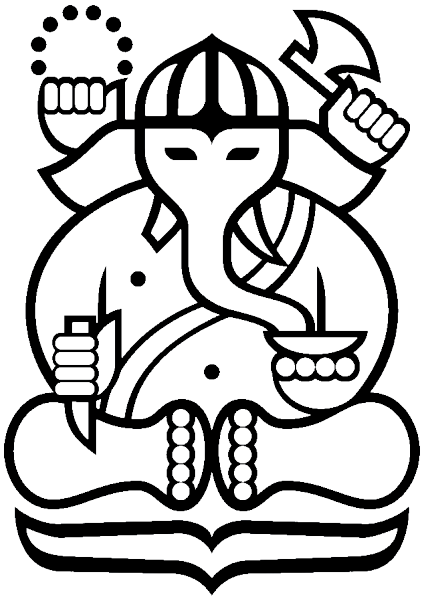
Laporan Tugas Kecil I

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah
IF2211 Strategi Algoritma



Disusun oleh:

I Putu Bakta Hari Sudewa 13521150

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

2023

Daftar Isi

1 Algoritma Brute Force

Algoritma *brute force* merupakan pendekatan yang lempeng (*straightforward*) untuk memecahkan suatu persoalan. Dalam program *24 Card Game Solver* ini algoritma *brute force* digunakan untuk memperoleh semua solusi yang mungkin. Berikut adalah langkah-langkah algoritma *brute force* yang telah disebutkan,

1. Terdapat 4 buah kartu. Dari 4 buah kartu tersebut, carilah semua permutasi yang mungkin, yaitu sebanyak $4! = 24$ buah kemungkinan.

2. Untuk setiap kemungkinan permutasi kartu, carilah semua kombinasi operator (+, -, *, /), yaitu sebanyak $4^3 = 64$ buah kemungkinan.

3. Berikutnya, tentukan kemungkinan banyaknya susunan kurung yang mungkin.

   (operand **operator** operand) **operator** (operand **operator** operand)
   (operand **operator** (operand **operator** operand)) **operator** operand
   operand **operator** ((operand **operator** operand) **operator** operand)
   ((operand **operator** operand) **operator** operand) **operator** operand
   operand **operator** (operand **operator** (operand **operator** operand))

   Terdapat 5 buah kemungkinan susunan kurung yang mungkin.

4. Dengan menggabungkan seluruh kemungkinan yang telah diperoleh pada langkah 1 hingga 3, diperoleh total $24 \times 64 \times 5 = 7680$ kemungkinan. Selanjutnya, untuk setiap kemungkinan ini akan dilakukan operasi aritmetika dan solusi yang diinginkan adalah yang menghasilkan 24.

Kompleksitas dari algoritma *brute force* ini adalah $O(n!m^{n-1})$, dengan n adalah banyaknya kartu dan m adalah banyaknya operator.

2 Source Program

Program dibuat dengan menggunakan bahasa pemrograman C++.

```cpp
#include <bits/stdc++.h>
#include <fstream>
using namespace std;

const int UNDEF = 99999;
const int MAX_RAND = 13;
const int MAX_CARD_AMT = 4;

vector<string> numbers(4);
vector<string> permutations;
vector<bool> chosen(4);
set<string> results;
```

