Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh:

13521156 – Brigita Tri Carolina

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

BAB 1

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma Brute Force

Algoritma *Brute Force* merupakan sebuah pendeketan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma ini didasarkan pada pernyataan pada persoalan (*problem statement*) dan definisi/konsep yang dilibatkan. Algoritma *Brute Force* memecahkan persoalan dengan sangat sederhana, jelas caranya, dan *just do it!*

Algoritma *Brute Force* biasanya mencari semua kemungkinan hingga sebuah solusi ditemukan. Algoritma *Brute Force* merupakan algoritma yang sederhana dan konsisten, namun seringkali membutuhkan waktu yang lama dalam penyelesaian beberapa masalah dengan algoritma ini.

1.2 Permainan Kartu 24 (*Make It 24*)

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.

1.3 Algoritma Penyelesaian Permainan Kartu 24 dengan Pendekatan Brute Force

Pada penyelesaian pemainan kartu 24 ini, digunakan algoritma dengan pendekatan *Brute Force*. Langkah-langkah penyelesaiannya dengan menggunakan metode Brute Force adalah sebagai berikut:

1. Program menerima input 4 angka (2-10) atau huruf (A, J, Q, K) *random* dari pengguna ataupun pengguna dapat memilih agar program men-*generate* sendiri 4 angka random tersebut.
2. Ke-empat angka yang terbaca sebagai string tersebut kemudian disimpan dalam sebuah *array of string*.
3. Angka-angka yang tersimpan pada *array of string* kemudian diubah menggunakan suatu fungsi menjadi nilai integer aslinya.
4. Langkah penyelesaian *brute force*:
    a. Misal 4 angka *random* tersebut adalah a, b, c, d, maka contoh ekspresi solusi adalah ((a + b) + c) + d. 4 angka tersebut dipermutasi sehingga terdapat 4! = 24 kemungkinan dari penempatan ke 4 angka tersebut.

b. Terdapat 4 operator yang dapat digunakan yaitu +, -, /, *. Dari ke empat operator tersebut dapat diambil 3 operator untuk dimasukkan ke dalam ekspresi solusi. Maka banyaknya kemungkinan dari 3 operator yang diambil dari 4 operator adalah 4*4*4 = 64 kemungkinan.
c. Misal N adalah angka dan O adalah operator, maka terdapat 5 bentuk solusi dari penukaran tanda kurung pada ekspresi, yaitu:
   1.) ((N O N) O N) O N
   2.) (N O N) O (N O N)
   3.) (N O (N O N)) O N
   4.) N O ((N O N) O N)
   5.) N O (N O (N O N))
d. Semua kemungkinan tersebut kemudian disatukan menjadi 24 * 64 * 5 = 7.680 kemungkinan solusi.

5. Untuk melakukan permutasi angka dan kombinasi operator yang digunakan maka dilakukan enam kali looping (4, 3, 2, 4, 4, 4), pada looping terdalam dibentuk 5 ekspresi di atas yang merupakan 5 bentuk solusi dari penukaran tanda kurung pada ekspresi.
6. Dari 7.680 solusi yang dihasilkan maka dipilih solusi yang menghasilkan angka 24. Kemudian dihitung berapa banyak solusi yang menghasilkan 24, solusi disimpan pada *buffer string* lalu dicetak pada terminal atau pada suatu file teks yang telah ditentukan pengguna.

BAB 2

IMPLEMENTASI ALGORITMA DALAM BAHASA C

Program ini dibuat dalam bahasa pemrograman C. Adapun struktur program terbagi menjadi dua bagian yaitu main.c dan operations.c

2.1. main.c

File merupakan driver utama dari program berisi algoritma penyelesaian permainan kartu 24 menggunakan pendekatan *brute force* dan implementasi fungsi-fungsi dari operations.c.

2.2 operations.c

File berisi fungsi-fungsi maupun prosedur-prosedur yang dapat membantu penyelesaian permainan kartu 24 pada program utama.

