Laporan Tugas Kecil 1

IF2211 - Strategi Algoritma

Semester II Tahun Ajaran 2022/2023

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Disusun oleh:

Muchammad Dimas Sakti Widyatmaja

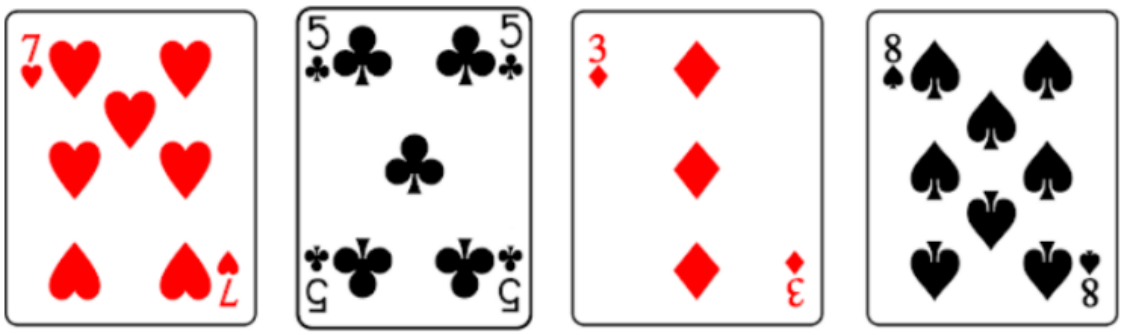
13521160

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 4013

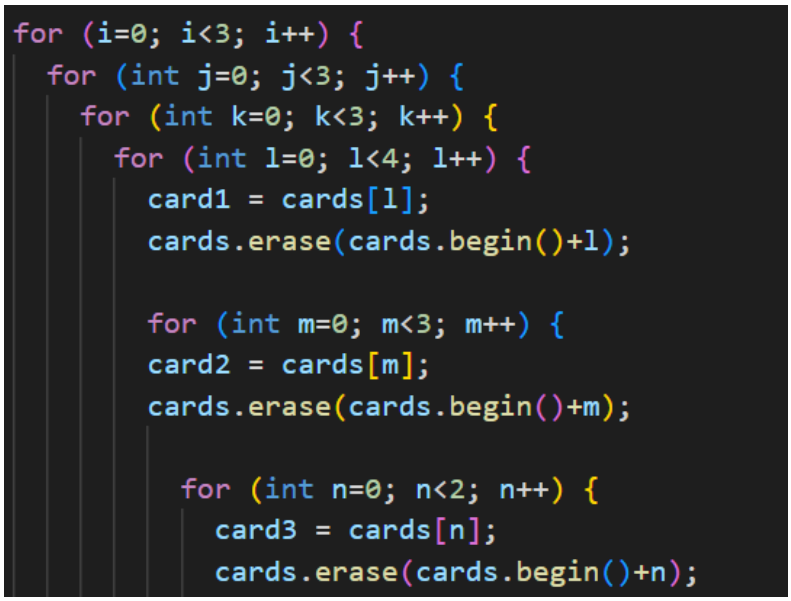
A. Algoritma Brute Force Permainan kartu 24

Algoritma dalam penyelesaian permainan kartu 24 ini adalah brute force. Algoritma brute force merupakan algoritma yang menggunakan pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma brute force merupakan algoritma yang relative sederhana. Hampir semua persoalan dapat diselesaikan dengan algoritma brute force.



MAKE IT 24

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



```
for (i=0; i<3; i++) {
  for (int j=0; j<3; j++) {
    for (int k=0; k<3; k++) {
      for (int l=0; l<4; l++) {
        card1 = cards[l];
        cards.erase(cards.begin()+l);

        for (int m=0; m<3; m++) {
        card2 = cards[m];
        cards.erase(cards.begin()+m);

          for (int n=0; n<2; n++) {
            card3 = cards[n];
            cards.erase(cards.begin()+n);
```

Penerapan algoritma brute force dalam permainan kartu 24 ini dapat dilihat pada penerapan *loop* yang memiliki banyak sarang dikarenakan pendekatan yang lempang. Jumlah tingkat dalam *loop* bersarang yang terdapat dalam program saya adalah 6 tingkat. 3 *loop* pertama digunakan untuk mengiterasi terhadap segala kemungkinan susunan operator yang ada. 3 *loop* terakhir digunakan untuk melakukan iterasi terhadap semua kemungkinan dari kartu yang tersedia. Pada loop terakhir juga, terdapat iterasi secara manual terhadap semua kemungkinan ekspresi berdasarkan penempatan kurung seperti (c0 op0 c1) op2 (c2 op3 c3), ((c0 op0 c1) op2 c2) op3 c3), (c0 op0 (c1 op2 c2)) op3 c3), dan lain sebagainya.

