

LAPORAN
TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA
PERMAINAN KARTU 24



Disusun oleh:
(13521087) Razzan Daksana Yoni

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

Daftar Isi

BAB 1 Deskripsi Masalah	1
BAB II Teori Singkat.....	2
A. Algoritma <i>Brute Force</i>	2
B. Exhaustive Search	2
BAB III Aplikasi Brute Force pada Permainan Kartu 24.....	3
BAB IV Implementasi Program	3
BAB V Eksperimen	11
BAB VI Kesimpulan.....	15

BAB 1 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 (\times), divisi ($/$) dan tanda kurung (). Tiap kartu harus digunakan tepat sekali dan urutan penggunaannya bebas.

BAB II Teori Singkat

A. Algoritma *Brute Force*

Algoritma *Brute Force* adalah pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Biasanya algoritma *brute force* didasarkan pada pernyataan pada persoalan (*problem statement*) dan konsep yang dilibatkan. Algoritma *brute force* secara langsung, dan sederhana, Algoritma *brute force* juga hampir bisa diimplementasikan untuk semua kasus.

Kasus yang dapat diselesaikan dengan Algoritma *brute force* seperti mencari elemen ekstrem (terbesar ataupun terkecil) dan mencari elemen unik. Akan tetapi, algoritma *brute force* umumnya tidak efisien, karena membutuhkan komputasi yang besar dan waktu yang lama. Oleh karena itu, algoritma *brute force* hanya cocok untuk kasus (n) yang kecil.

B. Exhaustive Search

Exhaustive Search adalah teknik pencarian dengan algoritma *brute force* untuk persoalan-persoalan kombinatorik seperti permutasi, kombinasi, dan himpunan bagian. Secara umum langkah-langkah *exhaustive search* yaitu,

1. Enumerasi (*list*) setiap kemungkinan solusi dengan cara yang sistematis.
2. Evaluasi setiap kemungkinan solusi satu per satu, simpan solusi terbaik yang ditemukan sampai sejauh ini.
3. Bila pencarian berakhir, umumkan solusi terbaik

Dapat dilihat dari langkah yang dilakukan, *exhaustive search* pasti menghasilkan solusi, tetapi waktu yang dibutuhkan dalam pencarian sangatlah besar karena solusi ini mengenumerasikan semua kemungkinan yang mungkin terjadi.

BAB III Aplikasi Brute Force pada Permainan Kartu 24

Langkah yang dilakukan pada permainan kartu 24 dengan implementasi *brute force* yaitu,

1. Permutasikan tiap kombinasi angka
2. Masukkan hasil langkah pertama ke suatu senarai
3. Enumerasikan kombinasi angka pada suatu perulangan
4. Enumerasikan operasi aritmatika pada suatu perulangan
5. Dalam langkah ketiga enumerasikan lagi langkah tersebut dalam perulangan langkah ketiga hingga terdapat tiga perulangan bersarang operasi aritmatika
6. Tinjau kemungkinan tanda kurung yang mungkin terjadi
7. Lakukan operasi aritmatika setelah didapatkannya kombinasi angka, operasi, dan tanda kurung
8. Cocokkan dengan angka 24 jika sama masukkan ke dalam senarai
9. Ulangi Langkah ke-tujuh hingga tidak ada lagi operasi yang dapat dilakukan

BAB IV Implementasi Program

Catatan : Implementasi dilakukan dalam bahasa pemrograman C++

1. Fungsi func

```
void func (vector<vector<string>>&result, vector<string>&
nums, vector<int>& vis, vector<string>& temp){
    /* Deskripsi : Mencari seluruh permutasi dari array of
string dengan merekursi tiap kemungkinan*/
    /* F.S : Mengembalikan kemungkinan permutasi dari array
of string */

    if(temp.size() == 4){
        result.push_back(temp);
        return;
    }

    for(int i = 0; i < 4; i++){
        if(vis[i] == 0){
            vis[i] = 1;
            temp.push_back(nums[i]);
            func(result, nums, vis, temp);
            temp.pop_back();
            vis[i] = 0;
        }
    }
}
```

2. Fungsi permute

```
vector<vector<string>> permute(vector<string>& nums) {  
    /* Deskripsi : Mencari seluruh permutasi dari array of string  
    */  
    /* F.S : Mengembalikan seluruh permutasi dari array of string  
    */  
  
    vector<vector<string>> result;  
    vector<int> vis(4,0);  
    vector<string> temp;  
    func(result, nums, vis, temp);  
    return result;  
}
```

