TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

PENYELESAIAN PERMAINAN KARTU 24 DENGAN ALGORITMA BRUTE FORCE

SEMESTER II TAHUN 2022/2023

Disusun Oleh

13521110 Yanuar Sano Nur Rasyid

PROGRAM STUDI TEKNIK INFORMATIKA SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

BAB I
DESKRIPSI MASALAH

Tugas kecil 1 ini merupakan aplikasi dari konsep algoritma *Brute Force* yang akan digunakan untuk menyelesaikan permainan kartu 24. Permainan kartu 24 merupakan permainan kartu yang bertujuan untuk mencari cara untuk mengubah 4 buah kartu secara aritmatika menghasilkan hasil akhir sejumlah 24. Kartu yang biasa digunakan pada permainan ini adalah kartu remi. Kartu remi terdiri atas 52 kartu yang terbagi menjadi empat buah suit yaitu sekop, hati, keriting, dan wajik. Setiap suit dari kartu remi tersebut terdiri atas 13 kartu yaitu As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, dan King. Dalam permainan kartu 24, nilai kartu sesuai dengan angka yang ada pada kartu tersebut dengan nilai kartu selain angka yait As sebagai 1, Jack sebagai 11, Queen sebagai 12, dan King sebagai 13. Permainan dimulai dengan mengambil 4 buah kartu secara acak dari dek. Kemudian kartu tersebut disusun sedemikian rupa agar mendapatkan hasil akhir sejumlah 24. Pengubahan nilai tersebut dilakukan dengan operasi operasi dasar matematika penjumlahan (+), pengurangan (-), perkalian (×), divisi (/) dan tanda kurung (()). Tiap kartu juga harus digunakan tepat sekali dan urutan penggunaannya bebas. Permainan berakhir saat pemain berhasil menemukan solusi agar kartu mendapatkan nilai 24.

Konsep algoritma *Brute Force* akan digunakan untuk membuat program yang mencari semua solusi yang mungkin dari permainan kartu 24. Program dibuat menggunakan bahasa pemrograman C++ dengan spesifikasi seperti berikut.

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 di atas 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

TEORI SINGKAT

2.1 Algoritma *Brute Force*

Sebuah algoritma yang memecahkan suatu masalah dengan lempan atau *straightforward* sehingga algoritma ini akan mencoba semua solusi yang mungkin dari suatu persoalan untuk menyelesaikannya.

Contoh dari algoritma *brute force* adalah mencari elemen terbesar atau terkecil dari sebuah *list.* Misalnya Diberikan sebuah senarai yang berisi n buah bilangan bulat (a1, a2, ..., an). Carilah elemen terbesar di dalam senarai tersebut. Maka solusi dengan *bruteforce* adalah membandingkan semua elemen yang ada dalam senarai hingga akhirnya mendapatkan elemen bernilai terbesar.

Berikut *pseudocode* solusi dari persoalan di atas.

```
procedure CariElemenTerbesar(input a1, a2, ..., an : integer, output maks : integer)
{ Mencari elemen terbesar di antara elemen a1, a2, ..., an.
Elemen terbesar disimpan di dalam maks.
Masukan: a1, a2, ..., an
Luaran: maks
}
Deklarasi
  k : integer

Algoritma:
  maks←a1
  for k←2 to n do
     if ak > maks then
        maks←ak
     endif
  endfor
```

BAB III

IMPLEMENTASI

Langkah penyelesaian permaianan kartu 24 dengan metode *brute force* adalah sebagai berikut.

1. Mengiterasikan semua kemungkinan susunan dari 4 kartu yang dimainkan.

   Dalam program langkah ini dilakukan dengan melakukan loop terhadap *array* yang menyimpan nilai kartu sebanyak empat kali dengan syarat indeks yang akan digunakan untuk menghitung tidak pernah sama dengan indeks yang lain.

```
// loop for all possible combination of card
for (int i = 0; i < N ; i++){
for (int j = 0; j < N; j++){
    if (i == j) continue;
for (int k = 0; k < N; k++){
    if (k == i || k == j) continue;
for (int l = 0; l < N; l++){
    if (l == i || l == j || l == k) continue;
```

