

LAPORAN TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA



Disusun oleh

Michael Utama (13521137)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

DAFTAR ISI.....	i
BAB I: DESKRIPSI MASALAH.....	1
BAB II: ALGORITMA BRUTE FORCE	3
BAB III: IMPLEMENTASI ALGORITMA	4
BAB IV: PENGUJIAN	10
SUMBER	15
LAMPIRAN.....	16

BAB I

DESKRIPSI MASALAH

Permainan 24 merupakan sebuah permainan kartu aritmatika yang memiliki tujuan membentuk angka 24 dari empat kartu acak. Permainan 24 pada versi ini menggunakan 13 kartu, yaitu As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King. Kartu bernomor memiliki nilai sesuai nomornya, sedangkan kartu As bernilai 1, Jack bernilai 11, Queen bernilai 12, dan King bernilai 13. Operasi yang dapat digunakan adalah penjumlahan(+), pengurangan(-), perkalian(\times), dan pembagian(\div).

Spesifikasi Tugas Kecil 1:

- Tulislah program sederhana dalam Bahasa C/C++/Java yang mengimplementasikan algoritma Brute Force untuk mencari seluruh solusi permainan kartu 24.
- **Input:** 4 angka/huruf yang terdiri dari: (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K).
Contoh input: A 8 9 Q

Selain itu, input juga dapat dilakukan dengan program men-generate 4 angka/hurufnya sendiri secara random. Pengguna dapat memilih apakah program meminta input dari pengguna atau generate sendiri. Apabila masukan tidak sesuai, maka program menampilkan luaran “Masukan tidak sesuai” dan akan meminta ulang.

- **Output:**
 1. Banyaknya solusi yang ditemukan.
 2. Solusi dari permainan kartu 24 ditampilkan di layar dan terdapat opsi untuk menyimpan solusi dalam file text. Untuk contoh kasus A 8 9 Q, maka salah satu solusinya adalah:
 $((9 + A) - 8) * Q$ atau $((9 + 1) - 8) * 12$
(Kedua jenis output dibebaskan)

Note: Format penulisan output yang dicetak tidak harus persis contoh, yang penting merepresentasikan solusinya sudah cukup. Output apabila tidak ada solusi untuk pasangan kombinasi input, cukup ditampilkan “Tidak ada solusi”. Untuk solusi setiap masukan, perlu dipertimbangkan urutan nilai (x1..x4), urutan operator, dan grouping dengan kurung yang mungkin.

Gambaran apabila terdapat solusi

A 8 9 Q
38 solutions found ((1 - 8) + 9) * 12 (1 - (8 - 9)) * 12 (1 * 8) * (12 - 9) (dst.)

Di akhir, program akan menanyakan “Apakah ingin menyimpan solusi?”. Jika iya, program akan meminta sebuah nama untuk file teks dan semua solusi yang didapat akan disimpan dalam file text tersebut.

3. Waktu eksekusi program (tidak termasuk waktu pembacaan file input). Luaran no 1 cukup ditampilkan pertama sebelum mencetak solusi, sementara luaran no 3 cukup ditampilkan di akhir program (saat program selesai)

BAB II

ALGORITMA BRUTE FORCE

Untuk menyelesaikan persoalan kartu 24, penulis menggunakan algoritma *brute force* dengan langkah-langkah:

1. Asumsikan pada langkah ini program sudah mendapat input valid.
2. Bangkitkan seluruh permutasi dari empat kartu pada input.
3. Untuk tiap permutasi dari langkah 2, bangkitkan seluruh kemungkinan pasangan operasi.
4. Untuk tiap permutasi kartu dan kemungkinan operasi dari langkah 3, pasangkan dengan lima kemungkinan urutan operasi. Urutan operasi yang mungkin adalah
 - a. $a \text{ op } (b \text{ op } (c \text{ op } d))$
 - b. $(a \text{ op } (b \text{ op } c)) \text{ op } d$
 - c. $((a \text{ op } b) \text{ op } c) \text{ op } d$
 - d. $a \text{ op } ((b \text{ op } c) \text{ op } d)$
 - e. $(a \text{ op } b) \text{ op } (c \text{ op } d)$
5. Evaluasi tiap operan dan operasi pada langkah 4. Jika hasilnya adalah 24, masukkan ke *list* berisi jawaban.

