**Tugas Kecil 1 IF 2211 Strategi Algoritma**

**Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force**

****

**Hosea Nathanael Abetnego 13521057**

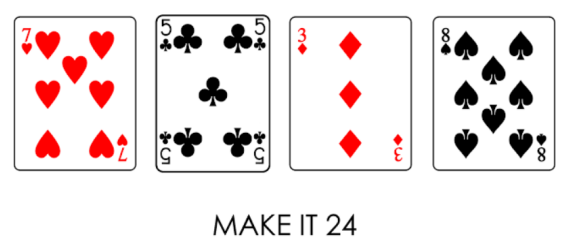
**Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2023**

1. **Deskripsi Persoalan**

****

Permainan kartu 24 adalah permainan kartu aritmatika dengan tujuan mencari cara untuk mengubah 4 buah angka random sehingga mendapatkan hasil akhir sejumlah 24. 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. **Algoritma**

Program ini menggunakan paradigma Algoritma Brute Force, yaitu algoritma yang menggunakan pendekatan yang lempang untuk memecahkan suatu persoalan. Pada program ini, berikut adalah pendekatan yang dilakukan dalam menyelesaikan masalah

1. Program harus memperoleh 4 buah angka yang akan dihitung melalui input user maupun secara random.
2. 4 angka yang diperoleh tersebut kemudian dicari semua kombinasi posisi angka.
3. Dicari pula semua kombinasi operator, sebanyak 3 operator, untuk perhitungan.
4. Perhitungan waktu *runtime* dimulai dan semua kombinasi angka dan operator tersebut kemudian dihitung berdasarkan 5 kombinasi posisi tanda kurung sehingga diperoleh hasilnya. Contoh : ((a [op] b) [op] c) [op] d)
5. Hasil yang diperoleh tersebut kemudian dibandingkan dengan nilai 24 dan jika sesuai, jumlah solusi bertambah dan solusi tersebut disimpan.
6. Jumlah solusi dan semua solusinya kemudian ditampilkan kepada pengguna beserta waktu *runtime*nya.

Penerapan pendekatan program ini dilakukan menggunakan bahasa C++.

1. **Kode Program**

Keseluruhan program berada pada *file* main.cpp. Terdapat beberapa fungsi berbeda yang digunakan untuk menerapkan pendekatan-pendekatan di atas.

1. Fungsi *count*

Fungsi ini digunakan untuk mengoperasikan 2 buah nilai dengan operator yang diinginkan. Parameter fungsi tersebut dalah 2 buah nilai float dan sebuah *char* yang merupakan operatornya.

**float** count(**float** num1, **float** num2, **char** op)

{

    if (op == '+')

    {

        return num1 + num2;

    }

    else if (op == '-')

    {

        return num1 - num2;

    }

    else if (op == '\*')

    {

        return num1 \* num2;

    }

    else if (op == '/')

    {

        return num1 / num2;

    }

    else

    {

        return 1.0;

    }

}

1. Fungsi *combination*

Fungsi ini digunakan untuk menghitung keseluruhan operasi dari 4 buah angka tersebut dengan 5 posisi tanda kurung yang berbeda-beda. Parameter *comb* menentukan kombinasi posisi tanda kurung yang mana yang akan digunakan.

**float** combination(**float** numbers[4], **char** ops[3], **int** comb)

{

    if (comb == 1) */\*((A B) C) D\*/*

    {

        return count(count(count(numbers[0], numbers[1], ops[0]), numbers[2], ops[1]), numbers[3], ops[2]);

    }

    else if (comb == 2) */\*(A (B C)) D\*/*

    {

        return count(count(numbers[0], count(numbers[1], numbers[2], ops[1]), ops[0]), numbers[3], ops[2]);

    }

    else if (comb == 3) */\*A (B (C D))\*/*

    {

        return count(numbers[0], count(numbers[1], count(numbers[2], numbers[3], ops[2]), ops[1]), ops[0]);