3. Fungsi stof

```
float stof(const char *s)  
{  
    /* Deskripsi : Mengkonversi string ke float dengan menambahkan  
    tiap bilangan pada setiap bilangan yang ada di dalam string hingga  
    menjadi satu bilangan utuh*/  
    /* F.S : Mengembalikan hasil konversi string ke float */  
    float i;  
    i = 0;  
    short sign = 1;  
  
    if (*s == '-')  
    {  
        sign = -1;  
        s++;  
    }  
    while(*s >= '0' && *s <= '9')  
    {  
        i = i * 10 + (*s - '0');  
        s++;  
    }  
    if (*s == '.')  
    {  
        s++;  
        float j = 0.1;  
        while(*s >= '0' && *s <= '9')  
        {  
            i = i + (*s - '0') * j;  
            j = j / 10;  
            s++;  
        }  
    }  
    return i * sign;  
}
```

4. Fungsi result

```
bool result(vector<string> arithmetic)  
{
```

```

/* Deskripsi : perhitungan akan dilakukan ketika menemukan ')'
lalu ia akan mundur ke belakang sejauh 3 langkah dan mengeksekusi
operasi aritmatika*/
/* F.S : Mengembalikan hasil operasi aritmatika */
auto p = arithmetic.begin();
float num1, num2, result;
char op;

while (arithmetic.front() != arithmetic.back())
{
    if (*p == ")") {
        p = prev(p, 1); // p di num2
        arithmetic.erase(next(p,1)); // hapus )

        num2 = stof(p->c_str()); // assign value num2
        p = prev(p, 1); // p di op
        arithmetic.erase(next(p,1)); // hapus num2

        op = p->c_str()[0]; // assign value op
        p = prev(p, 1); // p di num 1
        arithmetic.erase(next(p,1)); // hapus sop

        num1 = stof(p->c_str()); // assign value num1
        p = prev(p, 1); // p di (
        arithmetic.erase(next(p,1)); // hapus num1

        switch (op)
        {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                result = num1 / num2;
                break;
        }
        arithmetic.insert(p, to_string(result));
        p = prev(p, 1); // p di sebelum (
        arithmetic.erase(next(p,2)); // hapus (
    }
    p = next(p, 1);
}

if (result >= 23.98 && result <= 24.02)
{
    return true;
}
else
{
    return false;
}
}

```

5. Fungsi solution

```

void solution(vector<vector<string>> permutation, long ctime)
{
    /* Deskripsi : mencari solusi untuk setiap kemungkinan operasi
    aritmatik dan kemungkinan tanda kurung*/
    /* F.S : Mengembalikan hasil array of int */
    vector<vector<string>> arithmetic(5);
    vector<vector<string>> solution(7680);
    bool found = false;
    int count = 0;

    for (vector<string> temp : permutation)
    {
        for (string i : {"+", "-", "*", "/"})
        {
            for (string j : {"+", "-", "*", "/"})
            {
                for (string k : {"+", "-", "*", "/"})
                {
                    /* ((a op b) op (c op d))*/
                    arithmetic[0].emplace_back("(");
                    arithmetic[0].emplace_back("(");
                    arithmetic[0].emplace_back(temp[0]);
                    arithmetic[0].emplace_back(i);
                    arithmetic[0].emplace_back(temp[1]);
                    arithmetic[0].emplace_back(")");
                    arithmetic[0].emplace_back(j);
                    arithmetic[0].emplace_back("(");
                    arithmetic[0].emplace_back(temp[2]);
                    arithmetic[0].emplace_back(k);
                    arithmetic[0].emplace_back(temp[3]);
                    arithmetic[0].emplace_back(")");
                    arithmetic[0].emplace_back(")");

                    /* (((a op b) op c) op d)*/
                    arithmetic[1].emplace_back("(");
                    arithmetic[1].emplace_back("(");
                    arithmetic[1].emplace_back("(");
                    arithmetic[1].emplace_back(temp[0]);
                    arithmetic[1].emplace_back(i);
                    arithmetic[1].emplace_back(temp[1]);
                    arithmetic[1].emplace_back(")");
                    arithmetic[1].emplace_back(j);
                    arithmetic[1].emplace_back(temp[2]);
                    arithmetic[1].emplace_back(")");
                    arithmetic[1].emplace_back(k);
                    arithmetic[1].emplace_back(temp[3]);
                    arithmetic[1].emplace_back(")");

                    /* ((a op (b op c)) op d)*/
                    arithmetic[2].emplace_back("(");
                    arithmetic[2].emplace_back("(");
                    arithmetic[2].emplace_back(temp[0]);