Fungsi	Deskripsi	Implementasi
`float eval(float a, float b, char op)`	Menghitung hasil dari operasi float a dengan float b dengan char op	Digunakan pada perhitungan nilai ekspresi sama dengan 24
`void remove_element(int arr[], int index, int buffer[], int size)`	Prekondisi: arr memiliki elemen dan tidak kosong. F.S.: terbentuk array buffer yang berisi elemen arr tanpa elemen arr[index] di dalamnya	Digunakan untuk permutasi 4 nilai angka (menghapus nilai dari arr ketika sudah digunakan)
`boolean same(char *s1, char *s2)`	Menghasilkan *true* jika string s1 sama dengan string s2, *false* jika tidak	Digunakan pada fungsi isAlphabet dan isInt untuk membandingkan nilai string
`boolean isAlphabet(char num[])`	Menghasilkan *true* *string* num merupakan alphabet (A, J, Q, K)	Digunakan untuk memvalidasi input dari pengguna
`boolean isInt(char num[])`	Menghasilkan *true* jika *string* num merupakan *integer* (2-10).	Digunakan untuk memvalidasi input dari pengguna

`int transformToInt(char num[])`	Menghasilkan nilai *integer* dari *string* num	Digunakan untuk mengubah input dari user yang berupa *string number* menjadi nilai integernya
`int transformToNumber(char num[])`	Menghasilkan nilai *integer* dari *string* num alphabet (A = 1, J = 11, Q = 12, K = 13)	Digunakan untuk mengubah input dari user yang berupa *string alphabet* (A, J, Q, K) menjadi nilai integernya

2.3 boolean.h

File mendefinisikan tipe boolean.

BAB 3

SOURCE CODE PROGRAM

3.1. Repository Program

Repository program dapat diakses melalui:

- Github: https://github.com/BrigitaCarolina/Tucil1_13521156

3.2. Source Code Program

3.2.1 main.c

```c
#include "operations.h"
#include "boolean.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    clock_t start, end;
    float execution_time;
    FILE *fptr;
    FILE *pita;
    boolean flag;
    char ops[] = {'+', '-', '/', '*'};
    char op1, op2, op3, enter;
    char file[100];
    char txt[50];
    char *path;
    char solutions[200][200];
    char num[10];
    char num_array[10][10];
    // Word num1, num2, num3, num4;
    int i, j, k, x, y, z, a, b, c, d, count, count1, pilihan, retval;
    int nums[] = {0, 0, 0, 0};
    int buffer3[] = {0, 0, 0};
    int buffer2[] = {0, 0};
    int buffer[] = {0};

    printf("*********************************\n");
    printf("*                               *\n");
    printf("*         24 GAME SOLVER        *\n");
    printf("*                               *\n");
    printf("*********************************\n");
    printf("Pilih cara menentukan 4 angka: \n");
    printf("1. Program otomatis mengenerate angka\n");
    printf("2. Masukan manual dari pengguna/user\n");
    printf(">> ");
    scanf("%d", &pilihan);

    while (pilihan != 1 && pilihan != 2) {
        printf("Masukan tidak valid! Hanya masukkan 1 atau 2!\n");
        printf("Pilih cara menentukan 4 angka: \n");
        printf("1. Program otomatis mengenerate angka\n");
        printf("2. Masukan manual dari pengguna/user\n");
        printf(">> ");
        scanf("%d", &pilihan);
    }
```

```c
if (pilihan == 1) {
    srand((unsigned int) time(NULL));
    printf("4 angka yang ter-generate: ");
    for (int i = 0; i < 4; i++) {
        nums[i] = (rand() % 13) + 1;
        printf("%d ", nums[i]);
    }
    printf("\n");
} else if (pilihan == 2) {
    flag = true;
    printf("Masukkan 4 buah angka: \n");
    printf(">> ");
    count1 = 0;
    count_i = 0;
    count_j = 0;
    scanf("%c", &enter);
    ch = getchar();
    while (ch != '\n') {
        if (ch == ' ') {
            num_array[count_i][count_j] = '\0';
            count_i++;
            count_j = 0;
        } else {
            num_array[count_i][count_j] = ch;
            count_j++;
        }
        ch = getchar();
    }

    if (count_i > 3) {
        printf("Hanya masukan sejumlah 4 buah angka!\n");
        flag = false;
    }


    if (flag) {
        for (int count2 = 0; count2 < 4; count2++) {
            if (!isInt(num_array[count2]) && !isAlphabet(num_array[count2])) {
                flag = false;
            }
        }
        if (!flag) {
            printf("Masukan tidak valid, masukkan hanya angka (2-10) atau huruf (A, J, Q, K)!\n");
        }
    }
```

```
if (flag) {
    for (int count2 = 0; count2 < 4; count2++) {
        if (!isInt(num_array[count2]) && !isAlphabet(num_array[count2])) {
            flag = false;
        }
    }
    if (!flag) {
        printf("Masukan tidak valid, masukkan hanya angka (2-10) atau huruf (A, J, Q, K)!\n");
    }
}

while (!flag) {
    flag = true;
    printf("Masukkan 4 buah angka: \n");
    printf(">> ");
    count1 = 0;
    while (count1 < 4 && scanf("%s", num) != EOF) {
        strcpy(num_array[count1], num);
        count1++;
    }
    for (int count2 = 0; count2 < 4; count2++) {
        if (!isInt(num_array[count2]) && !isAlphabet(num_array[count2])) {
            flag = false;
        }
    }
    if (!flag) {
        printf("Masukan tidak valid, masukkan hanya angka (2-10) atau huruf (A, J, Q, K)!\n");
    }
}
```

```
start = clock();
count = 0;
for (i = 0; i < 4; i++) {
    a = nums[i];
    remove_element(nums, i, buffer3, 4);
    for (j = 0; j < 3; j++) {
        b = buffer3[j];
        remove_element(buffer3, j, buffer2, 3);
        for (k = 0; k < 2; k++) {
            c = buffer2[k];
            remove_element(buffer2, k, buffer, 2);
            d = buffer[0];
            for (x = 0; x < 4; x++) {
                op1 = ops[x];
                for (y = 0; y < 4; y++) {
                    op2 = ops[y];
                    for (z = 0; z < 4; z++) {
                        op3 = ops[z];
                        if ((eval(eval(eval(a, b, op1), c, op2), d, op3)) == 24) {
                            sprintf(solutions[count], "((%d %c %d) %c %d) %c %d\n", a, op1, b, op2, c, op3, d);
                            count++;
                        }
                        if (eval(eval(a, b, op1), eval(c, d, op3), op2) == 24) {
                            sprintf(solutions[count], "(%d %c %d) %c (%d %c %d)\n", a, op1, b, op2, c, op3, d);
                            count++;
                        }
                        if (eval(eval(a, eval(b, c, op2), op1), d, op3) == 24) {
                            sprintf(solutions[count], "(%d %c (%d %c %d)) %c %d\n", a, op1, b, op2, c, op3, d);
                            count++;
                        }
                        if (eval(a, (eval(eval(b, c, op2), d, op3)), op1) == 24) {
                            sprintf(solutions[count], "%d %c ((%d %c %d) %c %d)\n", a, op1, b, op2, c, op3, d);
                            count++;
                        }
                        if (eval(a, (eval(b, eval(c, d, op3), op2)), op1) == 24) {
                            sprintf(solutions[count], "%d %c (%d %c (%d %c %d))\n", a, op1, b, op2, c, op3, d);
                            count++;
                        }
                    }
                }
            }
        }
    }
}
end = clock();
execution_time = ((float)(end - start))/CLOCKS_PER_SEC;
```

```c
    if (count == 0) {
        printf("Tidak dapat menemukan solusi:(\n");
        printf("Execution time: %f seconds\n", execution_time);
    } else {
        printf("Banyaknya solusi yang ditemukan: %d\n", count);
        for (int i = 0; i < count; i++) {
            printf("%d. ", i+1);
            printf("%s", solutions[i]);
        }
        printf("Execution time: %f seconds\n", execution_time);
        printf("Apakah ingin menyimpan solusi?\n");
        printf("1. Ya\n");
        printf("2. Tidak\n");
        printf(">> ");
        scanf("%d", &pilihan);

        if (pilihan == 1) {
            printf("Masukkan nama file dengan ekstensi .txt\n");
            printf(">> ");
            scanf("%s", txt);
            path = "..\\test\\";
            i = 0;
            while (path[i] != '\0') {
                file[i] = path[i];
                i++;
            }
            j = 0;
            while (txt[j] != '\0') {
                file[i] = txt[j];
                i++;
                j++;
            }
            file[i] = '\0';
            printf("%s", file);

            fptr = fopen(file, "w");
            for (int i = 0; i < count; i++) {
                fprintf(fptr, "%s", solutions[i]);
            }
            fclose(fptr);
        } else if (pilihan == 2) {
            printf("Terima kasih telah menggunakan 24 game solver:)");
        }
    }
}
```