Source Code 1: Deklarasi library, variabel, dan konstanta

```cpp
int main() {
  clock_t start, end;
  string choice;

  cout << "===\n";
  cout << "WELCOME TO 24 CARD GAME SOLVER!\n";
  cout << "===\n\n";

  while (choice != "1" && choice != "2") {
    cout << "Options:\n";
    cout << "1. Manual Input\n";
    cout << "2. Randomize Input\n";
    cout << "Your choice (1/2): ";
    getline(cin, choice);
    if (choice != "1" && choice != "2") {
      cout << "Invalid input. Please try again!\n";
    }
    cout << '\n';
  }

  if (choice == "1") {
    do {
      cout << "Available Cards: (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K)\n
   ";
      cout << "Card 1: ";
      getline(cin, numbers[0]);
      cout << "Card 2: ";
      getline(cin, numbers[1]);
      cout << "Card 3: ";
      getline(cin, numbers[2]);
      cout << "Card 4: ";
      getline(cin, numbers[3]);
    } while (!validate_input());
  } else {
    // Seeding with current time
    srand(time(0));
    for (int i = 0; i < 4; i++) {
      numbers[i] = convert_to_s_number_ar(to_string((rand() % MAX_RAND) +
   1));
      cout << "Card " << i + 1 << ": " << numbers[i] << '\n';
    }
  }
  cout << '\n';
```

```
42
43    start = clock();
44    search();
45    end = clock();
46
47    cout << "===\n\n";
48    cout << results.size() << " solution(s) found.";
49    print_results();
50    cout << "\n===\n";
51
52    while (results.size() > 0 && choice != "y" && choice != "Y" &&
53           choice != "n" && choice != "N") {
54      cout << "Save to file? (y/n): ";
55      getline(cin, choice);
56    }
57
58    if (choice == "y" || choice == "Y") {
59      string file_name;
60      do {
61        cout << "File name: ";
62        getline(cin, file_name);
63      } while (!validate_file_name(file_name));
64      write_to_file(file_name);
65      cout << "Result saved to output/" << file_name << ".txt\n";
66    }
67
68    double time_taken = double(end - start) / double(CLOCKS_PER_SEC);
69    cout << "\nTotal time: " << fixed << time_taken << setprecision(5)
70         << " sec.\n";
71
72    return 0;
73  }
```

Source Code 2: Fungsi *main*

Pada Source Code 2 dilakukan proses permintaan input, pemanggilan prosedur *search*, dan proses perhitungan total waktu eksekusi algoritma *brute force*.

```
1  void search() {
2    if (permutations.size() == MAX_CARD_AMT) {
3      vector<string> optr = {"*", "+", "/", "-"};
4
5      vector<vector<pair<pair<int, int>, pair<int, int>>>> brackets = {
6          // (a + b) + (c + d)
7          {{{0, 1}, {0, 4}}, {{2, 3}, {6, 10}}},
8          // ((a + b) + c) + d
9          {{{0, 1}, {1, 5}}, {{2, -7}, {0, 8}}, {{3, 3}, {10, 10}}},
10         // a + ((b + c) + d)
11         {{{1, 2}, {3, 7}}, {{3, -9}, {2, 10}}, {{0, 0}, {0, 0}}},
12         // (a + (b + c)) + d
13         {{{1, 2}, {3, 7}}, {{0, -1}, {0, 8}}, {{3, 3}, {10, 10}}},
14         // a + (b + (c + d))
15         {{{2, 3}, {5, 9}}, {{1, -3}, {2, 10}}, {{0, 0}, {0, 0}}},
16     };
17
18     int solutions_found = 0;
19     for (auto x : brackets) {
20       vector<string> s(11);
21       for (auto y : x) {
22         if (y.first.first == y.first.second) {
23           s[y.second.second] = permutations[y.first.first];
24           continue;
```

```
25          }
26          s[y.second.first] = "(";
27          s[y.second.second] = ")";
28          if (y.first.second < 0) {
29            s[-y.first.second] = permutations[y.first.first];
30            continue;
31          }
32          s[y.second.first + 1] = permutations[y.first.first];
33          s[y.second.second - 1] = permutations[y.first.second];
34        }
35
36        int cnt_numbers = 0, i = 0;
37        vector<int> optr_indexs;
38        for (auto y : s) {
39          if (y != "" && y != "(" && y != ")") {
40            cnt_numbers++;
41          } else if (y == "") {
42            optr_indexs.push_back(i);
43          }
44          i++;
45        }
46
47        for (int i = 0; i < 4; i++) {
48          for (int j = 0; j < 4; j++) {
49            for (int k = 0; k < 4; k++) {
50              s[optr_indexs[0]] = optr[i];
51              s[optr_indexs[1]] = optr[j];
52              s[optr_indexs[2]] = optr[k];
53              float res = calculate_operation(s);
54              if (cmpf(res)) {
55                results.insert(generate_string(s));
56                solutions_found++;
57              }
58            }
59          }
60        }
61      }
62    } else {
63      for (int i = 0; i < 4; i++) {
64        if (chosen[i])
65          continue;
66        chosen[i] = true;
67        permutations.push_back(convert_to_s_number_ar(numbers[i]));
68        search();
69        chosen[i] = false;
70        permutations.pop_back();
71      }
72    }
73  }
```

Source Code 3: Prosedur *search*

Pada Source Code 3 dilakukan proses pencarian solusi permainan kartu 24 dengan mencoba berbagai kombinasi yang mungkin antara operator dan operand. Terdapat sebuah variabel bernama *brackets* (line 5) yang menyimpan indeks posisi dari masing-masing operator dan operand.

```
1 bool cmpf(float a, float b = 24) { return fabs(a - b) < 0.000005f; }
```