Konsep algoritma *brute force* digunakan pada langkah 2-4 untuk membangkitkan seluruh ekspresi matematika yang mungkin dari keempat kartu, lalu dievaluasi pada langkah ke-5.

BAB III

IMPLEMENTASI ALGORITMA

Algoritma *brute force* pada bab 2 diimplementasi menggunakan bahasa c++. Kode program terdiri atas satu file solution.cpp, yang terdiri atas beberapa prosedur dan fungsi:

- Variabel dan konstanta global

```
using namespace std;

string daftarOperasi = "+-/*";
string kartuValid[] = {"A", "2", "3", "4", "5", "6", "7", "8",
"9", "10", "J", "Q", "K"};
map<string, int> daftarNilaiKartu;

const int JENIS_KARTU = 13;
const double EPS = 1e-6;
```

- Prosedur init()

Bertugas mengisi map daftarNilaiKartu dan melakukan randomize

```
void init() {
    // kartu angka 2-9
    string tmp = "1";
    for(int i = 2; i <= 9; i++) {
        tmp[0] = i + '0';
        daftarNilaiKartu[tmp] = i;
    }

    // kartu angka 10
    daftarNilaiKartu["10"] = 10;

    // kartu huruf senilai 10
    daftarNilaiKartu["J"] = 11;
    daftarNilaiKartu["Q"] = 12;
    daftarNilaiKartu["K"] = 13;

    // kartu as
    daftarNilaiKartu["A"] = 1;

    // randomize
    srand(time(NULL));
}
```

- Fungsi randomCard()

Mengembalikan sebuah kartu random

```
string randomCard() {
    return kartuValid[rand() % JENIS_KARTU];
}
```

```
}
```

- Fungsi isKartuValid

Mengembalikan true jika string c adalah kartu valid

```
bool isKartuValid(string c) {  
    for(auto x : kartuValid) {  
        if(c == x) return true;  
    }  
    return false;  
}
```

- Fungsi isSame

Mengembalikan true jika selisih a dan b kurang dari EPS

```
bool isSame (double a, double b) {  
    return (a < b + EPS) && (b < a + EPS);  
}
```

- Fungsi operate

Mengembalikan hasil a op b

```
double operate(double a, double b, char op) {  
    if(op == '+') return a + b;  
    if(op == '-') return a - b;  
    if(op == '*') return a * b;  
    if(op == '/') return a / b;  
    assert(0);  
    return -1;  
}
```

- Procedure hitungJawaban

Mengisi vector ans dengan jawaban menggunakan urutan operasi dan kartu.

```
void hitungJawaban(vector<char> &op, vector<int> &nilaiKartu,  
vector<string> &ans) {  
    double a = nilaiKartu[0], b = nilaiKartu[1], c =  
    nilaiKartu[2], d = nilaiKartu[3];  
    // Lakukan seluruh kemungkinan urutan  
    // ((a op b) op c) op d  
    if (isSame(operate(operate(operate(a, b, op[0]), c, op[1]),  
d, op[2]), 24)) {  
        stringstream ss;  
        ss << "(" << a << " " << op[0] << " " << b << " " <<  
op[1] << " " << c << " " << op[2] << " " << d << "";  
        ans.push_back(ss.str());  
    }  
    // (a op (b op c)) op d  
    if (isSame(operate(operate(a, operate(b, c, op[1]), op[0]),  
d, op[2]), 24)) {  
        stringstream ss;  
        ss << "(" << a << " " << op[0] << " (" << b << " " <<  
op[1] << " " << c << " " << op[2] << " " << d << "";
```

```

        ans.push_back(ss.str());
    }
    // a op (b op (c op d))
    if (isSame(operate(a, operate(b, operate(c, d, op[2]),
op[1]), op[0]), 24)) {
        stringstream ss;
        ss << "" << a << " " << op[0] << " (" << b << " " <<
op[1] << " (" << c << " " << op[2] << " " << d << ")");
        ans.push_back(ss.str());
    }
    // a op ((b op c) op d)
    if (isSame(operate(a, operate(operate(b, c, op[1]), d,
op[2]), op[0]), 24)) {
        stringstream ss;
        ss << "" << a << " " << op[0] << " (" << b << " " <<
op[1] << " " << c << ") " << op[2] << " " << d << ")";
        ans.push_back(ss.str());
    }
    // (a op b) op (c op d)
    if (isSame(operate(operate(a, b, op[0]), operate(c, d,
op[2]), op[1]), 24)) {
        stringstream ss;
        ss << "(" << a << " " << op[0] << " " << b << ") " <<
op[1] << " (" << c << " " << op[2] << " " << d << ")";
        ans.push_back(ss.str());
    }
}
}