```



```

arithmetic[2].emplace_back(i);
arithmetic[2].emplace_back("(");
arithmetic[2].emplace_back(temp[1]);
arithmetic[2].emplace_back(j);
arithmetic[2].emplace_back(temp[2]);
arithmetic[2].emplace_back(")");
arithmetic[2].emplace_back(")");
arithmetic[2].emplace_back(k);
arithmetic[2].emplace_back(temp[3]);
arithmetic[2].emplace_back(")");

/* (a op (b op (c op d)))*
arithmetic[3].emplace_back("(");
arithmetic[3].emplace_back(temp[0]);
arithmetic[3].emplace_back(i);
arithmetic[3].emplace_back("(");
arithmetic[3].emplace_back(temp[1]);
arithmetic[3].emplace_back(j);
arithmetic[3].emplace_back("(");
arithmetic[3].emplace_back(temp[2]);
arithmetic[3].emplace_back(k);
arithmetic[3].emplace_back(temp[3]);
arithmetic[3].emplace_back(")");
arithmetic[3].emplace_back(")");
arithmetic[3].emplace_back(")");

/* (a op ((b op c) op d))*
arithmetic[4].emplace_back("(");
arithmetic[4].emplace_back(temp[0]);
arithmetic[4].emplace_back(i);
arithmetic[4].emplace_back("(");
arithmetic[4].emplace_back("(");
arithmetic[4].emplace_back(temp[1]);
arithmetic[4].emplace_back(j);
arithmetic[4].emplace_back(temp[2]);
arithmetic[4].emplace_back(")");
arithmetic[4].emplace_back(k);
arithmetic[4].emplace_back(temp[3]);
arithmetic[4].emplace_back(")");
arithmetic[4].emplace_back(")");

/* Cek apakah hasilnya 24 */
for (int i = 0; i < 5; i++)
{
    if (result(arithmetic[i]))
    {
        found = true;

        for (string x : arithmetic[i])
        {
            solution[count].emplace_back(x);
        }
        count++;
    }
    arithmetic[i].clear();
}
}

```

```

    }
}

etime = time(NULL);
cout << "Waktu Eksekusi : " << etime - ctime << " detik" <<
endl;

if (!found)
{
    cout << "Tidak ada solusi" << endl;
} else
{
    char c;
    string filename;

    while (true)
    {
        cout << "Apakah anda ingin menyimpan solusi? (y/n) :
";
        cin >> c;

        if (c == 'y')
        {
            cout << "Masukkan nama file : ";
            cin >> filename;
            ofstream fw("./test/" + filename + ".txt", ios::out
| ios::trunc);

            if (!fw.is_open())
            {
                cout << "File tidak dapat dibuka" << endl;
                break;
            }

            fw << permutation[0][0] << " " <<
permutation[0][1] << " " << permutation[0][2] << " " <<
permutation[0][3] << endl;

            fw << "Jumlah solusi : " << count << endl;

            for (int i = 0; i < count; i++)
            {
                for (auto x : solution[i])
                {
                    fw << x;
                }
                fw << endl;
            }
            fw.close();
            break;
        } else if (c == 'n')
        {
            break;
        } else
        {
            cout << "Input tidak valid" << endl;