3.2.2 operations.c

```c
#include <stdio.h>
#include "boolean.h"
#include <string.h>

float eval(float a, float b, char op) {
/* Menghitung hasil dari operasi float a dengan float b dengan operator char op */
    if (op == '+') {
        return a + b;
    } else if (op == '-') {
        return a - b;
    } else if (op == '/') {
        return a / b;
    } else if (op == '*') {
        return a * b;
    }
}

void remove_element(int arr[], int index, int buffer[], int size) {
/* Prekondisi: arr memiliki elemen dan tidak kosong */
/* F.S.: terbentuk array buffer yang berisi elemen array arr tanpa num di dalamnya */
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (i != index) {
            buffer[count] = arr[i];
            count++;
        }
    }
}

boolean same(char *s1 , char *s2) {
/* Menghasilkan true jika string s1 sama dengan string s2 */
    if (strlen(s1) == strlen(s2)) {
        for (int i = 0; i < strlen(s1); i++) {
            if (s1[i] != s2[i]) {
                return false;
            }
        }
        return true;
    }
    return false;
}
```

```
boolean isAlphabet(char num[]) {
/* Menghasilkan true jika x adalah huruf A, J, Q, atau K, false jika tidak */
    if (same(num, "A") ||
        same(num, "J")||
        same(num, "Q")||
        same(num, "K")) {
        return true;
    }
    return false;
}

boolean isInt(char num[]) {
/* Menghasilkan true jika x adalah angka (2-10)*/
    if (same(num, "2") ||
        same(num, "3")||
        same(num, "4")||
        same(num, "5")||
        same(num, "6")||
        same(num, "7")||
        same(num, "8")||
        same(num, "9")||
        same(num, "10")) {
            return true;
        }
    return false;
}
```

```
int transformToInt(char num[]) {
    if (same(num, "2")) {
        return 2;
    } else if (same(num, "3")) {
        return 3;
    } else if (same(num, "4")) {
        return 4;
    } else if (same(num, "5")) {
        return 5;
    } else if (same(num, "6")) {
        return 6;
    } else if (same(num, "7")) {
        return 7;
    } else if (same(num, "8")) {
        return 8;
    } else if (same(num, "9")) {
        return 9;
    } else if (same(num, "10")) {
        return 10;
    }
}

int transformToNumber(char num[]) {
/* Menghasilkan nilai integer dari word x yang berupa huruf A, J, Q, atau K */
/* A = 1
   J = 11
   Q = 12
   K = 13 */
    if (same(num, "A")) {
        return 1;
    } else if (same(num, "J")) {
        return 11;
    } else if (same(num, "Q")) {
        return 12;
    } else if (same(num, "K")) {
        return 13;
    }
}
```

3.2.3 boolean.h

```
/* Definisi type boolean */

#ifndef _BOOLEAN_h
#define _BOOLEAN_h

#define boolean unsigned char
#define true 1
#define false 0

#endif
```