Dari semua ekspresi tersebut akan diubah menjadi integer dan dihitung. Apabila hasilnya berjumlah 24 maka ekspresi tersebut disimpan dalam vector yang berisi solusi-solusi permainan. Di akhir program, solusi-solusi ini akan dicetak dan dapat disimpan ke file eksternal.

B. Source Code

```cpp
#include <iostream>
#include <cstdlib>
#include <vector>
#include<fstream>
#include <chrono>
using namespace std;
using namespace std::chrono;

// Buat komentar (v)
// Input random (v)
// Input handling ()
// Waktu (v)
// Simpan ke file (v)

void checkExist(string expression, vector<string>& solutions) {
  // Prosedur untuk mengecek apakah solusi sudah pernah muncul sebelumnya
  // Jika sudah pernah muncul maka solusi tidak akan dianggap

  for (int i=0; i<solutions.size(); i++) {
    if (solutions[i] == expression) {
      return;
    }
  }

  solutions.push_back(expression);
  return;
}
```

```cpp
void checkSolution(int num, string expression,vector<string>& solutions) {
  // Prosedur untuk mengetahui apakah hasil kombinasi 4 kartu merupakan solusi atau bukan
  // Dan menentukan apakah solusi sudah pernah muncul atau belum

  if (num == 24)
    checkExist(expression, solutions);
}
```

```
void printSolution(vector<string> solutions) {
  // Prosedur untuk mencetak solusi ke layar

  if (solutions.size() == 0) {
    cout << "Tidak ada solusi\n";
  } else {
    cout << "Jumlah solusi: " << solutions.size() << "\n";
    for (int i=0; i < solutions.size(); i++) {
      cout << solutions[i] << "\n";
    }
  }
}

int operation (int x, int y, char opr) {
  // Fungsi untuk melakukan operasi aritmetika berdasarkan dua input angka dan satu karakter sebagai operator

  if (opr == '+') {
    return  x+y;
  } else if (opr == '-') {
    return  x-y;
  } else if (opr == '/') {
    return  x/y;
  } else if (opr == '*') {
    return  x*y;
  } else return 0;
}
```

```
string convertToString(char* buffer) {
  // Fungsi untuk mengonversi array of character ke string

  string str = buffer;
  return str;
}

void saveToFile(vector<string> solutions) {
  // Prosedur untuk menyimpan solusi ke dalam file berdasarkan masukan pengguna

  // Kamus lokal
  string filename;

  // Algoritma
  cout << "Masukkan nama file output: ";
  cin >> filename;
  ofstream fout("../test/" + filename + ".txt");

  fout << "Jumlah solusi: " << solutions.size() << "\n";
  for (int i=0; i < solutions.size(); i++) {
    fout << solutions[i] << "\n";
  }

  fout.close();
  cout << "Output berhasil disimpan\n";
}
```

```
void charSplit(string input,  vector<int>& cards, bool &isRight) {
  int count = 0;

  for (int i=0; i < input.size(); i++) {
    if (input[i] == '2' || input[i] == '3' || input[i] == '4' || input[i] == '5' || input[i] == '6' ||
    input[i] == '7' || input[i] == '8' || input[i] == '9') {
      cards.push_back((int)input[i]-48);
      count++;

    } else if (input[i] == '1') {
      if (input[i+1] == '0') {
        cards.push_back(10);
        count++;
        i++;
      } else if (input[i+1] == ' ') {
        cout << "Untuk angka satu, masukkan huruf A!\n";
        cards.clear();
        return;
      } else {
        cout << "Masukkan angka dalam rentang 2-9\n";
      }
    } else if (input[i] == 'A') {
      cards.push_back(1);
      count++;
    } else if (input[i] == 'J') {
      cards.push_back(11);
      count++;
    } else if (input[i] == 'Q') {
      cards.push_back(12);
      count++;
```

```
    } else if (input[i] == 'K') {
      cards.push_back(13);
      count++;
    } else if (input[i] == ' ') {
    } else {
      cout << "Karakter masukan tidak benar, masukkan input dengan benar!\n";
      cards.clear();
      return;
    }

    if (count > 4) {