    }

    else if (comb == 4) */\*A ((B C) D)\*/*

    {

        return count(numbers[0], count(count(numbers[1], numbers[2], ops[1]), numbers[3], ops[2]), ops[0]);

    }

    else if (comb == 5) */\*(A B) (C D)\*/*

    {

        return count(count(numbers[0], numbers[1], ops[0]), count(numbers[2], numbers[3], ops[2]), ops[1]);

    }

}

1. Fungsi *Ops*

Fungsi ini digunakan untuk memperoleh semua kombinasi operator untuk memenuhi 3 posisi operator yang diperlukan dalam perhitungan 4 angka tersebut

**void** Ops(**char** allOps[64][3])

{

**char** ops[4] = {'+', '-', '\*', '/'};

**int** cnt = 0;

    for (**int** i = 0; i < 4; i++)

    {

        for (**int** j = 0; j < 4; j++)

        {

            for (**int** k = 0; k < 4; k++)

            {

                allOps[cnt][0] = ops[i];

                allOps[cnt][1] = ops[j];

                allOps[cnt][2] = ops[k];

                cnt++;

            }

        }

    }

}

1. Fungsi *Nums*

Fungsi ini digunakan untuk memperoleh semua kombinasi posisi 4 angka yang ingin dioperasikan.

**void** Ops(**char** allOps[64][3])

{

**char** ops[4] = {'+', '-', '\*', '/'};

**int** cnt = 0;

    for (**int** i = 0; i < 4; i++)

    {

        for (**int** j = 0; j < 4; j++)

        {

            for (**int** k = 0; k < 4; k++)

            {

                allOps[cnt][0] = ops[i];

                allOps[cnt][1] = ops[j];

                allOps[cnt][2] = ops[k];

                cnt++;

            }

        }

    }

}

1. Fungsi *numsManual*

Fungsi ini digunakan untuk mengambil input 4 buah angka secara manual dari user. Fungsi ini menerima input string yang kemudian memanfaatkan *hash* untuk memperoleh nilai float dari input tersebut dan memastikan input sesuai kemudian input yang sudah menjadi float tersebut disimpan dalam sebuah *array*.

**void** numsManual(**float** numbers[4])

{

    map<string, **float**> HashInput;

    HashInput["2"] = 2.0;

    HashInput["3"] = 3.0;

    HashInput["4"] = 4.0;

    HashInput["5"] = 5.0;

    HashInput["6"] = 6.0;

    HashInput["7"] = 7.0;

    HashInput["8"] = 8.0;

    HashInput["9"] = 9.0;

    HashInput["10"] = 10.0;

    HashInput["A"] = 1.0;

    HashInput["J"] = 11.0;

    HashInput["Q"] = 12.0;

    HashInput["K"] = 13.0;

    cout << "Masukkan 4 angka atau huruf :\n(A, 2-10, J, Q, K)\n";

    for (**int** i = 0; i < 4; i++)

    {

        string input;

        cin >> input;

        while (HashInput.find(input) == HashInput.end())

        {

            cout << "Input tidak sesuai, silahkan input perbaikan sebanyak input yang tidak sesuai\n";

            cin >> input;

        }

        numbers[i] = HashInput[input];

    }

}

1. Fungsi numsRandom

Fungsi ini digunakan untuk memperoleh 4 angka secara *random* untuk dioperasikan. Akan diambil sebuah angka *random* yang kemudian akan dicari nilainya tersebut pada sebuah list yang berisikan pilihan-pilihan angka dan huruf yang dapat dioperasikan kemudian disimpan pada semua *array of float*.