```

```
    }  
    }  
}
```

6. Fungsi main

```
int main(int argc, char const *argv[])  
{  
    bool PlayGame = true;  
    bool found;  
    short mode;  
    vector<string> temp(4);  
    vector<vector<string>> permutation(24);  
    short num[4];  
    long ctime;  
  
    cout << "==== Permainan Kartu 24 =====< endl;  
    while (PlayGame) {  
        cout << "Pilih Mode Permainan" << endl;  
        cout << "1. Main Sendiri" << endl;  
        cout << "2. Kartu Acak" << endl;  
        cout << "3. Bantuan" << endl;  
        cout << "4. Keluar" << endl;  
  
        cout << "Pilihan : ";  
        cin >> mode;  
  
        found = true;  
        switch (mode)  
        {  
            case 1:  
                cout << "Main Sendiri" << endl;  
                cout << "Masukkan 4 Kartu! " << endl;  
                for (int i = 0; i < 4; i++)  
                {  
                    cout << "Masukkan Kartu Ke-" << i + 1 << " : ";  
                    cin >> temp[i];  
                    if (temp[i] == "A")  
                    {  
                        temp[i] = "1";  
                    }  
                    else if (temp[i] == "J")  
                    {  
                        temp[i] = "11";  
                    }  
                    else if (temp[i] == "Q")  
                    {  
                        temp[i] = "12";  
                    }  
                    else if (temp[i] == "K")  
                    {  
                        temp[i] = "13";  
                    }  
                }  
            }  
        }  
    }  
}
```

```

        else if (!(temp[i] == "2" || temp[i] == "3" ||
temp[i] == "4" || temp[i] == "5" || temp[i] == "6" || temp[i] ==
"7" || temp[i] == "8" || temp[i] == "9" || temp[i] == "10" ))
        {
            cout << "Kamu salah memasukkan kartu" << endl;
            found = false;
            break;
        }
    }

    /* Solution Here */
    if (found)
    {
        ctime = time(NULL);
        permutation = permute(temp);
        solution(permutation, ctime);
    }

    for (int i = 0; i < 4; i++)
    {
        temp[i] = "";
    }
    break;
case 2:
    cout << "Kartu Acak" << endl;
    srand(time(NULL));

    for (int i = 0; i < 4; i++)
    {
        temp[i] = to_string(rand() % 13 + 1);
    }

    for (int i = 0; i < 4; i++)
    {
        if (temp[i] == "1")
        {
            cout << "A ";
        }
        else if (temp[i] == "11")
        {
            cout << "J ";
        }
        else if (temp[i] == "12")
        {
            cout << "Q ";
        }
        else if (temp[i] == "13")
        {
            cout << "K ";
        }
        else
        {
            cout << temp[i] << " ";
        }
    }
}

```

```
        cout << endl;

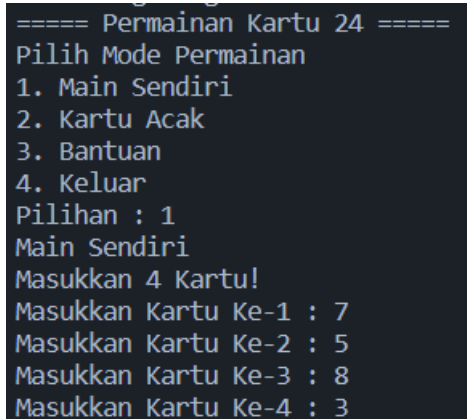
        ctime = time(NULL);
        permutation = permute(temp);
        solution(permutation, ctime);

        break;
    case 3:
        cout << "Bantuan" << endl;
        cout << "\nNilai-Nilai dari Kartu yang Bukan Angka :  

\nA : 1\nJ : 11\nQ : 12\nK : 13\n" << endl;
        break;
    case 4:
        cout << "Keluar" << endl;
        PlayGame = false;
        break;
    default:
        cout << "Pilihan Tidak Tersedia" << endl;
        break;
    }
}
return 0;
}
```

BAB V Eksperimen

1. 7 5 8 3 (Masukan)



```
===== Permainan Kartu 24 =====
Pilih Mode Permainan
1. Main Sendiri
2. Kartu Acak
3. Bantuan
4. Keluar
Pilihan : 1
Main Sendiri
Masukkan 4 Kartu!
Masukkan Kartu Ke-1 : 7
Masukkan Kartu Ke-2 : 5
Masukkan Kartu Ke-3 : 8
Masukkan Kartu Ke-4 : 3
```

Gambar 5.1 Input-an angka kasus pertama

```
Waktu Eksekusi : 0 detik
Jumlah solusi : 30
Solusi : ((7*5)-(8+3))
Solusi : (((7*5)-8)-3)
Solusi : ((7*5)-(3+8))
Solusi : (((7*5)-3)-8)
Solusi : ((7*(8-5))+3)
Solusi : (((7*3)-5)+8)
Solusi : ((7*3)-(5-8))
Solusi : ((7*3)+(8-5))
Solusi : (((7*3)+8)-5)
Solusi : ((5*7)-(8+3))
Solusi : (((5*7)-8)-3)
Solusi : ((5*7)-(3+8))
Solusi : (((5*7)-3)-8)
Solusi : ((8+(7*3))-5)
Solusi : (8+((7*3)-5))
Solusi : ((8-5)+(7*3))
Solusi : (8-(5-(7*3)))
Solusi : (((8-5)*7)+3)
Solusi : ((8-5)+(3*7))
Solusi : (8-(5-(3*7)))
Solusi : ((8+(3*7))-5)
Solusi : (8+((3*7)-5))
Solusi : (3-(7*(5-8)))
Solusi : (((3*7)-5)+8)
Solusi : ((3*7)-(5-8))
Solusi : (3+(7*(8-5)))
Solusi : ((3*7)+(8-5))
Solusi : (((3*7)+8)-5)
Solusi : (3-((5-8)*7))
Solusi : (3+((8-5)*7))
```