BAB 4

MASUKAN DAN LUARAN PROGRAM

Input	Output
Pilih cara menentukan 4 angka: 1. Program otomatis mengenerate angka 2. Masukan manual dari pengguna/user >> 1 4 angka yang ter-generate: 12 4 8 2 **test01.txt**	Banyaknya solusi yang ditemukan: 61 (12 - 4) + (8 * 2) 12 - (4 - (8 * 2)) ((12 + 4) * 2) - 8 (12 - 4) + (2 * 8) 12 - (4 - (2 * 8)) ((12 - 4) * 2) + 8 (12 - 8) * (4 + 2) 12 * (8 - (4 + 2)) 12 * ((8 - 4) - 2) (12 * (8 - 4)) / 2 12 * ((8 - 4) / 2) (12 + (8 * 2)) - 4 12 + ((8 * 2) - 4) ((12 - 8) + 2) * 4 (12 - (8 - 2)) * 4 (12 - 8) * (2 + 4) 12 * (8 - (2 + 4)) 12 * ((8 - 2) - 4) ((12 + 2) - 8) * 4 (12 + (2 - 8)) * 4 (12 + (2 * 8)) - 4 12 + ((2 * 8) - 4) 12 / (2 / (8 - 4)) (12 / 2) * (8 - 4) 4 * ((12 - 8) + 2) 4 * (12 - (8 - 2)) ((4 + 12) * 2) - 8 4 * ((12 + 2) - 8) 4 * (12 + (2 - 8)) (4 + 2) * (12 - 8) 4 * ((2 + 12) - 8) 4 * (2 + (12 - 8)) 4 * ((2 - 8) + 12) 4 * (2 - (8 - 12)) 8 + ((12 - 4) * 2) 8 - ((4 - 12) * 2) ((8 - 4) * 12) / 2 (8 - 4) * (12 / 2) (8 - (4 + 2)) * 12 ((8 - 4) - 2) * 12 ((8 - 4) / 2) * 12 8 + (2 * (12 - 4)) ((8 * 2) + 12) - 4 (8 * 2) + (12 - 4) (8 - (2 + 4)) * 12 ((8 - 2) - 4) * 12 8 - (2 * (4 - 12)) ((8 * 2) - 4) + 12 (8 * 2) - (4 - 12) (2 * (12 + 4)) - 8 (2 * (12 - 4)) + 8 ((2 + 12) - 8) * 4 (2 + (12 - 8)) * 4 (2 + 4) * (12 - 8) (2 * (4 + 12)) - 8 ((2 - 8) + 12) * 4 (2 - (8 - 12)) * 4 ((2 * 8) + 12) - 4 (2 * 8) + (12 - 4) ((2 * 8) - 4) + 12 (2 * 8) - (4 - 12)

```
Execution time: 0.000000 seconds
Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> 1
Masukkan nama file dengan ekstensi .txt
>> test01.txt
```

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 2
Masukkan 4 buah angka:
>> A 8 9 Q
4 angka masukan pengguna: 1 8 9 12
```

test02.txt

```
Banyaknya solusi yang ditemukan: 47
1. ((1 - 8) + 9) * 12
2. (1 - (8 - 9)) * 12
3. (1 * 8) * (12 - 9)
4. 1 * (8 * (12 - 9))
5. ((1 + 9) - 8) * 12
6. (1 + (9 - 8)) * 12
7. ((1 * 12) - 9) * 8
8. (1 * (12 - 9)) * 8
9. 1 * ((12 - 9) * 8)
10. (8 / 1) * (12 - 9)
11. (8 * 1) * (12 - 9)
12. 8 * ((1 * 12) - 9)
13. 8 * (1 * (12 - 9))
14. 8 * (12 - (1 * 9))
15. 8 * ((12 / 1) - 9)
16. 8 * ((12 * 1) - 9)
17. (8 * (12 - 9)) / 1
18. 8 * ((12 - 9) / 1)
19. 8 * (12 - (9 / 1))
20. (8 * (12 - 9)) * 1
21. 8 * ((12 - 9) * 1)
22. 8 * (12 - (9 * 1))
23. ((9 + 1) - 8) * 12
24. (9 + (1 - 8)) * 12
25. ((9 - 8) + 1) * 12
26. (9 - (8 - 1)) * 12
27. 12 * ((1 - 8) + 9)
28. 12 * (1 - (8 - 9))
29. (12 - (1 * 9)) * 8
30. ((12 / 1) - 9) * 8
31. 12 * ((1 + 9) - 8)
32. 12 * (1 + (9 - 8))
33. ((12 * 1) - 9) * 8
34. (12 - 9) / (1 / 8)
35. ((12 - 9) / 1) * 8
36. (12 - (9 / 1)) * 8
37. ((12 - 9) * 1) * 8
38. (12 - 9) * (1 * 8)
39. (12 - (9 * 1)) * 8
40. 12 * ((9 + 1) - 8)
41. 12 * (9 + (1 - 8))
42. ((12 - 9) * 8) / 1
43. (12 - 9) * (8 / 1)
44. ((12 - 9) * 8) * 1
45. (12 - 9) * (8 * 1)
46. 12 * ((9 - 8) + 1)
47. 12 * (9 - (8 - 1))
Execution time: 0.000000 seconds
```

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 2
Masukkan 4 buah angka:
>> A Q K J
4 angka masukan pengguna: 1 12 13 11
```