2. Mengiterasikan semua kemungkinan operasi yang akan digunakan

   Sama seperti langkah sebelumnya, akan digunakan tiga buah loop untuk mendapatkan semua kemungkinan dari operasi yang mungkin, tetapi tidak memiliki syarat tidak boleh sama dari langkah sebelumnya

```
// loop for three out of four operation
for (int m = 0; m < M; m++){
for (int n = 0; n < M; n++){
for (int o = 0; o < M; o++){
```

3. Menghitung nilai yang didapatkan dari operasi aritmatika dengan semua kemungkinan susunan nilai dan angka

   Terdapat 5 kemungkinan susunan pada langkah ini yaitu

```
// ((a op1 b) op2 c) op3 d
sum = a;
if (m == 0) sum += + b;
else if (m == 1) sum -= b;
else if (m == 2) sum *= b;
else if (m == 3) sum /= (float)b;

if (n == 0) sum += c;
else if (n == 1) sum -= c;
else if (n == 2) sum *= c;
else if (n == 3) sum /= (float)c;

if (o == 0) sum += d;
else if (o == 1) sum -= d;
else if (o == 2) sum *= d;
else if (o == 3) sum /= (float)d;
```

```
// (a op1 (b op2 c)) op3 d
if (n == 0) sum1 = b + c;
else if (n == 1) sum1 = b - c;
else if (n == 2) sum1 = b * c;
else if (n == 3) sum1 = b / (float)c;

if (m == 0) sum = a + sum1;
else if (m == 1) sum = a - sum1;
else if (m == 2) sum = a * sum1;
else if (m == 3) sum = a / (float)sum1;

if (o == 0) sum += d;
else if (o == 1) sum -= d;
else if (o == 2) sum *= d;
else if (o == 3) sum /= (float)d;
```

```
// a op1 ((b op2 c) op3 d)
if (n == 0) sum1 = b + c;
else if (n == 1) sum1 = b - c;
else if (n == 2) sum1 = b * c;
else if (n == 3) sum1 = b / (float)c;

if (o == 0) sum = sum1 + d;
else if (o == 1) sum = sum1 - d;
else if (o == 2) sum = sum1 * d;
else if (o == 3) sum = sum1 / (float)d;

if (m == 0) sum = a + sum;
else if (m == 1) sum = a - sum;
else if (m == 2) sum = a * sum;
else if (m == 3) sum = a / (float)sum;
```

```
// a op1 (b op2 (c op3 d))
if (o == 0) sum1 = c + d;
else if (o == 1) sum1 = c - d;
else if (o == 2) sum1 = c * d;
else if (o == 3) sum1 = c / (float)d;

if (n == 0) sum2 = b + sum1;
else if (n == 1) sum2 = b - sum1;
else if (n == 2) sum2 = b * sum1;
else if (n == 3) sum2 = b / (float)sum1;

if (m == 0) sum = a + sum2;
else if (m == 1) sum = a - sum2;
else if (m == 2) sum = a * sum2;
else if (m == 3) sum = a / (float)sum2;
```

```
// (a op1 b) op2 (c op3 d)
if (m == 0) sum1 = a + b;
else if (m == 1) sum1 = a - b;
else if (m == 2) sum1 = a * b;
else if (m == 3) sum1 = a / (float)b;

if (o == 0) sum2 = c + d;
else if (o == 1) sum2 = c - d;
else if (o == 2) sum2 = c * d;
else if (o == 3) sum2 = c / (float)d;

if (n == 0) sum = sum1 + sum2;
else if (n == 1) sum = sum1 - sum2;
else if (n == 2) sum = sum1 * sum2;
else if (n == 3) sum = sum1 / (float)sum2;
```

Hasil dari setiap kemungkinan akan dibandingkan dengan nilai 24. Jika bernilai sama maka susunan operasi tersebut merupakan sebuah solusi dari permainan kartu 24 ini.

```
if (sum == 24) {
    term1 = "(" + std::to_string(a) + op1 + std::to_string(b) + ")";
    term2 = "(" + std::to_string(c) + op3 + std::to_string(d) + ")";
    result = term1 + op2 + term2 ;
    results.insert(result);
}
```

Implementasi programnya itu sendiri terdiri atas dua file yaitu lib.cpp dan main.cpp. File lib.cpp berisi class TwentyFour yang memiliki semua fungsi yang digunakan dalam penyelesaian permainan game 24. Program juga menggunakan beberapa library untuk membantu dalam pengolahan data dan perhitungan waktu.

