Aplikasi Algoritma *Brute Force* dalam Penyelesaian Permainan Kartu 24

Manuella Ivana Uli Sianipar - 13521051[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521051@std.stei.itb.ac.id

***Abstrak*—Permainan kartu 24 adalah permainan aritmatika dimana pemain diminta untuk membuat angka 24 dari 4 angka yang diambil secara acak. Untuk membuat angka 24 dibutuhkan kombinasi operator dan susunan angka yang tepat agar membentuk angka 24. Kombinasi ini dapat dicari dengan menggunakan algoritma *brute force*.**

***Kata kunci*—algoritma *brute force*, permainan kartu 24, strategi algoritma.**

I. PENDAHULUAN

Permainan kartu merupakan permainan yang memiliki banyak peminat. Jenis-jenis permainan kartu ada banyak, beberapa diantaranya adalah Poker, Spider, Hearts, 24 dan lain-lain. Pada makalah ini penulis akan membahas tentang permainan kartu 24.

Permainan kartu 24 adalah permainan yang mengasah kemampuan aritmatika. Dalam permainan ini, pemain diminta untuk membuat angka 24 dari 4 angka acak dengan menggunakan operasi aritmatika. Pemain berusaha untuk memikirkan kombinasi yang tepat untuk menghasilkan angka 24. Permainan ini mengasah kemampuan berhitung dan melatih otak untuk berpikir secara cepat.

Untuk mendapatkan solusi yang tepat, dibutuhkan pengecekan kombinasi-kombinasi yang mungkin. Hal ini dapat diselesaikan dengan menggunakan algoritma pemrograman. Salah satu algoritma yang dapat digunakan adalah algoritma *brute force*. Algoritma *brute force* adalah algoritma yang sederhana dan mudah untuk diimplementasikan. Pada makalah ini, akan dibahas tentang aplikasi algoritma *brute force* dalam pembuatan penyelesaian permainan kartu 24.

II. DASAR TEORI

A. Permainan Kartu 24

Permainan kartu 24 dimainkan dengan menggunakan kartu remi. Kartu remi terdiri atas 52 kartu yang terdiri atas empat jenis (sekop, hati, keriting dan wajik), masing-masing memiliki 13 kartu dengan nilai berbeda (As, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K). As bernilai 1, J bernilai 11, Q bernilai 12 dan K bernilai 13.

Gambar 1 Kartu remi (Sumber: [3])

Kartu-kartu tersebut diacak, lalu diambil 4 kartu untuk dimainkan. Angka 24 dapat dibentuk hanya dengan menggunakan operator tambah, kurang, bagi, kali dan tanda kurung. Untuk mendapatkan penyelesaian permainan ini, pemain perlu memikirkan kombinasi angka dan operator yang mungkin.

Langkah-langkah permainannya adalah sebagai berikut:

1. Moderator mengambil empat buah kartu diambil dari dek kartu yang sudah diacak.
2. Para pemain memikirkan cara untuk membuat keempat angka tersebut menghasilkan angka 24 menggunakan operator tambah, kurang, kali, bagi dan tanda kurung.
3. Pemain yang sudah mendapatkan solusi dapat menunjukkan tangan atau memukul meja untuk menyatakan bahwa ia sudah mendapatkan solusi.
4. Jika pemain dapat menjelaskan bagaimana cara ia mendapatkan angka 24, maka pemain tersebut mendapatkan keempat kartu tadi.
5. Jika semua pemain menyerah karena tidak mendapatkan solusi, keempat kartu yang diambil dapat dikembalikan kembali ke dek kartu dan diacak kembali.
6. Pemain dengan kartu terbanyak adalah pemenangnya.

B. Algoritma Brute Force

Algoritma *brute force* adalah pendekatan yang lempang untuk memecahkan suatu persoalan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung dan dengan cara yang jelas.

Karakteristik algoritma *brute force* adalah:

7. Algoritma *brute force* uumunya tidak "cerdas" dan tidak mangkus. Algoritma ini membutuhkan waktu yang panjang

dalam penyelesaiannya. Karena itu, algoritma *brute force* sering juga disebut algoritma naif (*naive algorithm*).

8. Algoritma *brute force* lebih cocok untuk persoalan dengan jumlah input yang kecil karena algoritmanya sederhana dan mudah untuk diimplementasikan. Biasanya algoritma *brute force* dipakai sebagai pembanding dengan algoritma lain yang lebih mangkus.
9. Hampir semua persoalan dapat diselesaikan dengan algoritma *brute force*. Beberapa persoalan bahkan hanya dapat diselesaikan dengan algoritma *brute force*.