```

- Procedure searchKemungkinanOperasi

Membangkitkan seluruh kemungkinan urutan operasi untuk dipakai dalam prosedur hitungJawaban

```

void searchKemungkinanOperasi(int opNum, vector<char> op,
vector<int> &nilaiKartu, vector<string> &ans) {
    if(opNum == 3) {
        hitungJawaban(op, nilaiKartu, ans);
        return;
    }
    op.push_back('*');
    for(char c: daftarOperasi) {
        op[opNum] = c;
        searchKemungkinanOperasi(opNum+1, op, nilaiKartu, ans);
    }
}

```

- Procedure permutasiNilaiKartu

Membangkitkan seluruh kemungkinan permutasi nilai kartu untuk dipakai dalam prosedur searchKemungkinanOperasi

```

void permutasiNilaiKartu(int idx, vector<int> nilaiKartu,
vector<int> jumlahKartu, vector<string> &ans) {
    if(idx == 4) {

```



```

        searchKemungkinanOperasi(0, vector<char>(0), nilaiKartu,
ans);
        return;
    }
    nilaiKartu.push_back(0);
    for (int i = 1; i <= JENIS_KARTU; i++) {
        if (jumlahKartu[i]) {
            jumlahKartu[i]--;
            nilaiKartu[idx] = i;
            permutasiNilaiKartu(idx + 1, nilaiKartu, jumlahKartu,
ans);
            jumlahKartu[i]++;
        }
    }
}

```

- **Procedure output**

output banyak jawaban, jawaban, dan runtime dalam milisekon ke sebuah output stream

```

void output(ostream& output_stream, const vector<string>& ans,
chrono::milliseconds duration) {
    if (ans.size() == 0) {
        output_stream << "no solution found\n";
    }
    else if (ans.size() == 1) {
        output_stream << ans.size() << " solution found\n";
    }
    else {
        output_stream << ans.size() << " solutions found\n";
    }

    for(string s: ans) {
        output_stream << s << '\n';
    }
    output_stream<< "Execution time = " << duration.count() << "
milliseconds" << endl;
}

```

- **Fungsi main**

fungsi utama program

```

int32_t main(int argc, char *argv[]) {
    init();
    string path_prefix = "test/";
    bool strictCommandOutput = false;

    // Arguments, for testing purpose
    if(argc == 2 || argc > 3) {
        cout << "Invalid Argument count!" << endl;
        return 0;
    }
}