Source Code 4: Fungsi *cmpf*

Fungsi *cmpf* pada Source Code 4 menghasilkan sebuah boolean bernilai true jika float a = float b, false jika sebaliknya.

```
float calculate_basic_arithmetic(float a, float b, string s) {
  if (s == "+") {
    return a + b;
  }
  if (s == "-") {
    return a - b;
  }
  if (s == "*") {
    return a * b;
  }
  if (s == "/" && !cmpf(b, 0)) {
    return a / (b + 0.0);
  }
  return UNDEF;
}
```

Source Code 5: Fungsi *calculate_basic_arithmetic*

Fungsi *calculate_basic_arithmetic* pada Source Code 5 menghasilkan sebuah float yang merupakan hasil operasi aritmetika antara float a dan float b, atau UNDEF jika terdapat pembagian oleh nol.

```
float stack_calculation(stack<float> &oprd, stack<string> &optr) {
  float second_number = oprd.top();
  oprd.pop();
  float first_number = oprd.top();
  oprd.pop();

  float result =
      calculate_basic_arithmetic(first_number, second_number, optr.top());
  if (result == UNDEF)
    return UNDEF;

  oprd.push(result);

  optr.pop();

  return result;
}
```

Source Code 6: Fungsi *stack_calculation*

Fungsi *stack_calculation* pada Source Code 6 menghasilkan sebuah float hasil operasi aritmetika dua buah float teratas pada stack *oprd*, atau menghasilkan UNDEF jika terdapat pembagian oleh nol.

```
float calculate_operation(vector<string> s) {
  stack<float> oprd;
  stack<string> optr;

  for (auto x : s) {
    if (x == "")
      continue;
    if (x != "(" && x != ")" && x != "*" && x != "/" && x != "-" && x != "
   +") {
      oprd.push(stoi(x));
    } else {
      if (optr.empty() || x == "*" || x == "/" || optr.top() == "(" ||
          x == "(") {
        optr.push(x);
      } else if (x == ")" || ((x == "-" || x == "+") && oprd.size() >= 2))
    {
        if (stack_calculation(oprd, optr) == UNDEF)
```

```
          return UNDEF;
        if (x == ")")
          optr.pop();
      }
    }
  }

  while (!optr.empty()) {
    if (stack_calculation(oprd, optr) == UNDEF)
      return UNDEF;
  }

  return oprd.top();
}
```

Source Code 7: Fungsi *calculate_operation*

Fungsi *calculate_operation* pada Source Code 7 menghasilkan sebuah float hasil dari operasi yang direpresentasikan oleh vector of string *s*. Jika operasi tidak valid akan menghasilkan UNDEF.

```
bool validate_input() {
  for (auto number : numbers) {
    bool is_valid = false;
    vector<string> valid_input = {"A", "2", "3",  "4", "5", "6", "7",
                                  "8", "9", "10", "J", "Q", "K"};
    for (auto input : valid_input) {
      if (input == number) {
        is_valid = true;
        break;
      }
    }
    if (!is_valid) {
      cout << "Invalid input. Please try again!\n\n";
      return false;
    }
  }
  return true;
}
```

Source Code 8: Fungsi *validate_input*

Fungsi *validate_input* pada Source Code 8 menghasilkan sebuah boolean bernilai true jika input kartu yang dimasukkan sesuai dengan ketentuan yang diberikan, false jika sebaliknya.

```
string convert_to_s_number_ar(string data) {
  if (data == "A")
    data = "1";
  else if (data == "1")
    data = "A";
  else if (data == "J")
    data = "11";
  else if (data == "11")
    data = "J";
  else if (data == "Q")
    data = "12";
  else if (data == "12")
    data = "Q";
  else if (data == "K")
    data = "13";
  else if (data == "13")
    data = "K";
  return data;
```

```
}
```

Source Code 9: Fungsi *convert_to_s_number_ar*

Fungsi *convert_to_s_number_ar* pada Source Code 9 menghasilkan sebuah string hasil konversi dari simbol kartu menjadi simbol angka dan sebaliknya.

```
string generate_string(vector<string> data) {
  string result = "";
  for (auto s : data) {
    result += convert_to_s_number_ar(s) + " ";
  }
  return result;
}
```