**void** numsRandom(**float** numbers[])

{

    string cardList[13] = {"A",

                           "2",

                           "3",

                           "4",

                           "5",

                           "6",

                           "7",

                           "8",

                           "9",

                           "10",

                           "J",

                           "Q",

                           "K"};

    cout << "Kartu yang diperoleh :\n";

    srand(time(0));

    for (**int** i = 0; i < 4; i++)

    {

**int** number = rand() % 13;

        cout << cardList[number] << " ";

        numbers[i] = (**float**)(number + 1);

    }

    cout << "\n\n";

}

1. Fungsi mainCounter

Fungsi ini merupakan fungsi utama dari program. Fungsi ini yang akan mengolah 4 angka yang telah diperoleh melalui semua kombinasi poisisi tanda kurung, memastikan hasilnya, dan menyimpannya. Semua hasil yang diperoleh kemudian ditampilkan bersama dengan jumlah solusi dan waktu pengolahan. User kemudian diberi pilihan untuk menyimpan solusi dalam file atau tidak. File yang ingin disimpan kemudian disimpan pada folder *test*.

**void** mainCounter(**float** numbers[4])

{

**char** allOps[64][3];

**float** allNums[24][4];

**auto** timeStart = std::chrono::high\_resolution\_clock::now();

    Ops(allOps);

    Nums(allNums, numbers);

    std::stringstream solutions;

**int** cntSolution = 0;

    for (**int** i = 0; i < 24; i++)

    {

        for (**int** j = 0; j < 64; j++)

        {

            if (abs(combination(allNums[i], allOps[j], 1) - 24) < 0.00001) */\*((A B) C) D\*/*

            {

                solutions << "((" << allNums[i][0] << allOps[j][0] << allNums[i][1] << ")" << allOps[j][1] << allNums[i][2] << ")" << allOps[j][2] << allNums[i][3] << endl;

                cntSolution++;

            }

            if (abs(combination(allNums[i], allOps[j], 2) - 24) < 0.00001) */\*(A (B C)) D\*/*

            {

                solutions << "(" << allNums[i][0] << allOps[j][0] << "(" << allNums[i][1] << allOps[j][1] << allNums[i][2] << "))" << allOps[j][2] << allNums[i][3] << endl;

                cntSolution++;

            }

            if (abs(combination(allNums[i], allOps[j], 3) - 24) < 0.00001) */\*A (B (C D))\*/*

            {

                solutions << allNums[i][0] << allOps[j][0] << "(" << allNums[i][1] << allOps[j][1] << "(" << allNums[i][2] << allOps[j][2] << allNums[i][3] << "))" << endl;

                cntSolution++;

            }

            if (abs(combination(allNums[i], allOps[j], 4) - 24) < 0.00001) */\*A ((B C) D)\*/*

            {

                solutions << allNums[i][0] << allOps[j][0] << "((" << allNums[i][1] << allOps[j][1] << allNums[i][2] << ")" << allOps[j][2] << allNums[i][3] << ")" << endl;

                cntSolution++;

            }

            if (abs(combination(allNums[i], allOps[j], 5) - 24) < 0.00001) */\*(A B) (C D)\*/*

            {

                solutions << "(" << allNums[i][0] << allOps[j][0] << allNums[i][1] << ")" << allOps[j][1] << "(" << allNums[i][2] << allOps[j][2] << allNums[i][3] << ")" << endl;

                cntSolution++;

            }

        }

    }

**auto** timeEnd = std::chrono::high\_resolution\_clock::now();

**auto** runTime = std::chrono::duration\_cast<std::chrono::nanoseconds>(timeEnd - timeStart);

    cout << "Terdapat " << cntSolution << " solusi\n";

    cout << solutions.str() << endl;

    cout << "Runtime :" << runTime.count() \* 1e-9 << " Detik\n" << endl;

    cout << "Apakah ingin disimpan dalam file?\n(y/Y untuk ya n/N untuk tidak)\n";

**char** choice;

    cin >> choice;

    if (choice == 'y' || choice == 'Y')

    {

        string path;

        cout << "Masukkan nama file :\n";

        cin >> path;

        ofstream file;

        file.open("./test/" + path + ".txt");

        file << "Terdapat " << cntSolution << " solusi" << endl

             << solutions.str() << endl

             << "Runtime :" << runTime.count() \* 1e-9 << endl;

        file.close();

        cout << "Berhasil menuliskan hasil pada file " << path << ".txt\nThank you and goodbye :)\n\n";

    }

    else if (choice == 'n' || choice == 'N')

    {

        cout << "Ok, bye >:(\n\n";

    }

}

1. Fungsi Main

Fungsi ini yang akan memulai program. Akan ditampilkan pilihan untuk memperoleh 4 angka melalui manual dari user atau secara random atau menghentikan program. Angka yang diperoleh diberikan kepada fungsi mainCounter.