Gambar 5.2 Output kasus pertama

2. A 8 9 Q (Masukan)

```
Pilih Mode Permainan
1. Main Sendiri
2. Kartu Acak
3. Bantuan
4. Keluar
Pilihan : 1
Main Sendiri
Masukkan 4 Kartu!
Masukkan Kartu Ke-1 : A
Masukkan Kartu Ke-2 : 8
Masukkan Kartu Ke-3 : 9
Masukkan Kartu Ke-4 : Q
```

Gambar 5.3 Input kasus kedua

```

Waktu Eksekusi : 0 detik
Jumlah solusi : 49
Solusi : (((1-8)+9)*12)
Solusi : ((1-(8-9))*12)
Solusi : ((1*8)*(12-9))
Solusi : (1*(8*(12-9)))
Solusi : (((1+9)-8)*12)
Solusi : ((1+(9-8))*12)
Solusi : (((1*12)-9)*8)
Solusi : ((1*(12-9))*8)
Solusi : (1*((12-9)*8))
Solusi : ((8*1)*(12-9))
Solusi : (8*(1*(12-9)))
Solusi : (8*((1*12)-9))
Solusi : ((8/1)*(12-9))
Solusi : (8/(1*(12-9)))
Solusi : (8*(12-(1*9)))
Solusi : (8*((12*1)-9))
Solusi : (8*((12/1)-9))
Solusi : ((8*(12-9))*1)
Solusi : (8*(12-(9*1)))
Solusi : (8*((12-9)*1))
Solusi : ((8*(12-9))/1)
Solusi : (8*(12-(9/1)))
Solusi : (8*((12-9)/1))
Solusi : (8/((12/9)-1))
Solusi : (((9+1)-8)*12)
Solusi : ((9+(1-8))*12)
Solusi : (((9-8)+1)*12)
Solusi : ((9-(8-1))*12)
Solusi : (12*((1-8)+9))
Solusi : (12*(1-(8-9)))
Solusi : ((12-(1*9))*8)
Solusi : (12*(1+(9-8)))
Solusi : (12*((1+9)-8))
Solusi : (((12*1)-9)*8)
Solusi : (((12/1)-9)*8)
Solusi : ((12-9)*(1*8))
Solusi : (((12-9)*1)*8)
Solusi : ((12-(9*1))*8)
Solusi : (((12-9)/1)*8)
Solusi : ((12-(9/1))*8)
Solusi : ((12-9)/(1/8))
Solusi : (12*(9+(1-8)))
Solusi : (12*((9+1)-8))
Solusi : ((12-9)*(8*1))
Solusi : (((12-9)*8)*1)
Solusi : ((12-9)*(8/1))
Solusi : (((12-9)*8)/1)
Solusi : (12*((9-8)+1))
Solusi : (12*(9-(8-1)))

```

Gambar 5.4 Output kasus kedua

3. 7 7 7 7 (Masukan)

```

Pilih Mode Permainan
1. Main Sendiri
2. Kartu Acak
3. Bantuan
4. Keluar
Pilihan : 1
Main Sendiri
Masukkan 4 Kartu!
Masukkan Kartu Ke-1 : 7
Masukkan Kartu Ke-2 : 7
Masukkan Kartu Ke-3 : 7
Masukkan Kartu Ke-4 : 7
Tidak ada solusi
Waktu Eksekusi : 0 detik

```

Gambar 5.5 Input dan output kasus ketiga

4. 5 7 7 A (Kartu Acak)

```

Pilih Mode Permainan
1. Main Sendiri
2. Kartu Acak
3. Bantuan
4. Keluar
Pilihan : 2
Kartu Acak
5 7 7 A
Tidak ada solusi
Waktu Eksekusi : 1 detik
```