test03.txt

```
Banyaknya solusi yang ditemukan: 31
1. (1 * 12) * (13 - 11)
2. 1 * (12 * (13 - 11))
3. ((1 * 13) - 11) * 12
4. (1 * (13 - 11)) * 12
5. 1 * ((13 - 11) * 12)
6. 12 / (1 / (13 - 11))
7. (12 / 1) * (13 - 11)
8. (12 * 1) * (13 - 11)
9. 12 * ((1 * 13) - 11)
10. 12 * (1 * (13 - 11))
11. 12 * (13 - (1 * 11))
12. 12 * ((13 / 1) - 11)
13. 12 * ((13 * 1) - 11)
14. (12 * (13 - 11)) / 1
15. 12 * ((13 - 11) / 1)
16. 12 * (13 - (11 / 1))
17. (12 * (13 - 11)) * 1
18. 12 * ((13 - 11) * 1)
19. 12 * (13 - (11 * 1))
20. (13 - (1 * 11)) * 12
21. ((13 / 1) - 11) * 12
22. ((13 * 1) - 11) * 12
23. ((13 - 11) / 1) * 12
24. (13 - (11 / 1)) * 12
25. ((13 - 11) * 1) * 12
26. (13 - 11) * (1 * 12)
27. (13 - (11 * 1)) * 12
28. ((13 - 11) * 12) / 1
29. (13 - 11) * (12 / 1)
30. ((13 - 11) * 12) * 1
31. (13 - 11) * (12 * 1)
Execution time: 0.000000 seconds
```

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 2
Masukkan 4 buah angka:
>> 10 J 4 Q
4 angka masukan pengguna: 10 11 4 12
```

test04.txt

```
Banyaknya solusi yang ditemukan: 12
1. (10 + 11) + (12 / 4)
2. 10 + (11 + (12 / 4))
3. (10 + (12 / 4)) + 11
4. 10 + ((12 / 4) + 11)
5. (11 + 10) + (12 / 4)
6. 11 + (10 + (12 / 4))
7. (11 + (12 / 4)) + 10
8. 11 + ((12 / 4) + 10)
9. ((12 / 4) + 10) + 11
10. (12 / 4) + (10 + 11)
11. ((12 / 4) + 11) + 10
12. (12 / 4) + (11 + 10)
Execution time: 0.000000 seconds
```

<table>
<tr>
<td>

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 1
4 angka yang ter-generate: 10 2 12 9
```