Source Code 10: Fungsi *generate_string*

Fungsi *generate_string* pada Source Code 10 menghasilkan sebuah string dari parameter vector of string *data*.

```
void write_to_file(string file_name) {
  ofstream new_file("output/" + file_name + ".txt");
  new_file << "Cards: ";
  for (auto card : numbers) {
    new_file << card << ' ';
  }
  new_file << '\n' << results.size() << " solution(s) found.\n";
  for (auto result : results) {
    new_file << result << '\n';
  }
  new_file.close();
}
```

Source Code 11: Prosedur *write_to_file*

Prosedur *write_to_file* pada Source Code 11 melakukan proses penulisan file dengan nama file sesuai paramater *file_name* yang diberikan.

```
bool validate_file_name(string file_name) {
  if (file_name.size() > 100) {
    cout << "File name maximum 100 characters!\n";
    return false;
  }
  return true;
}
```

Source Code 12: Fungsi *validate_file_name*

Fungsi *validate_file_name* pada Source Code 12 menghasilkan sebuah boolean bernilai true jika *file_name* yang dimasukkan oleh user tidak melebihi 100 karakter, false jika sebaliknya.

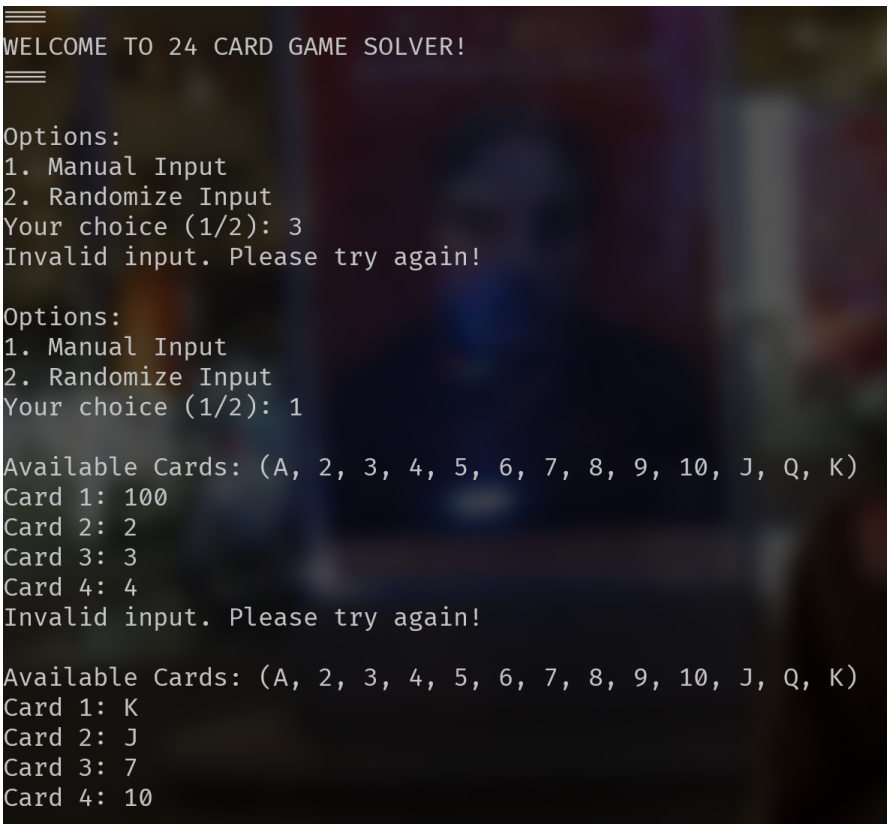
```
void print_results() {
  cout << '\n';
  for (auto result : results) {
    cout << result << '\n';
  }
}
```

