

Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Semester II tahun 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force



Rachel Gabriela Chen – 13521044

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

2023

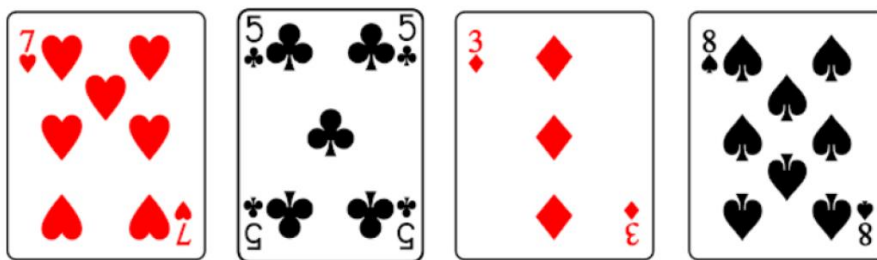
DAFTAR ISI

DAFTAR ISI.....	1
BAB I : DESKRIPSI MASALAH	2
BAB 2 : ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN PERMAINAN KARTU 24	3
BAB 3 : IMPLEMENTASI PROGRAM DENGAN C++	5
BAB 4 : EKSPERIMEN	12
LAMPIRAN	16
Github Repository	16
Tabel Spesifikasi	16

BAB I : 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. (Paragraf di atas dikutip dari: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2015-2016/Makalah-2016/MakalahStima-2016-038.pdf>).

Laporan ini membahas program yang dapat digunakan untuk mencari solusi dari permainan kartu 24 dengan algoritma *Brute Force*.



Gambar 1. Permainan Kartu 24

BAB 2 : ALGORITMA BRUTE FORCE UNTUK PENYELESAIAN PERMAINAN KARTU 24

Algoritma Brute Force dapat digunakan untuk menyelesaikan permainan kartu 24 dengan mencoba semua kemungkinan ekspresi matematika yang valid dengan angka-angka yang didapatkan pada kartu dan operator yang dipilih. Jika hasil evaluasi ekspresi matematika tersebut sama dengan 24, maka ekspresi tersebut diambil sebagai salah satu solusi.

Langkah 1

Cari semua permutasi dari 4 angka yang digunakan. Permutasi dicari dengan menggunakan fungsi rekursif:

Basis : Jika indeks kiri dan kanan sama, simpan permutasi.

Rekurens : Tukar elemen paling kiri dari array dengan setiap elemen pada sub-array.

Untuk semua permutasi, cari semua ekspresi matematika dan hasil evaluasinya.

Langkah 2

Digunakan fungsi rekursif untuk mempartisi *list* angka dan menghitung semua kemungkinan pengelompokan dengan tanda kurung. Partisi 4 angka tersebut menjadi “kiri” dan “kanan” untuk semua kemungkinan partisi, yaitu kiri 1 angka pertama dan kanan 3 angka selanjutnya, kiri 2 angka pertama dan kanan 2 angka selanjutnya, kiri 3 angka pertama dan kanan 1 angka selanjutnya. Dicari kemungkinan ekspresi matematika dan hasil evaluasi dari setiap partisi menggunakan fungsi yang sama secara rekursif. Basisnya ada 2, yaitu:

- Jika hasil partisi hanya 1 angka, kembalikan angka tersebut sebagai hasil evaluasi.
- Jika hasil partisi 2 angka, operasikan kedua angka tersebut dengan setiap operator $+$, $-$, $/$, $*$, kemudian kembalikan hasil operasinya dan juga ekspresi matematikanya.

Jika panjang partisi 3, *list* tersebut dipartisi lagi menggunakan fungsi yang sama secara rekursif. Operasikan setiap kemungkinan operasi “kiri” dan setiap kemungkinan operasi “kanan” dengan setiap operator yang ada. Dengan fungsi ini, setiap kemungkinan operasi matematika dari 4 angka akan diperiksa.

Langkah 3

Untuk setiap hasil operasi pada fungsi rekursif di langkah 2, cek apakah hasil evaluasinya adalah 24. Jika hasilnya 24, simpan ekspresi matematika tersebut.

Algoritma ini tidak mencari *similar solutions*. Artinya, solusi yang secara matematis sama (karena sifat komutatif, asosiatif, dan distributif) dihitung sebagai solusi yang berbeda. Misalnya, $((6+6)+6)+6$ dihitung sebagai solusi yang berbeda dengan $(6+(6+6))+6$.

BAB 3 : IMPLEMENTASI PROGRAM DENGAN C++

Algoritma pada Bab 2 diimplementasikan dengan sebuah program C++. Program diimplementasikan secara modular dengan 4 file, yaitu main.cpp yang memuat program utama, utils.cpp yang memuat fungsi dan prosedur interaksi pengguna, permutation.cpp yang memuat fungsi-fungsi untuk mencari permutasi dengan pengulangan, dan solver.cpp yang memuat algoritma penyelesaian permainan kartu 24.

main.cpp

main.cpp memuat splash screen dan merupakan program utama aplikasi.



```
1 #include <iostream>
2 #include "../include/utils.h"
3
4 using namespace std;
5
6 int main()
7 {
8     cout << endl;
9     cout << endl;
10    cout << "
11    .oooo.      .o      .oooooo. .o      ooooo      oooooo\n"
12    ".dPooY88b   .d88    d8P' `Y8 d8P' `Y8b `888'    `888.  .8' `888' `8 `888 `Y88.\n"
13    "   j8P'   .d'888   Y88bo. 888 888 888   `888. .8' 888 888 888 .d88'\n"
14    "   .d8P' .d' 888   \"Y8888o. 888 888 888   `888. .8' 888oooo8 888ooo88P'\n"
15    "   .dP'   88ooo888oo \"Y88b 888 888 888   `888.8' 888  \" 888'88b. \n"
16    "   .oP   .o 888    oo   .d8P `88b d88' 888   o 888' 888  o 888 `88b. \n"
17    "8888888888  o888o 88888888P' `Y8bood8P' o888ooooood8 `8'  o888ooooood8 o888o  o888o\n"
18    << endl;
19    cout << "
20    Welcome to 24 Card Game Solver!
21    24 Card Game is a game where the players are given 4 random cards and they have to
22    have to make math expression with those numbers so that it evaluates to 24.
23    Legal operators : +, -, *, /, () \n";
24    menu();
25    return 0;
26 }
```

Gambar 2. main.cpp

utils.cpp

utils.cpp memuat fungsi dan prosedur untuk interaksi i/o pengguna. Pertama, terdapat prosedur menu() yang digunakan untuk memilih kartu secara acak atau input dari pengguna. 1 untuk kartu acak, 2 untuk input kartu dari pengguna, 0 untuk keluar. Terdapat juga validasi input pengguna.

```

1 void menu()
2 {
3     string option;
4     cout << endl;
5     cout << "Menu:\n"
6         << "1. Randomize cards\n"
7         << "2. Input cards\n"
8         << "0. Exit \n"
9     << endl;
10    cout << "Option (1/2/0): ";
11    getline(cin, option);
12    if (option == "1")
13    {
14        getSolution(1);
15    }
16    else if (option == "2")
17    {
18        getSolution(2);
19    }
20    else if (option == "0")
21    {
22        cout << endl;
23        cout << "##### " << endl;
24        cout << "                Thanks for using 24 Solver                " << endl;
25        cout << "##### " << endl;
26        cout << endl;
27    }
28    else
29    {
30        cout << "Unknown input! Please only input 1/2/0" << endl;
31        menu();
32    }
33 }

```

Gambar 3. menu

Terdapat variabel global berupa map yang menyimpan nilai setiap kartu.

```

1 map<string, int> toInt{{"A", 1}, {"2", 2}, {"3", 3}, {"4", 4}, {"5", 5}, {"6", 6}, {"7", 7}, {"8", 8}, {"9", 9}, {"10", 10}, {"J", 11}, {"Q", 12}, {"K", 13}};

```

Gambar 4. map toInt

Prosedur `getSolution()` mengambil input *nums* tergantung pilihan user. Jika pilihan 1, *nums* diambil secara acak melalui fungsi `randomizeCards`, jika 2 *nums* diambil dari input user lewat fungsi `getInput`. Setelah itu, solusi dicari dengan memanggil fungsi `solveAll` yang diimplementasikan di `solver.cpp`. Setelah solusi dicetak, user dapat memilih untuk menyimpan ke hasil ke file.

```

1 void getSolution(int option)
2 {
3     vector<int> nums;
4     if (option == 1)
5     {
6         nums = randomizeCards();
7     }
8     else
9     {
10        nums = getInput();
11    }
12    auto start = high_resolution_clock::now();
13    vector<string> ans = solveAll(nums);
14    auto stop = high_resolution_clock::now();
15    float duration = (duration_cast<microseconds>(stop - start)).count() / (float)1000;
16    int cnt = ans.size();
17    int i = 0;
18    if (cnt != 0)
19    {
20        cout << endl;
21        cout << ans.size() << " solutions found!" << endl;
22        for (auto n : ans)
23        {
24            // substr is used to removed the first and last unnecessary brackets
25            cout << "Solution #" << i + 1 << ": " << n.substr(1, n.size() - 2) << endl;
26            i++;
27        }
28    }
29    else
30    {
31        cout << "No solution found" << endl;
32    }
33    cout << setprecision(3) << fixed;
34    cout << "Execution time: " << duration << " ms" << endl;
35    saveToFile(ans, nums);
36 }

```

Gambar 5. getSolution

Prosedur getInput() menerima input dari user dengan getline. Input user kemudian diparsing dengan stringstream. Tujuannya agar input user yang tidak valid, seperti "A A A A A A" atau "A A A" juga dapat divalidasi. Jika input user tidak valid, user akan terus dimintai input.

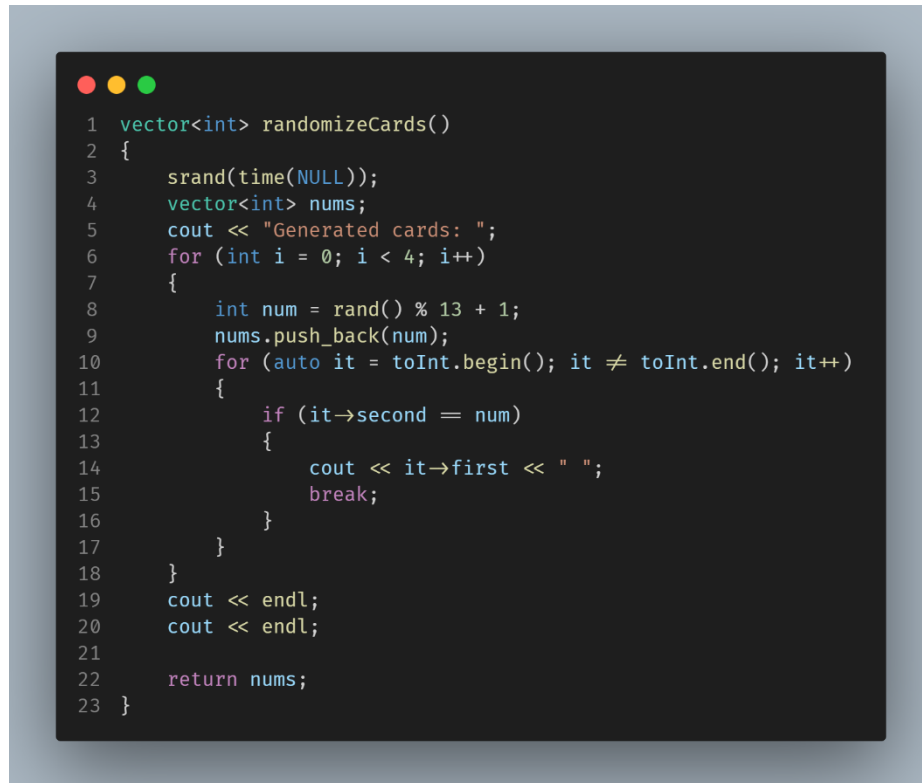
```

1 vector<int> getInput()
2 {
3     cout << endl;
4     cout << "Input 4 cards (integers 2-9, A, J, K, Q) separated by space." << endl;
5     cout << "e. g. \"A J 3 4\"." << endl;
6     cout << "Cards: ";
7     string input;
8     getline(cin, input);
9     vector<int> res;
10    stringstream ss(input);
11    string chr;
12    while (ss >> chr)
13    {
14        auto it = toInt.find(chr);
15        if (it != toInt.end())
16        {
17            res.push_back(it->second);
18        }
19        else
20        {
21            cout << chr << " is invalid." << endl;
22            cout << "Valid input is integer 2-9, A, Q, K, J" << endl;
23            return getInput();
24        }
25    }
26    if (res.size() != 4)
27    {
28        cout << "Please input exactly 4 cards separated by space." << endl;
29        return getInput();
30    }
31    return res;
32 }

```

Gambar 6. getInput

Fungsi `randomizeCards()` men-generate kartu secara acak dengan random seeder `time`.



```
1  vector<int> randomizeCards()
2  {
3      srand(time(NULL));
4      vector<int> nums;
5      cout << "Generated cards: ";
6      for (int i = 0; i < 4; i++)
7      {
8          int num = rand() % 13 + 1;
9          nums.push_back(num);
10         for (auto it = toInt.begin(); it != toInt.end(); it++)
11         {
12             if (it->second == num)
13             {
14                 cout << it->first << " ";
15                 break;
16             }
17         }
18     }
19     cout << endl;
20     cout << endl;
21
22     return nums;
23 }
```

Gambar 7. randomizeCards

Prosedur `saveToFile(vector<string> ans, vector<int> nums)` memberikan pilihan kepada user untuk menyimpan solusi ke file. Jika user memilih untuk menyimpan solusi, solusi akan disimpan dengan kartu-kartu yang digunakan sebagai nama file txt. File disimpan di folder test.

```

1 void saveToFile(vector<string> ans, vector<int> nums)
2 {
3     string option;
4     cout << endl;
5     cout << "Save solutions to file (Y/N)? ";
6     getline(cin, option);
7     if (option == "N" || option == "n")
8     {
9         menu();
10    }
11    else if (option == "Y" || option == "y")
12    {
13        string filename = "";
14        int i = 0;
15        for (auto n : nums)
16        {
17            for (auto it = toInt.begin(); it != toInt.end(); it++)
18            {
19                if (it->second == n)
20                {
21                    filename += it->first;
22                    break;
23                }
24            }
25            if (i != 3)
26            {
27                filename += "_";
28            }
29            i++;
30        }
31        filename += ".txt";
32        ofstream fout("../test/" + filename);
33        if (fout.is_open())
34        {
35            int cnt = ans.size();
36            int i = 0;
37            if (cnt != 0)
38            {
39                fout << ans.size() << " solutions found!" << endl;
40                for (auto n : ans)
41                {
42                    // substr is used to removed the first and last unnecessary brackets
43                    fout << "Solution #" << i + 1 << ": " << n.substr(1, n.size() - 2) << endl;
44                    i++;
45                }
46            }
47            else
48            {
49                fout << "No solution found" << endl;
50            }
51            cout << "Successfully saved to \"test\" folder as " << filename << endl;
52        }
53        else
54        {
55            cout << "An error ocured. The file was not saved." << endl;
56        }
57        menu();
58    }
59    else
60    {
61        cout << "Input is invalid!" << endl;
62        saveToFile(ans, nums);
63    }
64 }

```

Gambar 8. saveToFile

solver.cpp

Fungsi solveAll (vector<int> input) mengembalikan semua operasi matematika yang menghasilkan angka 24 dari setiap permutasi *nums*.

```

1  vector<string> solveAll(vector<int> input)
2  {
3      vector<string> ret;
4      vector<vector<int>> nums = getPermutation(input);
5      for (auto n : nums)
6      {
7          vector<pair<float, string>> ans = solve(n);
8          for (auto i : ans)
9          {
10             ret.push_back(i.second);
11         }
12     }
13     return ret;
14 }
15

```

Gambar 9. solveAll

Fungsi solve merupakan implementasi dari fungsi rekursif pada bab 2 yang mem-bruteforce semua kemungkinan operasi yang ada.

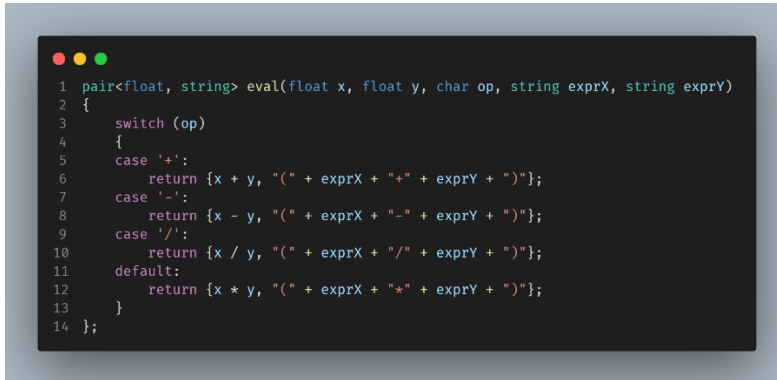
```

1  vector<pair<float, string>> solve(vector<int> arr)
2  {
3      vector<pair<float, string>> ans;
4      char ops[] = {'+', '-', '*', '/'};
5      int size = arr.size();
6      if (size == 1)
7      {
8          return {{(float)arr[0], to_string(arr[0])}};
9      }
10     if (size == 2)
11     {
12         for (int i = 0; i < 4; i++)
13         {
14             if (ops[i] == '/' && arr[1] == 0)
15             {
16                 continue;
17             }
18             ans.push_back(eval(((float)arr[0], (float)arr[1], ops[i], to_string(arr[0]), to_string(arr[1]))));
19         }
20     }
21     else
22     {
23         for (int i = 1; i < size; i++)
24         {
25             vector<int> left_arr(arr.begin(), arr.begin() + i);
26             vector<int> right_arr(arr.begin() + i, arr.end());
27
28             auto left = solve(left_arr);
29             auto right = solve(right_arr);
30
31             for (auto l : left)
32             {
33                 for (auto r : right)
34                 {
35                     for (int i = 0; i < 4; i++)
36                     {
37                         if (ops[i] == '/' && r.first == 0)
38                         {
39                             continue;
40                         }
41                         pair<float, string> temp = eval(l.first, r.first, ops[i], l.second, r.second);
42                         if (size == 4 && temp.first == 24)
43                         {
44                             ans.push_back(temp);
45                         }
46                         else if (arr.size() != 4)
47                         {
48                             ans.push_back(temp);
49                         }
50                     }
51                 }
52             }
53         }
54     }
55     return ans;
56 }

```

Gambar 10. Solve

Fungsi `eval(float x, float y, char op, string exprX, string exprY)` mengembalikan hasil evaluasi dari `x` dan `y`, serta *string* ekspresi matematikanya.



```
1 pair<float, string> eval(float x, float y, char op, string exprX, string exprY)
2 {
3     switch (op)
4     {
5         case '+':
6             return {x + y, "(" + exprX + "+" + exprY + ")"};
7         case '-':
8             return {x - y, "(" + exprX + "-" + exprY + ")"};
9         case '/':
10            return {x / y, "(" + exprX + "/" + exprY + ")"};
11        default:
12            return {x * y, "(" + exprX + "*" + exprY + ")"};
13    }
14 }
```

Gambar 11. eval

permutation.cpp

Fungsi `permute` mencari permutasi dengan pengulangan dari angka inputan secara rekursif.



```
1 void permute(vector<int> arr, int l, int r, set<vector<int>> &permutations)
2 {
3     if (l == r)
4     {
5         permutations.insert(arr);
6     }
7     else
8     {
9         for (int i = l; i <= r; i++)
10        {
11            swap(arr[l], arr[i]);
12            permute(arr, l + 1, r, permutations);
13            swap(arr[l], arr[i]);
14        }
15    }
16 }
17
```

Gambar 12. Permute

BAB 4 : EKSPERIMEN

Run program

```
.0000.      .o      .000000..o      .000000.      000000      000000      0000 00000000000000 0000000000.
.dPooY88b      .d88      d8P' `Y8 d8P' `Y8b `888'      `888.      .8' `888'      `8 `888 `Y88.
      ]8P'      .d'888      Y88bo.      888      888 888      `888.      .8'      888      888      .d88'
      .d8P'      .d' 888      ``Y8888o. 888      888 888      `888.      .8'      88800008      88800088P'
      .dP'      8800088800      ``Y88b 888      888 888      `888.8'      888      "      888`88b.
.oP      .o      888      oo      .d8P `88b d88' 888      o      `888'      888      o 888 `88b.
8888888888      o888o      88888888P' `Y8bood8P' o888000000d8      `8'      o888000000d8 o888o o888o

Welcome to 24 Card Game Solver!
24 Card Game is a game where the players are given 4 random cards and they have to
have to make math expression with those numbers so that it evaluates to 24.
Legal operators : +, -, *, /, ( )

Menu:
1. Randomize cards
2. Input cards
0. Exit

Option (1/2/0):
```

Gambar 13. Main

Opsi 1:

```
Option (1/2/0): 1
Generated cards: Q 7 9 7

No solution found
Execution time: 14.655 ms

Save solutions to file (Y/N)? Y
Successfully saved to "test" folder as Q_7_9_7.txt
```

Gambar 14. Opsi 1 - 1 : No solution found

```
Option (1/2/0): 1
Generated cards: 3 6 9 10

33 solutions found!
Solution #1: (3-9)*(6-10)
Solution #2: (3-(9-10))*6
Solution #3: ((3-9)+10)*6
Solution #4: (3+(10-9))*6
Solution #5: ((3+10)-9)*6
Solution #6: 6*(3-(9-10))
Solution #13: 6*((10+3)-9)
Solution #14: (6-10)*(3-9)
Solution #15: (6-(10/3))*9
Solution #16: 6*(10-(9-3))
Solution #17: 6*((10-9)+3)
Solution #18: (9-3)*(10-6)
Solution #19: ((9/3)*10)-6
```

Gambar 15. Opsi 1 - 2 Solution found (1)

```

Solution #20: (9*6)-(3*10)
Solution #21: 9*(6-(10/3))
Solution #22: (9*6)-(10*3)
Solution #23: (9*(10/3))-6
Solution #24: ((9*10)/3)-6
Solution #25: (10+(3-9))*6
Solution #26: (10/(3/9))-6
Solution #27: ((10+3)-9)*6
Solution #28: ((10/3)*9)-6
Solution #29: (10-6)*(9-3)
Solution #30: (10-(9-3))*6
Solution #31: (10*(9/3))-6
Solution #32: ((10-9)+3)*6
Solution #33: ((10*9)/3)-6
Execution time: 1.951 ms

Save solutions to file (Y/N)? N

```

Gambar 16. Opsi 1 - 2 Solution found (2)

```

Option (1/2/0): 1
Generated cards: 7 9 A Q

36 solutions found!
Solution #18: (9-7)*(12/1)
Solution #19: ((9-7)*12)*1
Solution #20: ((9-7)*12)/1
Solution #21: 12*(1*(9-7))
Solution #22: 12/(1/(9-7))
Solution #23: 12*((1*9)-7)
Solution #24: (12*1)*(9-7)
Solution #25: (12/1)*(9-7)
Solution #26: 12*(9-(1*7))
Solution #27: 12*((9*1)-7)
Solution #28: 12*((9/1)-7)
Solution #29: (12-9)*(1+7)
Solution #30: 12*(9-(7*1))
Solution #31: 12*(9-(7/1))
Solution #32: 12*((9-7)*1)
Solution #33: 12*((9-7)/1)
Solution #34: (12-9)*(7+1)
Solution #35: (12*(9-7))*1
Solution #36: (12*(9-7))/1
Execution time: 1.998 ms

```

Gambar 17. Opsi 1 - 3 Solution found

Opsi 2:

```
Option (1/2/0): 2

Input 4 cards (integers 2-9, A, J, K, Q) separated by space.
e. g. "A J 3 4".
Cards: 9 8 7 6

8 solutions found!
Solution #1: 8/((9-7)/6)
Solution #2: (8/(9-7))*6
Solution #3: 8*(6/(9-7))
Solution #4: (8*6)/(9-7)
Solution #5: 6/((9-7)/8)
Solution #6: (6/(9-7))*8
Solution #7: 6*(8/(9-7))
Solution #8: (6*8)/(9-7)
Execution time: 1.964 ms

Save solutions to file (Y/N)? Y
Successfully saved to "test" folder as 9_8_7_6.txt
```

Gambar 18. Opsi 2 - 1 Solution found

```
Option (1/2/0): 2

Input 4 cards (integers 2-9, A, J, K, Q) separated by space.
e. g. "A J 3 4".
Cards: A A A A
No solution found
Execution time: 0.000 ms

Save solutions to file (Y/N)? N
```

Gambar 19. Opsi 2 - 2 No solution found

```
Option (1/2/0): 2

Input 4 cards (integers 2-9, A, J, K, Q) separated by space.
e. g. "A J 3 4".
Cards: 6 6 6 6

7 solutions found!
Solution #1: 6+(6+(6+6))
Solution #2: 6+((6+6)+6)
Solution #3: (6+6)+(6+6)
Solution #4: (6*6)-(6+6)
Solution #5: (6+(6+6))+6
Solution #6: ((6+6)+6)+6
Solution #7: ((6*6)-6)-6
Execution time: 0.975 ms
```

Gambar 20. Opsi 2 - 3 Solution found

```
Option (1/2/0): 2
```

```
Input 4 cards (integers 2-9, A, J, K, Q) separated by space.
```

```
e. g. "A J 3 4".
```

```
Cards: A J 3 X
```

```
X is invalid.
```

```
Valid input is integer 2-9, A, Q, K, J
```

Gambar 21. Opsi 2 - 4 Invalid input

```
Input 4 cards (integers 2-9, A, J, K, Q) separated by space.
```

```
e. g. "A J 3 4".
```

```
Cards: 6 6 6 6 6 6
```

```
Please input exactly 4 cards separated by space.
```

Gambar 22. Opsi 2 - 5 Invalid input

Opsi 0 :

```
Option (1/2/0): 0
```

```
#####
```

```
Thanks for using 24 Solver
```

```
#####
```

Gambar 23. Opsi 0

LAMPIRAN

Github Repository

https://github.com/chaerla/Tucil1_13521044

Tabel Spesifikasi

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	✓	
2	Program berhasil running	✓	
3	Program dapat membaca input / generate sendiri dan memberi luaran	✓	
4	Solusi yang diberikan program memenuhi (berhasil mencapai 24)	✓	
5	Program dapat menyimpan solusi dalam file teks	✓	