Algoritma *brute force* memiliki beberapa kelebihan yaitu:

1. Algoritma *brute force* dapat diterapkan untuk memecahkan hampir sebagian besar masalah (*wide capability*).
2. Algoritma *brute force* sederhana dan mudah dimengerti.
3. Algoritma *brute force* menghasilkan algoritma yang layak untuk beberapa masalah penting.
4. Algoritma *brute force* menghasilkan algoritma baku.

Namun, algoritma ini juga memiliki beberapa kelemahan yaitu:

1. Algoritma *brute force* jarang menghasilkan algoritma yang mangkus.
2. Algoritma *brute force* umumnya lambat untuk jumlah input yang besar.
3. Algoritma *brute force* tidak kreatif.

III. Penerapan Algoritma

Pencarian kombinasi yang tepat dapat dilakukan dengan algoroitma *brute force*. Pada program penulis, kombinasi angka dibuat menggunakan permutasi, kombinasi operator dibuat menggunakan *nested loop for* dan kombinasi tanda kurung dibuat manual dalam bentuk fungsi.

Pada program penulis terdapat juga ADT yang memudahkan proses pembuatan program yaitu ADT equation dan ADT list dinamik. ADT equation berisi definisi bentukan Eq untuk menyimpan persamaan dan fungsi-fungsi yang memudahkan perhitungan. ADT list dinamik berisi definisi bentukan ListDin yaitu list dinamik untuk menyimpan solusi yang ada.

Langkah-langkah algoritma *brute force* yang dipakai adalah:

1. Empat angka akan disusun dalam sebuah persamaan, masukkan susunan pertama ke dalam kombinasi tanda kurung pertama.
2. Untuk setiap posisi operator dilakukan *loop for* untuk mencoba semua operator yang mungkin.
3. Persamaan dicari hasilnya dengan memanfaatkan fungsi dari ADT equation.
4. Jika hasilnya adalah 24, maka persamaan tersebut dimasukkan ke list dinamis.
5. Langkah 2 sampai 4 diulangi untuk setiap kombinasi tanda kurung.
6. Langkah 1 sampai 4 diulangi untuk setiap permutasi angka.
7. Setelah semua kombinasi dicoba, program akan mencetak semua solusi yang tersimpan.

Kombinasi tanda kurung yang mungkin ada 11 yaitu:

```
void combin1(int arr[4], int op[4], ListDin *leq){
    // _ op _ op _ op _
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[1] = op[i];
        eq.TabVal[2] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[3] = op[j];
            eq.TabVal[4] = arr[2];
            for(int k = 0; k < 4; k++){
                eq.TabVal[5] = op[k];
                eq.TabVal[6] = arr[3];
                eq.LenVal = 7;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 2 Kombinasi tanda kurung _ op _ op _ op _

```
void combin2(int arr[4], int op[4], ListDin *leq){
    // ( _ op _ ) op ( _ op _ )
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = left;
    eq.TabVal[1] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[2] = op[i];
        eq.TabVal[3] = arr[1];
        eq.TabVal[4] = right;
        for(int j = 0; j < 4; j++){
            eq.TabVal[5] = op[j];
            eq.TabVal[6] = left;
            eq.TabVal[7] = arr[2];
            for(int k = 0; k < 4; k++){
                eq.TabVal[8] = op[k];
                eq.TabVal[9] = arr[3];
                eq.TabVal[10] = right;
                eq.LenVal = 11;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 3 Kombinasi tanda kurung (_ op _) op (_ op _)

```
void combin3(int arr[4], int op[4], ListDin *leq){
    // ( _ op _ ) op _ op _
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = left;
    eq.TabVal[1] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[2] = op[i];
        eq.TabVal[3] = arr[1];
        eq.TabVal[4] = right;
        for(int j = 0; j < 4; j++){
            eq.TabVal[5] = op[j];
            eq.TabVal[6] = arr[2];
            for(int k = 0; k < 4; k++){
                eq.TabVal[7] = op[k];
                eq.TabVal[8] = arr[3];
                eq.LenVal = 9;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 4 Kombinasi tanda kurung (_ op _) op _ op _