Source Code 13: Prosedur *print_results*

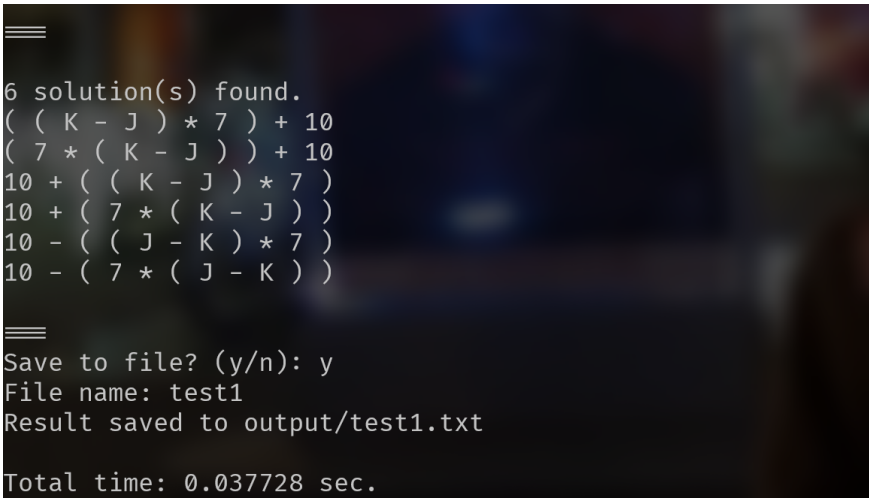
Prosedur *print_results* pada Source Code 13 mencetak semua solusi yang ditemukan ke layar.

3 Input dan Output

3.1 Test Case 1

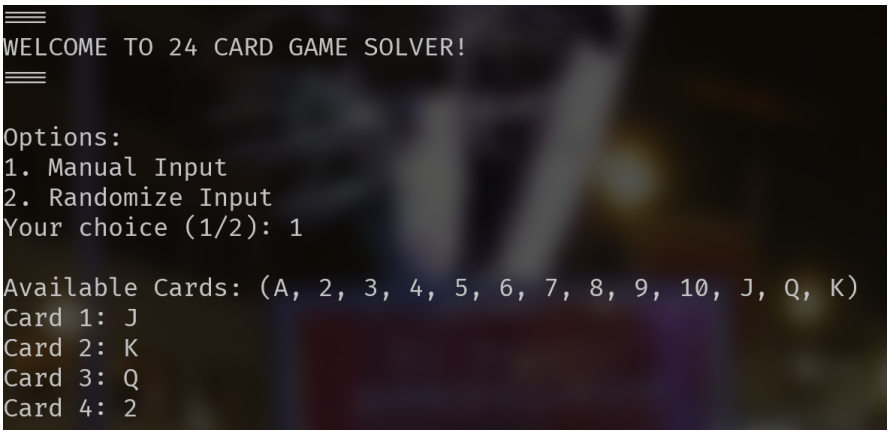


Gambar 1: Test Case 1 - Input manual, validasi input pilihan, dan validasi input kartu