```

```

else if(argc == 3) {
    string path_input = path_prefix, path_output =
path_prefix;
    path_input.append(argv[1]);
    path_output.append(argv[2]);
    if (strcmp(argv[1], "cin")) {
        freopen(path_input.c_str(), "r", stdin);
    }
    if (strcmp(argv[2], "cout")) {
        freopen(path_output.c_str(), "w", stdout);
    }
    strictCommandOutput = true;
}

// Input
vector<string> kartu(4);
vector<int> jumlahKartu(JENIS_KARTU + 1, 0);
bool isRandom = false;
while (true) {
    jumlahKartu = vector<int>(JENIS_KARTU + 1, 0);
    string input_string;
    getline(cin, input_string);
    stringstream input_stream(input_string);

    bool invalid_input = false;
    for(int i = 0; i < 4; i++) {
        if (!(input_stream >> kartu[i])) {
            // validasi input < 4
            invalid_input = true;
            break;
        }
        if (kartu[i] == "*") {
            // random kartu
            kartu[i] = randomCard();
            isRandom = true;
        }
        if (!isKartuValid(kartu[i])) {
            // validasi input bukan jenis kartu
            invalid_input = true;
            break;
        }
        jumlahKartu[daftarNilaiKartu[kartu[i]]]++;
    }

    string tmp;
    if (invalid_input || input_stream >> tmp) {
        cout << "Masukan tidak sesuai\n";
        continue;
    }

    // lolos seluruh validasi, keluar dari loop
    break;
}

```

```

        if (isRandom) {
            cout << "Kartu : " << kartu[0] << ' ' << kartu[1] << ' '
<< kartu[2] << ' ' << kartu[3] << '\n';
        }

        // Selesai input, mulai menghitung waktu eksekusi program
        auto start = chrono::high_resolution_clock::now();

        vector<string> ans;
        permutasiNilaiKartu(0, vector<int>(), jumlahKartu, ans);

        auto end = chrono::high_resolution_clock::now();
        auto duration =
chrono::duration_cast<chrono::milliseconds>(end - start);

        output(cout, ans, duration);

        if (!strictCommandOutput) {
            while(true) {
                cout << "Apakah ingin menyimpan solusi?(Y/N)\n";
                string response;
                cin >> response;
                if (response == "Y" || response == "y") {
                    cout << "Nama file (contoh: testcase3.out) : ";
                    string nama_file;
                    cin >> nama_file;
                    string path_output = path_prefix + nama_file;
                    ofstream output_stream(path_output);

                    output_stream << "Kartu : " << kartu[0] << ' ' <<
kartu[1] << ' ' << kartu[2] << ' ' << kartu[3] << '\n';
                    output(output_stream, ans, duration);

                    cout << "Solusi sudah dimasukkan ke " <<
path_output << '\n';
                    break;
                }
                else if (response == "N" || response == "n") {
                    break;
                }
            }
        }
    }
}

```

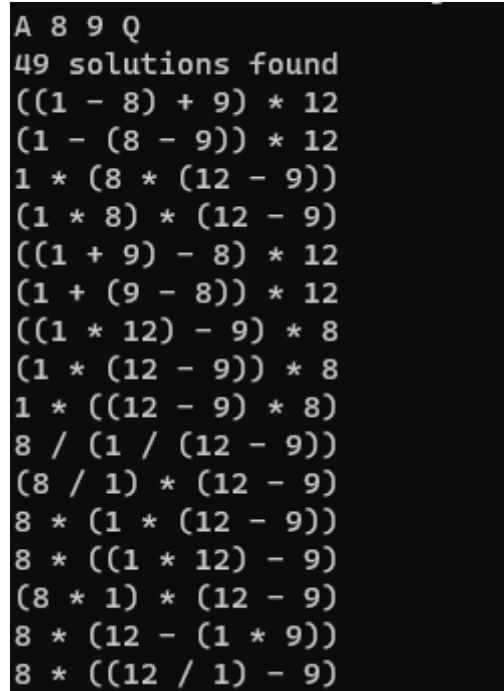
BAB IV

PENGUJIAN

1. Kasus uji 1

File terkait: tc1.in, tc1.out

Kasus yang diuji adalah kasus contoh pada spesifikasi tugas.



```
A 8 9 Q
49 solutions found
((1 - 8) + 9) * 12
(1 - (8 - 9)) * 12
1 * (8 * (12 - 9))
(1 * 8) * (12 - 9)
((1 + 9) - 8) * 12
(1 + (9 - 8)) * 12
((1 * 12) - 9) * 8
(1 * (12 - 9)) * 8
1 * ((12 - 9) * 8)
8 / (1 / (12 - 9))
(8 / 1) * (12 - 9)
8 * (1 * (12 - 9))
8 * ((1 * 12) - 9)
(8 * 1) * (12 - 9)
8 * (12 - (1 * 9))
8 * ((12 / 1) - 9)
```