test05.txt

</td>
<td>

```
Banyaknya solusi yang ditemukan: 23
1. (10 - 2) * (12 - 9)
2. ((10 - 9) * 2) * 12
3. (10 - 9) * (2 * 12)
4. ((10 - 9) * 12) * 2
5. (10 - 9) * (12 * 2)
6. (2 - 10) * (9 - 12)
7. (2 / (10 - 9)) * 12
8. (2 * (10 - 9)) * 12
9. 2 * ((10 - 9) * 12)
10. (2 * 12) / (10 - 9)
11. 2 * (12 / (10 - 9))
12. (2 * 12) * (10 - 9)
13. 2 * (12 * (10 - 9))
14. 12 / ((10 - 9) / 2)
15. (12 / (10 - 9)) * 2
16. (12 * (10 - 9)) * 2
17. 12 * ((10 - 9) * 2)
18. (12 * 2) / (10 - 9)
19. 12 * (2 / (10 - 9))
20. (12 * 2) * (10 - 9)
21. 12 * (2 * (10 - 9))
22. (12 - 9) * (10 - 2)
23. (9 - 12) * (2 - 10)
Execution time: 0.000000 seconds
```

</td>
</tr>
<tr>
<td>

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 2
Masukkan 4 buah angka:
>> 10 9 8 7
4 angka masukan pengguna: 10 9 8 7
```

test06.txt

</td>
<td>

```
Banyaknya solusi yang ditemukan: 5
1. 9 / ((10 - 7) / 8)
2. (9 / (10 - 7)) * 8
3. (9 * 8) / (10 - 7)
4. (8 * 9) / (10 - 7)
5. 8 * (9 / (10 - 7))
Execution time: 0.000000 seconds
```

</td>
</tr>
<tr>
<td>

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 1
4 angka yang ter-generate: 11 10 7 1
```

</td>
<td>

```
Tidak ada solusi!
Execution time: 0.000000 seconds
```

</td>
</tr>
<tr>
<td>

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 2
Masukkan 4 buah angka:
>> A Z X Y
```

</td>
<td>

```
Masukan tidak valid, masukkan hanya angka (2-10) atau huruf (A, J, Q, K)!
Masukkan 4 buah angka:
>>
```

</td>
</tr>
<tr>
<td>

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 2
Masukkan 4 buah angka:
>> 1 14 13 12
```

</td>
<td>

```
Masukan tidak valid, masukkan hanya angka (2-10) atau huruf (A, J, Q, K)!
Masukkan 4 buah angka:
>>
```

</td>
</tr>
</table>

```
Pilih cara menentukan 4 angka:
1. Program otomatis mengenerate angka
2. Masukan manual dari pengguna/user
>> 2
Masukkan 4 buah angka:
>> 2 3 4 5 6
Hanya masukan sejumlah 4 buah angka atau huruf!
Masukkan 4 buah angka:
>> 2 3 4 5
4 angka masukan pengguna: 2 3 4 5
```

test07.txt

```
1. 2 * ((3 + 4) + 5)
2. 2 * (3 + (4 + 5))
3. 2 * ((3 + 5) + 4)
4. 2 * (3 + (5 + 4))
18. (3 + (5 - 2)) * 4
19. ((3 + 5) + 4) * 2
20. (3 + (5 + 4)) * 2
21. 4 * ((3 - 2) + 5)
22. 4 * (3 - (2 - 5))
23. ((4 + 3) + 5) * 2
24. (4 + (3 + 5)) * 2
25. 4 * ((3 + 5) - 2)
26. 4 * (3 + (5 - 2))
27. 4 * ((5 - 2) + 3)
28. 4 * (5 - (2 - 3))
29. ((4 + 5) + 3) * 2
30. (4 + (5 + 3)) * 2
31. 4 * ((5 + 3) - 2)
32. 4 * (5 + (3 - 2))
33. ((5 - 2) + 3) * 4
34. (5 - (2 - 3)) * 4
35. ((5 + 3) - 2) * 4
36. (5 + (3 - 2)) * 4
37. ((5 + 3) + 4) * 2
38. (5 + (3 + 4)) * 2
39. ((5 + 4) + 3) * 2
40. (5 + (4 + 3)) * 2
Execution time: 0.001000 seconds
Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> 1
Masukkan nama file dengan ekstensi .txt
>> test07.txt
..\test\test07.txt
```

REFERENSI

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/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf

LAMPIRAN

Checklist Fitur

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 dapat memenuhi (berhasil mencapai 24)		
5. Program dapat menyimpan solusi dalam file teks		