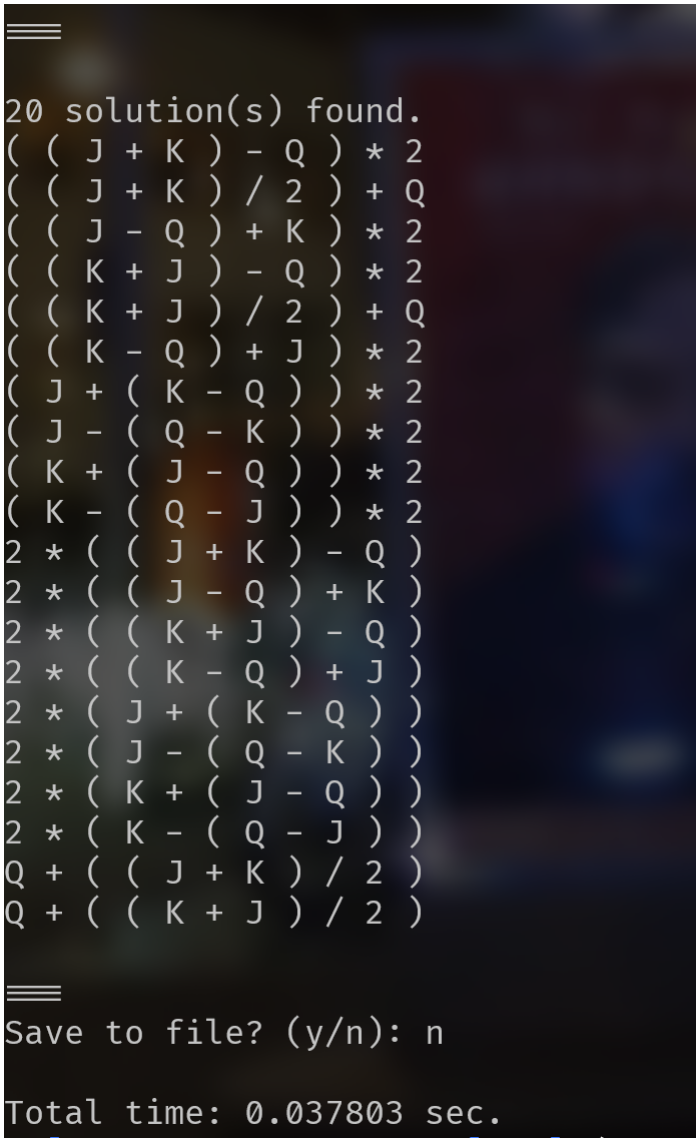


Gambar 2: Test Case 1 - Output, simpan ke file, dan total waktu eksekusi

3.2 Test Case 2

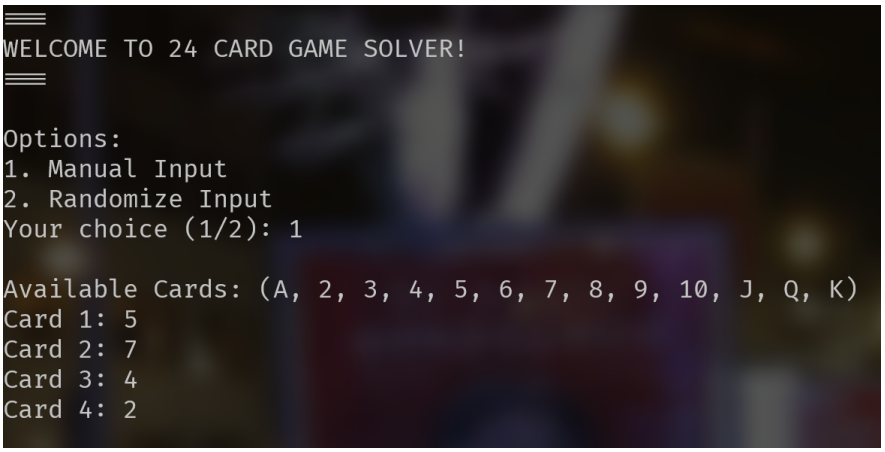


Gambar 3: Test Case 2 - Input manual

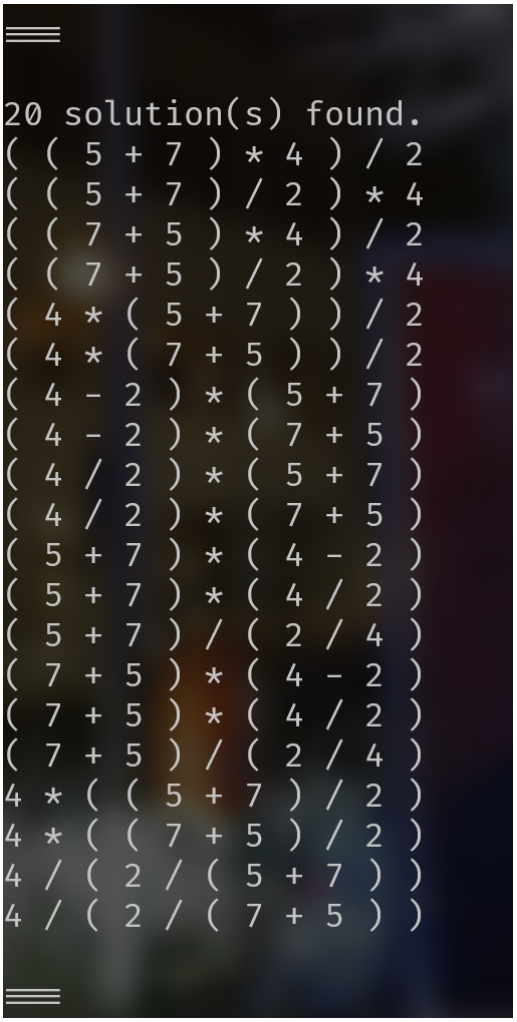


Gambar 4: Test Case 2 - Output dan total waktu eksekusi

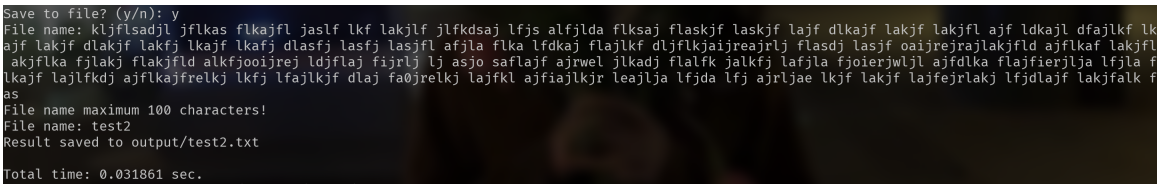
3.3 Test Case 3



Gambar 5: Test Case 3 - Input manual

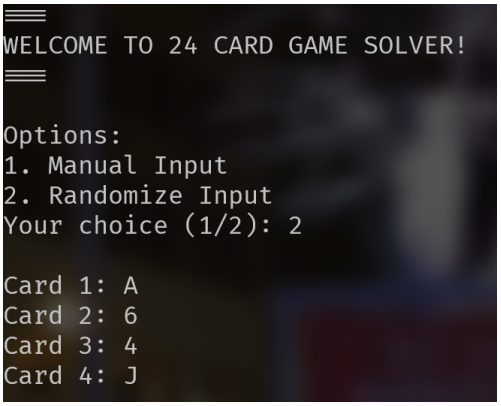


Gambar 6: Test Case 3 - Output