Gambar 1. Masukan dan Sebagian Keluaran Kasus Uji 1

Terdapat diskrepansi antara jawaban pada spesifikasi tugas dan jawaban program penulis. Setelah diteliti, perbedaan jawaban disebabkan:

- a. Ekspresi yang dikerjakan dari belakang ($a \text{ op } (b \text{ op } (c \text{ op } d))$) tidak dihitung pada website referensi, tetapi dihitung pada program penulis.
- b. Penanganan bilangan real berbeda, mengakibatkan jawaban $8 / ((12 / 9) - 1)$ tidak terdeteksi pada website referensi.

2. Kasus uji 2

File terkait: tc2.in, tc2.out

Kasus yang diuji memiliki sedikit jawaban (sumber:

<https://samidavies.wordpress.com/2017/04/14/the-hardest-games-of-24/>)

```

5 2 3 Q
1 solution found
12 / (3 - (5 / 2))
Execution time = 3 milliseconds
Apakah ingin menyimpan solusi?(Y/N)
y
Nama file (contoh: testcase3.out) : tc2.out
Solusi sudah dimasukkan ke test/tc2.out

```

Gambar 2. Masukan dan Keluaran Kasus Uji 2

Kasus uji ini merupakan salah satu kasus yang menunjukkan perlunya pengecekan ekspresi yang dikerjakan dari belakang.

3. Kasus uji 3

File terkait: tc3.in, tc3.out

Kasus yang diuji tidak memiliki jawaban

```

A 2 2 A
no solution found
Execution time = 0 milliseconds
Apakah ingin menyimpan solusi?(Y/N)
y
Nama file (contoh: testcase3.out) : tc3.out
Solusi sudah dimasukkan ke test/tc3.out

```

Gambar 3. Masukan dan Keluaran Kasus Uji 3

4. Kasus uji 4

File terkait: tc4.in, tc4.out

Kasus yang diuji memiliki banyak jawaban

```

A 2 3 4
242 solutions found
((1 + 2) + 3) * 4
(1 + (2 + 3)) * 4
((1 * 2) * 3) * 4
(1 * (2 * 3)) * 4
1 * (2 * (3 * 4))
1 * ((2 * 3) * 4)
(1 * 2) * (3 * 4)
((1 * 2) * 4) * 3
(1 * (2 * 4)) * 3
1 * (2 * (4 * 3))
1 * ((2 * 4) * 3)
(1 * 2) * (4 * 3)

```

Gambar 4. Masukan dan Sebagian Keluaran Kasus Uji 4

```

4 * (3 * (2 / 1))
4 * ((3 * 2) / 1)
(4 * 3) * (2 / 1)
((4 * 3) * 2) * 1
(4 * (3 * 2)) * 1
4 * (3 * (2 * 1))
4 * ((3 * 2) * 1)
(4 * 3) * (2 * 1)
Execution time = 1 milliseconds
Apakah ingin menyimpan solusi?(Y/N)
y
Nama file (contoh: testcase3.out) : tc4.out
Solusi sudah dimasukkan ke test/tc4.out

```

Gambar 5. Bagian Akhir Keluaran Kasus Uji 4

5. Kasus uji 5

File terkait: tc5.in, tc5.out

Terdapat kesalahan input pada kasus uji

```

1 2 2 1
Masukan tidak sesuai
A 2 3 4 5 6 7 8
Masukan tidak sesuai
A A
Masukan tidak sesuai
A A A B
Masukan tidak sesuai
21 A B C
Masukan tidak sesuai
A A A A
no solution found
Execution time = 0 milliseconds
Apakah ingin menyimpan solusi?(Y/N)
tidak
Apakah ingin menyimpan solusi?(Y/N)
y
Nama file (contoh: testcase3.out) : tc5.out
Solusi sudah dimasukkan ke test/tc5.out

```

Gambar 6. Masukan dan Keluaran Kasus Uji 5

Pada kasus uji, terdapat masukan yang salah seperti karakter 1 (seharusnya A), jumlah masukan tidak sama dengan 4, dan respon menyimpan solusi bukan 'y'/'Y' atau 'n'/'N'. Seluruh masalah pada kasus uji 5 dapat ditangani oleh program.

6. Kasus uji 6

File terkait: tc6.in, tc6.out

Kasus yang diuji sebagian random

```
8 * * 8
Kartu : 8 3 6 8
24 solutions found
(3 + (8 / 8)) * 6
(6 / (3 / 8)) + 8
((6 / 3) * 8) + 8
6 * (3 + (8 / 8))
(6 * 8) - (3 * 8)
((6 * 8) / 3) + 8
(6 * (8 / 3)) + 8
(6 * 8) - (8 * 3)
6 * ((8 / 8) + 3)
(8 / (3 / 6)) + 8
((8 / 3) * 6) + 8
8 + (6 / (3 / 8))
8 + ((6 / 3) * 8)
(8 * 6) - (3 * 8)
((8 * 6) / 3) + 8
(8 * (6 / 3)) + 8
8 + (6 * (8 / 3))
8 + ((6 * 8) / 3)
(8 * 6) - (8 * 3)
8 + (8 / (3 / 6))
8 + ((8 / 3) * 6)
((8 / 8) + 3) * 6
8 + (8 * (6 / 3))
8 + ((8 * 6) / 3)
Execution time = 0 milliseconds
Apakah ingin menyimpan solusi?(Y/N)
y
```

Gambar 7. Masukan dan Keluaran Kasus Uji 6

7. Kasus uji random 1

File terkait: tc_rand1.out

```

* * * *
Kartu : A 9 Q 10
16 solutions found
((1 - 9) + 10) * 12
(1 - (9 - 10)) * 12
((1 + 10) - 9) * 12
(1 + (10 - 9)) * 12
((10 + 1) - 9) * 12
(10 + (1 - 9)) * 12
((10 - 9) + 1) * 12
(10 - (9 - 1)) * 12
12 * ((1 - 9) + 10)
12 * (1 - (9 - 10))
12 * (1 + (10 - 9))
12 * ((1 + 10) - 9)
12 * (10 + (1 - 9))
12 * ((10 + 1) - 9)
12 * ((10 - 9) + 1)
12 * (10 - (9 - 1))
Execution time = 1 milliseconds
Apakah ingin menyimpan solusi?(Y/N)
y
Nama file (contoh: testcase3.out) : tc_rand1.out
Solusi sudah dimasukkan ke test/tc_rand1.out

```

Gambar 8. Masukan dan Keluaran Kasus Uji Random 1

8. Kasus uji random 2

File terkait: tc_rand2.out

```

* * * *
Kartu : J 8 7 5
no solution found
Execution time = 0 milliseconds
Apakah ingin menyimpan solusi?(Y/N)
y
Nama file (contoh: testcase3.out) : tc_rand2.out
Solusi sudah dimasukkan ke test/tc_rand2.out

```

Gambar 9. Masukan dan Keluaran Kasus Uji Random 2

SUMBER

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf>

<https://www.geeksforgeeks.org/measure-execution-time-function-cpp/>

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil1-Stima-2023.pdf>

<https://samidavies.wordpress.com/2017/04/14/the-hardest-games-of-24/>

LAMPIRAN

Checklist program

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	✓	

Link repository program: https://github.com/Michaelu670/Tucil1_13521137