```c
void combin4(int arr[4], int op[4], ListDin *leq){
    // _ op ( _ op _ ) op _
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[1] = op[i];
        eq.TabVal[2] = left;
        eq.TabVal[3] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[4] = op[j];
            eq.TabVal[5] = arr[2];
            eq.TabVal[6] = right;
            for(int k = 0; k < 4; k++){
                eq.TabVal[7] = op[k];
                eq.TabVal[8] = arr[3];
                eq.LenVal = 9;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 5 Kombinasi tanda kurung _ op (_ op _) op _

```c
void combin5(int arr[4], int op[4], ListDin *leq){
    // _ op _ op ( _ op _ )
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[1] = op[i];
        eq.TabVal[2] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[3] = op[j];
            eq.TabVal[4] = left;
            eq.TabVal[5] = arr[2];
            for(int k = 0; k < 4; k++){
                eq.TabVal[6] = op[k];
                eq.TabVal[7] = arr[3];
                eq.TabVal[8] = right;
                eq.LenVal = 9;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 6 Kombinasi tanda kurung _ op _ op (_ op _)

```c
void combin6(int arr[4], int op[4], ListDin *leq){
    // ( _ op _ op _ ) op _
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = left;
    eq.TabVal[1] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[2] = op[i];
        eq.TabVal[3] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[4] = op[j];
            eq.TabVal[5] = arr[2];
            eq.TabVal[6] = right;
            for(int k = 0; k < 4; k++){
                eq.TabVal[7] = op[k];
                eq.TabVal[8] = arr[3];
                eq.LenVal = 9;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 7 Kombinasi tanda kurung (_ op _ op _) op _

```c
void combin7(int arr[4], int op[4], ListDin *leq){
    // _ op ( _ op _ op _ )
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[1] = op[i];
        eq.TabVal[2] = left;
        eq.TabVal[3] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[4] = op[j];
            eq.TabVal[5] = arr[2];
            for(int k = 0; k < 4; k++){
                eq.TabVal[6] = op[k];
                eq.TabVal[7] = arr[3];
                eq.TabVal[8] = right;
                eq.LenVal = 9;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 8 Kombinasi tanda kurung _ op (_ op _ op _)

```c
void combin8(int arr[4], int op[4], ListDin *leq){
    // (( _ op _ ) op _ ) op _
    int sol = 0;
    Eq eq, eqs;
    eq.TabVal[0] = left;
    eq.TabVal[1] = left;
    eq.TabVal[2] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[3] = op[i];
        eq.TabVal[4] = arr[1];
        eq.TabVal[5] = right;
        for(int j = 0; j < 4; j++){
            eq.TabVal[6] = op[j];
            eq.TabVal[7] = arr[2];
            eq.TabVal[8] = right;
            for(int k = 0; k < 4; k++){
                eq.TabVal[9] = op[k];
                eq.TabVal[10] = arr[3];
                eq.LenVal = 11;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 9 Kombinasi tanda kurung ((_ op _) op _) op _

```c
void combin9(int arr[4], int op[4], ListDin *leq){
    // ( _ op ( _ op _ )) op _
    int sol;
    Eq eq, eqs;
    eq.TabVal[0] = left;
    eq.TabVal[1] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[2] = op[i];
        eq.TabVal[3] = left;
        eq.TabVal[4] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[5] = op[j];
            eq.TabVal[6] = arr[2];
            eq.TabVal[7] = right;
            eq.TabVal[8] = right;
            for(int k = 0; k < 4; k++){
                eq.TabVal[9] = op[k];
                eq.TabVal[10] = arr[3];
                eq.LenVal = 11;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 10 Kombinasi tanda kurung (_ op (_ op _)) op _

```c
void combin10(int arr[4], int op[4], ListDin *leq){
    // _ op (( _ op _ ) op _ )
    int sol;
    Eq eq, eqs;
    eq.TabVal[0] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[1] = op[i];
        eq.TabVal[2] = left;
        eq.TabVal[3] = left;
        eq.TabVal[4] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[5] = op[j];
            eq.TabVal[6] = arr[2];
            eq.TabVal[7] = right;
            for(int k = 0; k < 4; k++){
                eq.TabVal[8] = op[k];
                eq.TabVal[9] = arr[3];
                eq.TabVal[10] = right;
                eq.LenVal = 11;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 11 Kombinasi tanda kurung _ op ((_ op _) op _)

```c
void combin11(int arr[4], int op[4], ListDin *leq){
    // _ op ( _ op ( _ op _ ))
    int sol;
    Eq eq, eqs;
    eq.TabVal[0] = arr[0];
    for(int i = 0; i < 4; i++){
        eq.TabVal[1] = op[i];
        eq.TabVal[2] = left;
        eq.TabVal[3] = arr[1];
        for(int j = 0; j < 4; j++){
            eq.TabVal[4] = op[j];
            eq.TabVal[5] = left;
            eq.TabVal[6] = arr[2];
            for(int k = 0; k < 4; k++){
                eq.TabVal[7] = op[k];
                eq.TabVal[8] = arr[3];
                eq.TabVal[9] = right;
                eq.TabVal[10] = right;
                eq.LenVal = 11;
                eqs = eq;
                if (solveEq(&eqs) == 24){
                    insertLast(leq, eq);
                }
            }
        }
    }
}
```

Gambar 12 Kombinasi tanda kurung _ op (_ op (_ op _))

Fungsi untuk melakukan permutasi angka adalah fungsi rekursif. Jika angka sudah dipermutasikan, maka dilakukan pencarian kombinasi menggunakan fungsi-fungsi kombinasi tanda kurung.

Fungsi permutasinya adalah sebagai berikut:

```c
void combinarr(int arr[4], int start, int end, ListDin *leq){
    int op[4];
    op[0] = tambah;
    op[1] = kurang;
    op[2] = kali;
    op[3] = bagi;

    int sol = 0;
    if(start == end){
        for(int i = 0; i < 4; i++){
            combin1(arr, op, leq);
            combin2(arr, op, leq);
            combin3(arr, op, leq);
            combin4(arr, op, leq);
            combin5(arr, op, leq);
            combin6(arr, op, leq);
            combin7(arr, op, leq);
            combin8(arr, op, leq);
            combin9(arr, op, leq);
            combin10(arr, op, leq);
            combin11(arr, op, leq);
        }
        return;
    }
    else{
        combinarr(arr, start + 1, end, leq);
        for(int i = start + 1; i < end; i++){
            if(arr[start] != arr[i]){
                int temp = arr[start];
                arr[start] = arr[i];
                arr[i] = temp;
                combinarr(arr, start + 1, end, leq);
                temp = arr[start];
                arr[start] = arr[i];
                arr[i] = temp;
            }
        }
    }
}
```

Gambar 13 Permutasi angka pada array dengan 4 elemen

Program utama terdiri atas pilihan input angka (input pengguna atau secara random), pencarian solusi dan penyimpanan solusi dalam file jika pengguna ingin menyimpannya.

```c
int main(){
    ListDin leq;
    CreateListDin(&leq, 200);
    int input;
    printf("---- 24 GAME SOLVER ----\nPilih masukan input:\n1. Input user\n2. Random\nMasukkan pilihan: ");
    scanf("%d", &input);
    while (input != 1 && input != 2){
        printf("Masukan tidak sesuai. Masukkan pilihan: ");
        scanf("%d", &input);
    }

    int arr[4];
    if (input == 1){
        printf("Masukkan angka: ");
        char inp[12];
        int i = 0;
        char x;

        scanf("%c", &x);
        while(i < 12){
            inp[i] = x;
            i++;
            scanf("%c", &x);
            if (x == '\n'){
                break;
            }
        }
```

Gambar 14 Program utama (1)

```
    int k = 0;
    for(int j = 0; j < i; j++){
        if(inp[j] == 'K'){
            arr[k] = 13;
            k++;
        }
        else if(inp[j] == 'Q'){
            arr[k] = 12;
            k++;
        }
        else if(inp[j] == 'J'){
            arr[k] = 11;
            k++;
        }
        else if(inp[j] == '1'){
            j++;
            if(inp[j] == '0'){
                arr[k] = 10;
                k++;
            }
        }
        else if(inp[j] == '9'){
            arr[k] = 9;
            k++;
        }
        else if(inp[j] == '8'){
            arr[k] = 8;
            k++;
        }
        else if(inp[j] == '7'){
            arr[k] = 7;
            k++;
        }
```

Gambar 15 Program utama (2)

```
        else if(inp[j] == '6'){
            arr[k] = 6;
            k++;
        }
        else if(inp[j] == '5'){
            arr[k] = 5;
            k++;
        }
        else if(inp[j] == '4'){
            arr[k] = 4;
            k++;
        }
        else if(inp[j] == '3'){
            arr[k] = 3;
            k++;
        }
        else if(inp[j] == '2'){
            arr[k] = 2;
            k++;
        }
        else if(inp[j] == 'A'){
            arr[k] = 1;
            k++;
        }
    }
}
```

Gambar 16 Program utama (3)

```
else{
    srand(time(NULL));
    for(int i = 0; i < 4; i++){
        arr[i] = rand()%14;
        while(arr[i] == 0){
            arr[i] = rand()%14;
        }
    }

    printf("Angka: %d %d %d %d\n", arr[0], arr[1], arr[2], arr[3]);
}

clock_t begin = clock();
combinarr(arr, 0, 4, &leq);
clock_t end = clock();
printf("Jumlah solusi: %d\n", NEFF(leq));
printList(leq);
printf("Waktu eksekusi program: %f s\n", ((double)(end - begin) / CLOCKS_PER_SEC));

printf("Apakah ingin menyimpan solusi?\n1. Ya\n2. Tidak\nMasukkan pilihan: ");
scanf("%d", &input);
while (input != 1 && input != 2){
    printf("Masukan tidak sesuai. Masukkan pilihan: ");
    scanf("%d", &input);
}
```

Gambar 17 Program utama (4)

```
if (input == 1){
    printf("Masukkan nama file: ");
    char name[100];
    scanf("%s", &name);
    FILE *f;
    char dest[] = "../test/";
    strcat(dest,name);
    f = fopen(dest, "w");

    for(int i = 0; i < NEFF(leq); i++){
        for(int j = 0; j < ELMT(leq,i).LenVal; j++){
            if(ELMT(leq,i).TabVal[j] == left){
                fputc('(',f);
            }
            else if(ELMT(leq,i).TabVal[j] == right){
                fputc(')',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 1){
                fputc('1',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 2){
                fputc('2',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 3){
                fputc('3',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 4){
                fputc('4',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 5){
                fputc('5',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 6){
                fputc('6',f);
            }
```

Gambar 18 Program utama (5)

```
            else if(ELMT(leq,i).TabVal[j] == 7){
                fputc('7',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 8){
                fputc('8',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 9){
                fputc('9',f);
            }
            else if(ELMT(leq,i).TabVal[j] == 10){
                fputs("10",f);
            }
            else if(ELMT(leq,i).TabVal[j] == 11){
                fputs("11",f);
            }
            else if(ELMT(leq,i).TabVal[j] == 12){
                fputs("12",f);
            }
            else if(ELMT(leq,i).TabVal[j] == 13){
                fputs("13",f);
            }
            else if(ELMT(leq,i).TabVal[j] == tambah){
                fputc('+',f);
            }
            else if(ELMT(leq,i).TabVal[j] == kurang){
                fputc('-',f);
            }
            else if(ELMT(leq,i).TabVal[j] == kali){
                fputc('*',f);
            }
            else if(ELMT(leq,i).TabVal[j] == bagi){
                fputc('/',f);
            }
        }
```

Gambar 19 Program utama (6)

```
        fputc('\n', f);
    }

    fclose(f);
}

return 0;
}
```

Gambar 20 Program utama (7)

IV. KESIMPULAN

Penyelesaian permainan kartu 24 dapat diselesaikan dengan algoritma *brute force*. Meskipun terlihat tidak efektif, cara ini merupakan cara yang sederhana untuk mendapatkan solusi permainan kartu 24.

Berdasarkan hasil percobaan, algoritma yang penulis buat memiliki poin-poin berikut:

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	✓	

V. Repository

https://github.com/manuellaiv/Tucil1_13521051

VI. Ucapan Terima Kasih

Puji dan syukur penulis ucapkan kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya penulis dapat menyelesaikan makalah ini. Penulis ingin menyampaikan terima kasih kepada dosen pengampu mata kuliah Strategi Algoritma yaitu Pak Rinaldi Munir dan dosen-dosen lainnya yang telah memberikan ilmunya dalam mata kuliah ini. Penulis berharap makalah ini dapat bermanfaat bagi pembaca.

Referensi

[1] https://www.pagat.com/adders/24.html#cards, diakses 22 Januari 2023.
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/stima22-23.htm, diakses 22 Januari 2023.
[3] https://boardgamegeek.com/, diakses 23 Januari 2023

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 23 Januari 2023

Manuella Ivana Uli Sianipar 13521051