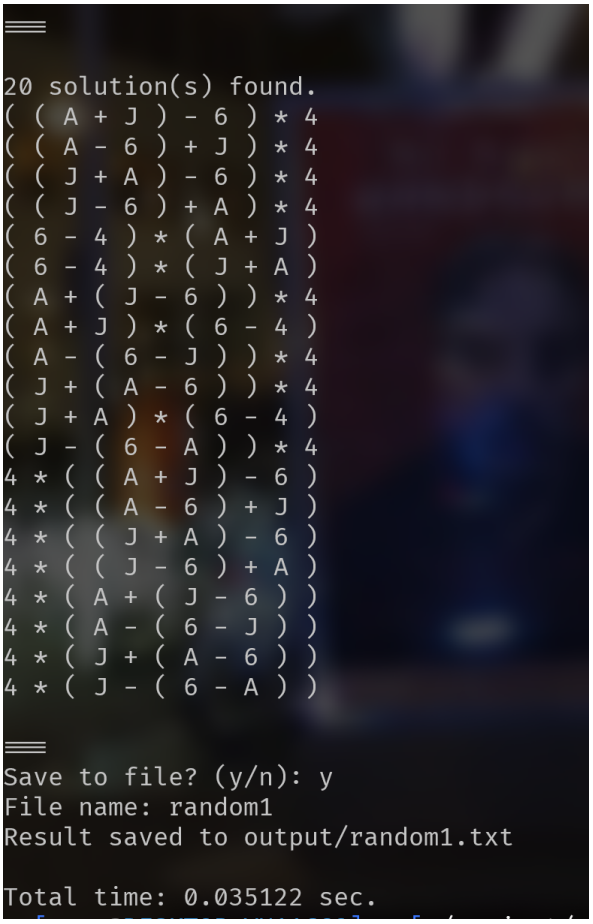


Gambar 7: Test Case 3 - Validasi nama file dan total waktu eksekusi

3.4 Test Case 4

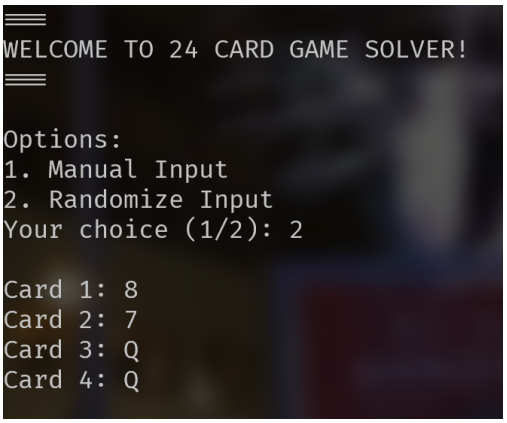


Gambar 8: Test Case 4 - Input random

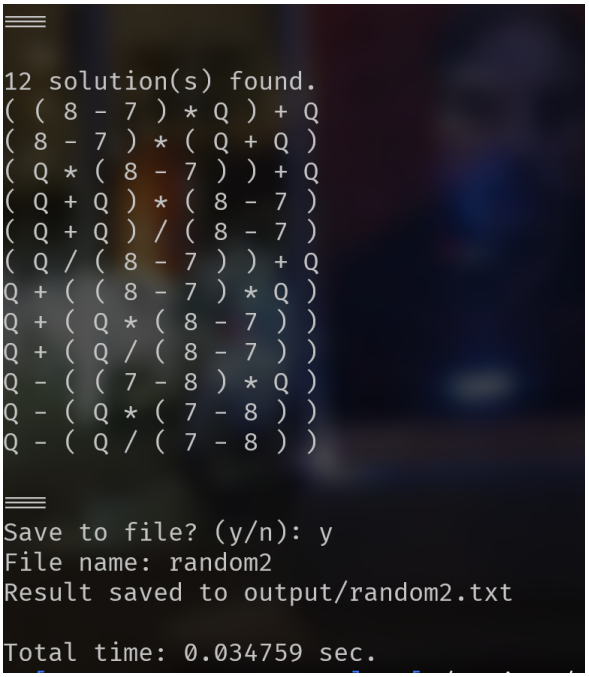


Gambar 9: Test Case 4 - Ouput dan total waktu eksekusi

3.5 Test Case 5

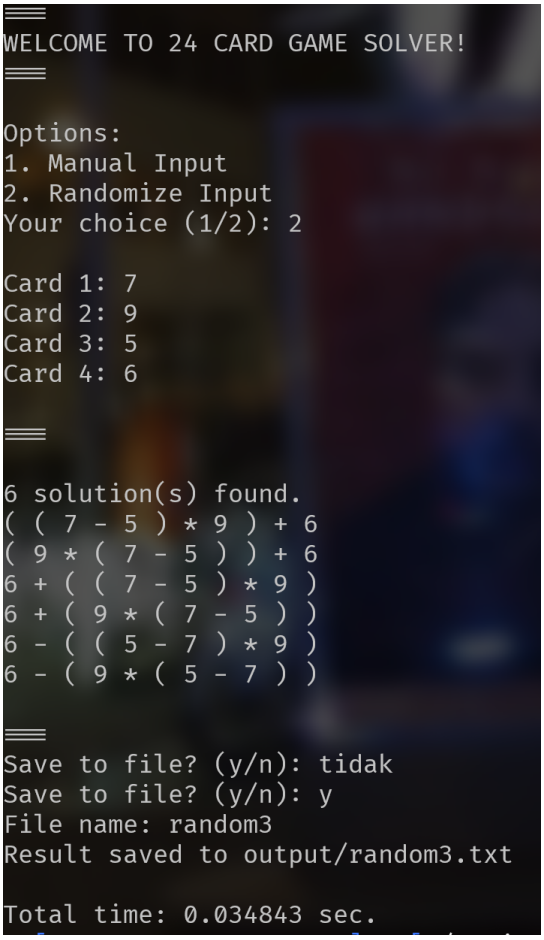


Gambar 10: Test Case 5 - Input random



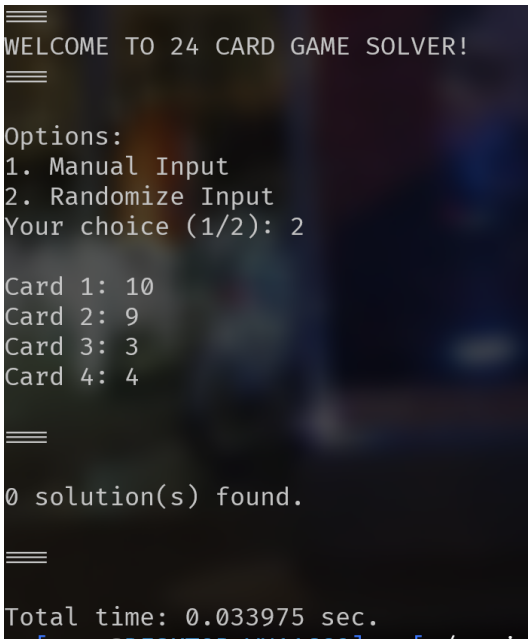
Gambar 11: Test Case 5 - Output dan total waktu eksekusi

3.6 Test Case 6



Gambar 12: Test Case 6 - Input random, output, validasi pilihan *save to file* dan total waktu eksekusi

3.7 Test Case 7



Gambar 13: Test Case 7 - Input random, output tanpa solusi, dan total waktu eksekusi

4 Repositori

https://github.com/sozyGithub/Tucil1_13521150

5 Checklist

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

6 Referensi

- R. Munir, (2023). *Algoritma Brute Force (bagian 1)*[Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf