dan hasil output ketiga

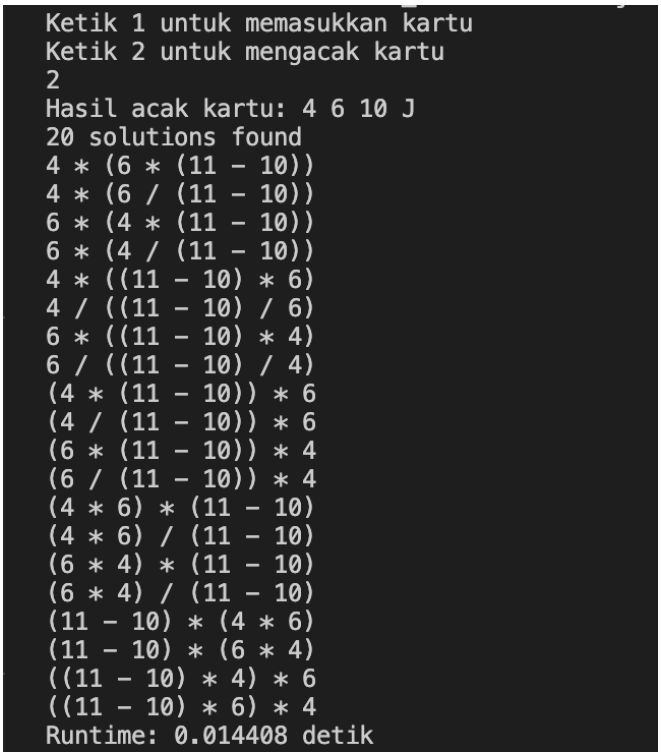
```
Ketik 1 untuk memasukkan kartu
Ketik 2 untuk mengacak kartu
2
Hasil acak kartu: Q 4 9 9
16 solutions found
4 * (9 - (12 - 9))
4 * (9 + (9 - 12))
4 * (9 - (12 - 9))
4 * (9 + (9 - 12))
4 * ((9 - 12) + 9)
4 * ((9 + 9) - 12)
4 * ((9 - 12) + 9)
4 * ((9 + 9) - 12)
(9 - (12 - 9)) * 4
(9 + (9 - 12)) * 4
(9 - (12 - 9)) * 4
(9 + (9 - 12)) * 4
((9 - 12) + 9) * 4
((9 + 9) - 12) * 4
((9 - 12) + 9) * 4
((9 + 9) - 12) * 4
Runtime: 0.013178 detik
```

Input dan hasil output keempat





Input dan hasil output kelima



Input dan hasil output keenam

```
Ketik 1 untuk memasukkan kartu
Ketik 2 untuk mengacak kartu
2
Hasil acak kartu: 9 A 9 7
90 solutions found
9 - (1 - (9 + 7))
9 - (1 - (7 + 9))
9 + (9 - (1 - 7))
9 + (9 + (7 - 1))
9 + (7 - (1 - 9))
9 + (7 + (9 - 1))
9 + (9 - (1 - 7))
9 + (9 + (7 - 1))
9 - (1 - (9 + 7))
9 - (1 - (7 + 9))
9 + (7 + (9 - 1))
9 + (7 - (1 - 9))
7 + (9 - (1 - 9))
7 + (9 + (9 - 1))
7 - (1 - (9 + 9))
7 - (1 - (9 + 9))
7 + (9 + (9 - 1))
7 + (9 - (1 - 9))
9 - ((1 - 9) - 7)
9 - ((1 - 7) - 9)
9 + ((9 - 1) + 7)
9 + ((9 + 7) - 1)
9 + ((7 - 1) + 9)
9 + ((7 + 9) - 1)
9 + ((9 - 1) + 7)
9 + ((9 + 7) - 1)
9 - ((1 - 9) - 7)
9 - ((1 - 7) - 9)
9 + ((7 + 9) - 1)
9 + ((7 - 1) + 9)
7 + ((9 - 1) + 9)
7 + ((9 + 9) - 1)
7 - ((1 - 9) - 9)
7 - ((1 - 9) - 9)
7 + ((9 + 9) - 1)
7 + ((9 - 1) + 9)
(9 - (1 - 9)) + 7
(9 - (1 - 7)) + 9
(9 + (9 - 1)) + 7
(9 + (9 + 7)) - 1
(9 + (7 - 1)) + 9
(9 + (7 + 9)) - 1
(9 + (9 - 1)) + 7
(9 + (9 + 7)) - 1
(9 - (1 - 9)) + 7
(9 - (1 - 7)) + 9
(9 + (7 + 9)) - 1
```

```
(9 + (7 - 1)) + 9
(7 + (9 - 1)) + 9
(7 + (9 + 9)) - 1
(7 - (1 - 9)) + 9
(7 - (1 - 9)) + 9
(7 + (9 + 9)) - 1
(7 + (9 - 1)) + 9
(9 - 1) + (9 + 7)
(9 - 1) + (7 + 9)
(9 + 9) - (1 - 7)
(9 + 9) + (7 - 1)
(9 + 7) - (1 - 9)
(9 + 7) + (9 - 1)
(9 + 9) - (1 - 7)
(9 + 9) + (7 - 1)
(9 - 1) + (9 + 7)
(9 - 1) + (7 + 9)
(9 + 7) + (9 - 1)
(9 + 7) - (1 - 9)
(7 + 9) - (1 - 9)
(7 + 9) + (9 - 1)
(7 - 1) + (9 + 9)
(7 - 1) + (9 + 9)
(7 + 9) + (9 - 1)
(7 + 9) - (1 - 9)
((9 - 1) + 9) + 7
((9 - 1) + 7) + 9
((9 + 9) - 1) + 7
((9 + 9) + 7) - 1
((9 + 7) - 1) + 9
((9 + 7) + 9) - 1
((9 + 9) - 1) + 7
((9 + 9) + 7) - 1
((9 - 1) + 9) + 7
((9 - 1) + 7) + 9
((9 + 7) + 9) - 1
((9 + 7) - 1) + 9
((7 + 9) - 1) + 9
((7 + 9) + 9) - 1
((7 - 1) + 9) + 9
((7 - 1) + 9) + 9
((7 + 9) + 9) - 1
((7 + 9) - 1) + 9
Runtime: 0.013779 detik
```

LAMPIRAN

Link Repository: https://github.com/arieljovananda88/Tucil1_13521086

Tabel Keselesaian:

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	✓	