**int** main()

{

**float** numbers[4];

**bool** running = true;

    while (running)

    {

**int** choice;

        cout << "24 Game Solver with Brute Force\nBy Hosea N.A.\n\n";

        cout << "Pilih cara input\n";

        cout << "Ketik 1 untuk input manual\nKetik 2 untuk input random\nKetik 3 untuk keluar dari program\n";

        cin >> choice;

        if (choice == 1)

        {

            numsManual(numbers);

            mainCounter(numbers);

        }

        else if (choice == 2)

        {

            numsRandom(numbers);

            mainCounter(numbers);

        }

        else if (choice == 3)

        {

            running = false;

        }

    }

    return 0;

}

1. **Percobaan**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi tanpa kesalahan | ✓ |  |
| 1. Program berhasil *running* | ✓ |  |
| 1. Program dapat membaca input / generate sendiri dan memberikan luaran | ✓ |  |
| 1. Solusi yang diberikan program memenuhi (berhasil mencapai 24) | ✓ |  |
| 1. Program dapat menumpan solusi dalam file teks | ✓ |  |

Berikut adalah beberapa hasil percobaan dari program di atas.

1. Percobaan 1

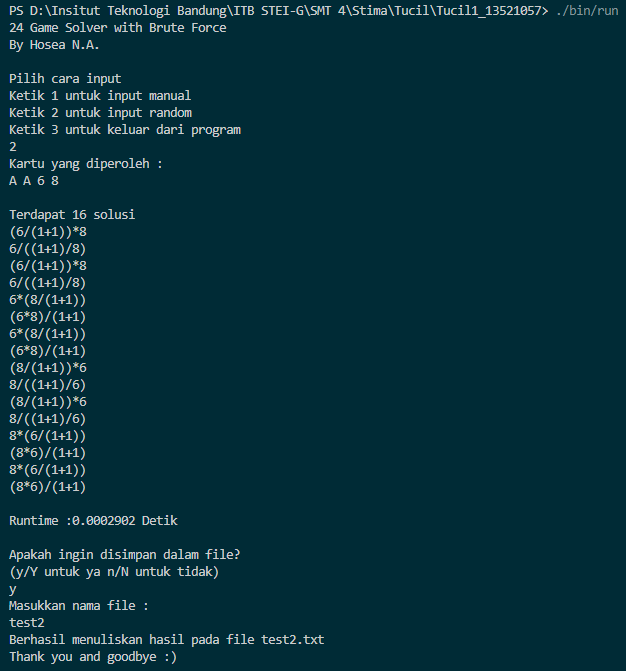
Text

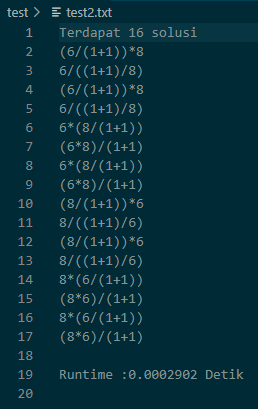
Description automatically generated

Text

Description automatically generated

1. Percobaan 2





1. Percobaan 3

Text

Description automatically generated

Text

Description automatically generated

1. Percobaan 4

Text

Description automatically generated

Graphical user interface

Description automatically generated

1. Percobaan 5

Text

Description automatically generated

Text

Description automatically generated

1. Percobaan 6

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

***Link Repository***

https://github.com/HoseaNA/Tucil1\_13521057

***Referensi***

1. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf>
2. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag2.pdf>
3. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tucil1-Stima-2023.pdf>