Gambar 5.6 Input dan output kasus acak pertama

5. 7 A 2 8 (Kartu Acak)

```

Pilih Mode Permainan
1. Main Sendiri
2. Kartu Acak
3. Bantuan
4. Keluar
Pilihan : 2
Kartu Acak
7 A 2 8
```

Gambar 5.7 Input kasus acak kedua


```
Waktu Eksekusi : 0 detik
Jumlah solusi : 40
Solusi : ((7+1)+(2*8))
Solusi : (7+(1+(2*8)))
Solusi : (((7+1)*2)+8)
Solusi : (((7-1)/2)*8)
Solusi : ((7-1)/(2/8))
Solusi : ((7+1)+(8*2))
Solusi : (7+(1+(8*2)))
Solusi : ((7-1)*(8/2))
Solusi : (((7-1)*8)/2)
Solusi : ((7+(2*8))+1)
Solusi : (7+((2*8)+1))
Solusi : ((7+(8*2))+1)
Solusi : (7+((8*2)+1))
Solusi : ((1+7)+(2*8))
Solusi : (1+(7+(2*8)))
Solusi : (((1+7)*2)+8)
Solusi : ((1+7)+(8*2))
Solusi : (1+(7+(8*2)))
Solusi : ((1+(2*8))+7)
Solusi : (1+((2*8)+7))
Solusi : ((1+(8*2))+7)
Solusi : (1+((8*2)+7))
Solusi : ((2*(7+1))+8)
Solusi : ((2*(1+7))+8)
Solusi : ((2*8)+(7+1))
Solusi : (((2*8)+7)+1)
Solusi : ((2*8)+(1+7))
Solusi : (((2*8)+1)+7)
Solusi : (8+((7+1)*2))
Solusi : ((8*(7-1))/2)
Solusi : (8*((7-1)/2))
Solusi : (8+((1+7)*2))
Solusi : (8+(2*(7+1)))
Solusi : ((8*2)+(7+1))
Solusi : (((8*2)+7)+1)
Solusi : ((8/2)*(7-1))
Solusi : (8/(2/(7-1)))
Solusi : (8+(2*(1+7)))
Solusi : ((8*2)+(1+7))
Solusi : (((8*2)+1)+7)
```

Gambar 5.8 Output kasus acak kedua

6. A 6 Q A (Kartu Acak)

```
Pilih Mode Permainan
1. Main Sendiri
2. Kartu Acak
3. Bantuan
4. Keluar
Pilihan : 2
Kartu Acak
A 6 Q A
```

Gambar 5.9 Input kasus acak ketiga

```
Jumlah solusi : 8
Solusi : (((1+1)*6)+12)
Solusi : ((6*(1+1))+12)
Solusi : (((6*(1+1))+12)
Solusi : (12+((1+1)*6))
Solusi : (12+(6*(1+1)))
Solusi : (12+(6*(1+1)))
Solusi : (12+((1+1)*6))
Solusi : (((1+1)*6)+12)
```

Gambar 5.10 Output kasus acak ketiga

BAB VI Kesimpulan

Permainan Kartu 24 dapat diselesaikan dengan algoritma *brute force*. Mengenumerasikan kemungkinan yang terjadi adalah caranya. Dengan mengenumerasikan tiap kemungkinan dapat diketahui semua kasus yang mungkin terjadi tanpa terkecuali. Dengan dibantu alat dengan mudah untuk mengenumerasikan dan mengeksekusikan hal tersebut. Dengan kata lain, pemecahan masalah permainan kartu 24 menggunakan pendekatan algoritma *brute force* ini menjadikan problem tersebut computable.

Saran pengembangan dari kami adalah penjelasan spesifikasi tugas bisa lebih mendetail karena cukup banyak bagian yang kurang jelas, jika tertulis ada batasan cantumkan batasan tersebut, karena tanpa itu mahasiswa ragu untuk melakukan pendekatan yang dipikirkan.

Referensi

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

[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)

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf)

<https://www.geeksforgeeks.org/vector-in-cpp-stl/>

Lampiran

https://github.com/razzanYoni/Tucil1_13521087

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	✓	
Program berhasil <i>running</i>	✓	
Program dapat membaca <i>input</i> / <i>generate</i> sendiri dan memberikan luaran	✓	
Solusi yang diberikan program memenuhi (berhasil mencapai 24)	✓	
Program dapat menyimpan solusi dalam file teks	✓	