Berikut adalah isi dari file lib.cpp.

```cpp
#include <iostream>
#include <string>
#include <set>
#include <ctime>
#include <fstream>

#define N 4 // number of card per game
#define M 4 // number of operator

class TwentyFour {
    private:
        std::string cards[N];
        int cardsValues[N];
        std::string operators[M] = {" + ", " - ", " * ", " / "};
        std::set <std::string> results;
```

Library dan attribute dari class.

```cpp
// Mengubah sebuah kartu (string) menjadi nilai yang sesuai (int)
// contoh "A" -> 1
int toValue(std::string s){···

// Mengubah sebuah nilai (int) menjadi kartu yang sesuai (string)
// contoh 13 -> "K"
std::string toCard(int val){···

// Memeriksa apakah input string yang diterima sesuai dengan kartu yang mungkin
bool validate(std::string s){···

// Memeriksa semua kartu yang dimainkan
bool checkCard(){···

// Melakukan input kartu dari user
bool inputCard(){···

// Meng-output nilai dari kartu yang dimainkan
void printValues(){···

// Meng-output kartu yang dimainkan
void printCards(){···

// Menghitung semua kemungkinan yang menghasilkan nilai 24
void calculate(){···

// Meng-output jumlah solusi dan solusi yang ditemukan
void printResult(){···

// Menanyakan user apakah hasil ingin disimpan
int savePrompt(){···

// Menyimpan solusi ke dalam file yang ditentukan user
void writeToFile(long long time_m, long long time_p){···
```

Semua method private di class.

```
public:
    // Menginisiasi game 24 dengan input dari user
    void newGame(){
        while(!inputCard());

        for (int i = 0; i < N; i++) {
            cardsValues[i] = toValue(cards[i]);
        }
    }

    // Menginisiasi game 24 dengan kartu random
    void randomGame(){
        srand ( time(NULL) );
        for (int i = 0; i < N; i++) {
            cardsValues[i] = rand() % 13 + 1;
        }
        for (int i = 0; i < N; i++) {
            cards[i] = toCard(cardsValues[i]);
        }
        std::cout << "Kartu random: " << '\n';
        printCards();
    }

    // Menghasilkan solusi yang ditemukan
    void resultGame(){
        calculate();
        printResult();
    }

    // Mengakhiri game dengan save prompt
    void endGame(long long time_m, long long time_p){
        if(savePrompt()) writeToFile(time_m, time_p);
    }
```

Semua method public di class.

Sementara itu, file main.cpp berisi menu utama program, penggunaan class, dan perhitungan waktu.

Berikut adalah isi dari file main.cpp

```cpp
#include <limits>
#include <chrono>
#include "lib.cpp"

// Menu utama dari aplikasi
// output yang mungkin hanya 1 untuk input manual atau 2 untuk rando
int menu(){
    std::string input;
    std::cout << "Selamat datang ke Make It 24 Solver!" << '\n';
    std::cout << "Menu:" << '\n';
    std::cout << "1. Memasukkan kartu sendiri" << '\n';
    std::cout << "2. Mendapatkan kartu secara random" << '\n';

    while(true){
        std::cout << "Pilihan menu: ";
        std::cin >> input;
        if(input == "1") {
            return 1;
        } else if (input == "2"){
            return 2;
        } else {
            std::cout << "Menu tidak dikenal. Silahkan coba lagi" << '\n';
        }
    }
}
```

Fungsi menu dan library yang digunakan.

```cpp
int main(){

    TwentyFour game;
    int input = menu();

    input == 1 ? game.newGame() : game.randomGame();

    auto start = std::chrono::high_resolution_clock::now();
    game.resultGame();
    auto end = std::chrono::high_resolution_clock::now();

    auto exec_time_m = std::chrono::duration_cast<std::chrono::milliseconds>(end-start).count();
    auto exec_time_p = std::chrono::duration_cast<std::chrono::microseconds>(end-start).count();

    if(exec_time_m == 0){
        std::cout << "Waktu eksekusi algoritma: " << exec_time_p << " microsecond" << '\n';
    } else {
        std::cout << "Waktu eksekusi algoritma: " << exec_time_m << " millisecond" << '\n';
    }

    game.endGame(exec_time_m, exec_time_p);

    return 0;
}
```

Fungsi main dengan perhitungan waktu algoritma.

BAB IV

EKSPERIMEN

Tampilan program saat pertama kali dijalankan

```
Selamat datang ke Make It 24 Solver!
Menu:
1. Memasukkan kartu sendiri
2. Mendapatkan kartu secara random
Pilihan menu:
```

4.1 Test Case 1

Kartu A, 8, 9, Q

```
Masukan 4 kartu yang akan dihitung :
A 8 9 Q
48 solusi ditemukan
((1 * 12) - 9) * 8
((1 + 9) - 8) * 12
((1 - 8) + 9) * 12
((12 * 1) - 9) * 8
((12 - 9) * 1) * 8
((12 - 9) * 8) * 1
((12 - 9) * 8) / 1
((12 - 9) / 1) * 8
((12 / 1) - 9) * 8
((9 + 1) - 8) * 12
((9 - 8) + 1) * 12
(1 * (12 - 9)) * 8
(1 * 8) * (12 - 9)
(1 + (9 - 8)) * 12
(1 - (8 - 9)) * 12
(12 - (1 * 9)) * 8
(12 - (9 * 1)) * 8
(12 - (9 / 1)) * 8
(12 - 9) * (1 * 8)
(12 - 9) * (8 * 1)
(12 - 9) * (8 * 1)
(12 - 9) * (8 / 1)
(12 - 9) / (1 / 8)
(8 * (12 - 9)) * 1
(8 * (12 - 9)) / 1
(8 * 1) * (12 - 9)
(8 / 1) * (12 - 9)
(9 + (1 - 8)) * 12
12 * (1 - (8 - 9))
12 * (9 + (1 - 8))
12 * (9 - (8 - 1))
8 * ((1 * 12) - 9)
8 * ((12 * 1) - 9)
8 * ((12 - 9) * 1)
8 * ((12 - 9) / 1)
8 * ((12 / 1) - 9)
8 * (1 * (12 - 9))
8 * (12 - (1 * 9))
8 * (12 - (9 * 1))
8 * (12 - (9 / 1))
8 / (1 / (12 - 9))
Waktu eksekusi algoritma: 9 millisecond
Apakah ingin menyimpan solusi? [y/N]
```

4.2 Test Case 2

Kartu 7, 5, 3, 8

```
Masukan 4 kartu yang akan dihitung :
7 5 3 8
30 solusi ditemukan
((3 * 7) + 8) - 5
((3 * 7) - 5) + 8
((5 * 7) - 3) - 8
((5 * 7) - 8) - 3
((7 * 3) + 8) - 5
((7 * 3) - 5) + 8
((7 * 5) - 3) - 8
((7 * 5) - 8) - 3
((8 - 5) * 7) + 3
(3 * 7) + (8 - 5)
(3 * 7) - (5 - 8)
(5 * 7) - (3 + 8)
(5 * 7) - (8 + 3)
(7 * (8 - 5)) + 3
(7 * 3) + (8 - 5)
(7 * 3) - (5 - 8)
```

```
(5 * 7) - (8 + 3)
(7 * (8 - 5)) + 3
(7 * 3) + (8 - 5)
(7 * 3) - (5 - 8)
(7 * 5) - (3 + 8)
(7 * 5) - (8 + 3)
(8 + (3 * 7)) - 5
(8 + (7 * 3)) - 5
(8 - 5) + (3 * 7)
(8 - 5) + (7 * 3)
3 + ((8 - 5) * 7)
3 + (7 * (8 - 5))
3 - ((5 - 8) * 7)
3 - (7 * (5 - 8))
8 + ((3 * 7) - 5)
8 + ((7 * 3) - 5)
8 - (5 - (3 * 7))
8 - (5 - (7 * 3))
Waktu eksekusi algoritma: 6 millisecond
Apakah ingin menyimpan solusi? [y/N]
```

4.3 Test Case 3

Kartu J, 4, A, 4

```
Masukan 4 kartu yang akan dihitung :
J 4 A 4
8 solusi ditemukan
((11 - 1) - 4) * 4
((11 - 4) - 1) * 4
(11 - (1 + 4)) * 4
(11 - (4 + 1)) * 4
4 * ((11 - 1) - 4)
4 * ((11 - 4) - 1)
4 * (11 - (1 + 4))
4 * (11 - (4 + 1))
Waktu eksekusi algoritma: 2 millisecond
Apakah ingin menyimpan solusi? [y/N]
```

4.4 Test Case 4

Kartu K, 6, 3, K

```
Masukan 4 kartu yang akan dihitung :
K 6 3 k
8 solusi ditemukan
((13 / 13) + 3) * 6
(13 + 13) - (6 / 3)
(13 - (6 / 3)) + 13
(3 + (13 / 13)) * 6
13 + (13 - (6 / 3))
13 - ((6 / 3) - 13)
6 * ((13 / 13) + 3)
6 * (3 + (13 / 13))
Waktu eksekusi algoritma: 2 millisecond
Apakah ingin menyimpan solusi? [y/N]
```

4.5 Test Case 5

Kartu 8, 7, J, A

```
Masukan 4 kartu yang akan dihitung :
8 7 J A
12 solusi ditemukan
((11 - 1) - 7) * 8
((11 - 7) - 1) * 8
(1 + 7) * (11 - 8)
(11 - (1 + 7)) * 8
(11 - (7 + 1)) * 8
(11 - 8) * (1 + 7)
(11 - 8) * (7 + 1)
(7 + 1) * (11 - 8)
8 * ((11 - 1) - 7)
8 * ((11 - 7) - 1)
8 * (11 - (1 + 7))
8 * (11 - (7 + 1))
Waktu eksekusi algoritma: 2 millisecond
Apakah ingin menyimpan solusi? [y/N]
```

4.6 Test Case 6

Kartu J, Q, A, A

```
Masukan 4 kartu yang akan dihitung :
J Q  A A
168 solusi ditemukan
((1 * 1) + 11) + 12
((1 * 1) + 12) + 11
((1 * 11) + 1) + 12
((1 * 11) + 12) + 1
((1 * 12) + 1) + 11
((1 * 12) + 11) + 1
((1 + 11) * 1) + 12
((1 + 11) + 12) * 1
((1 + 11) + 12) / 1
((1 + 11) / 1) + 12
((1 + 12) * 1) + 11
((1 + 12) + 11) * 1
((1 + 12) + 11) / 1
((1 + 12) / 1) + 11
((1 / 1) + 11) + 12
((1 / 1) + 12) + 11
((11 * 1) + 1) + 12
((11 * 1) + 12) + 1
((11 + 1) * 1) + 12
((11 + 1) + 12) * 1
((11 + 1) + 12) / 1
```

...

```
11 + (1 + (1 * 12))
11 + (1 + (12 * 1))
11 + (1 + (12 / 1))
11 + (12 + (1 * 1))
11 + (12 + (1 / 1))
12 + ((1 * 1) + 11)
12 + ((1 * 11) + 1)
12 + ((1 + 11) * 1)
12 + ((1 + 11) / 1)
12 + ((1 / 1) + 11)
12 + ((11 * 1) + 1)
12 + ((11 + 1) * 1)
12 + ((11 + 1) / 1)
12 + ((11 / 1) + 1)
12 + (1 * (1 + 11))
12 + (1 * (11 + 1))
12 + (1 + (1 * 11))
12 + (1 + (11 * 1))
12 + (1 + (11 / 1))
12 + (11 + (1 * 1))
12 + (11 + (1 / 1))
Waktu eksekusi algoritma: 33 millisecond
Apakah ingin menyimpan solusi? [y/N]
```

4.7 Test Case 7

Kartu 3, 7, 6, 4

```
Masukan 4 kartu yang akan dihitung :
3 7 6 4
Tidak ditemukan solusi
Waktu eksekusi algoritma: 318 microsecond
Apakah ingin menyimpan solusi? [y/N]
y
Masukan nama file yang akan disimpan
./test/result7
```

DAFTAR PUSTAKA

Munir, Rinaldi. 2023. IF2211 Strategi Algoritma
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf, diakses 25 Januari 2023

LAMPIRAN

Cek List Tambahan

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	✓	

Link Repository

https://github.com/yansans/Tucil1_13521110