      cout << "Masukan terlalu banyak, masukkan 4 kartu\n";
      cards.clear();
      return;
    }
  }

  if (count < 4) {
    cout << "Masukan terlalu sedikit, masukkan 4 kartu\n";
    cards.clear();
    return;
  } else {
    isRight = true;
    cout << "Kartu anda adalah: ";
    for (int i=0; i<4; i++) {
      cout << cards[i] << " ";
    }
    return;
  }
}
```

```cpp
int main() {

  // Kamus Lokal
  bool isRight = false;
  char inputChar;
  int input, card1, card2, card3, i, num;
  string expression;
  string cardString;
  char oprs[4] = {'+', '-', '*', '/'};
  char buffer[50];
  vector<int> cards;
  vector<string> solutions;

  // Input
  cout << "Pilih jenis input: (Ketikkan angkanya saja)\n";
  while (!isRight) {
    cout << "1. Manual\n2. Acak\n";
    cin >> input;

    if (input > 0 && input < 3) isRight = true;
    else cout << "Masukan salah, tolong masukkan angka 1 atau 2!\n";
  }

  isRight = false;
  if (input == 1) {
    // Input dari pengguna
    while (!isRight) {
      cout << "Masukkan 4 kartu: \n";
      cout << "<card 1> <card 2> <card 3> <card 4>\n";
```

```cpp
    // Input dari pengguna
    while (!isRight) {
      cout << "Masukkan 4 kartu: \n";
      cout << "<card 1> <card 2> <card 3> <card 4>\n";
      getline(cin >> ws, cardString);

      charSplit(cardString, cards, isRight);
      // isRight = true;
      // cout << cardString << "\n";
    }
  } else {
    // Input acak
    srand(time(0));

    cout << "Kartu anda adalah: ";
    for (i=0; i<4; i++) {
      card1 = rand() % 13 + 1;
      cards.push_back(card1);
      if (card1 == 1) {
        cout << "A ";
      } else if (card1 == 11) {
        cout << "J ";
      } else if (card1 == 12) {
        cout << "Q ";
      } else if (card1 == 13) {
        cout << "K ";
      } else {
        cout << card1 << " ";
      }
    }
```

```
auto stop = high_resolution_clock::now();
// Menghentikan perhitungan waktu algoritma bruteforce

}
    cout << "\n";
```

```
// Memulai perhitungan waktu algoritma bruteforce
auto start = high_resolution_clock::now();

for (i=0; i<3; i++) {
  for (int j=0; j<3; j++) {
    for (int k=0; k<3; k++) {
      for (int l=0; l<4; l++) {
        card1 = cards[l];
        cards.erase(cards.begin()+l);

        for (int m=0; m<3; m++) {
        card2 = cards[m];
        cards.erase(cards.begin()+m);

          for (int n=0; n<2; n++) {
            card3 = cards[n];
            cards.erase(cards.begin()+n);

            // ((c0 op0 c1) op1 c2) op2 c3
            num = operation((operation(operation(card1, card2, oprs[i]), card3, oprs[j])), cards[0], oprs[k]);
            sprintf(buffer, "(( %d %c %d ) %c %d) %c %d", card1, oprs[i], card2, oprs[j], card3, oprs[k], cards[0]);
            expression = convertToString(buffer);
            checkSolution(num, expression, solutions);

            // (c0 op0 (c1 op1 c2)) op2 c3
            num = operation((operation(card1, operation(card2, card3, oprs[i]), oprs[j])), cards[0], oprs[k]);
            sprintf(buffer, "( %d %c ( %d %c %d )) %c %d", card1, oprs[i], card2, oprs[j], card3, oprs[k], cards[0]);
            expression = convertToString(buffer);
            checkSolution(num, expression, solutions);
```

```
            // c0 op0 ((c1 op1 c2) op2 c3)
            num = operation(card1, oprs[i], (operation(operation(card2, card3, oprs[j]), cards[0], oprs[k])));
            sprintf(buffer, "%d %c (( %d %c %d ) %c %d )", card1, oprs[i], card2, oprs[j], card3, oprs[k], cards[0]);
            expression = convertToString(buffer);
            checkSolution(num, expression, solutions);

            // c0 op0 (c1 op1 (c2 op2 c3))
            num = operation(card1, oprs[i], (operation(card2, operation(card3, cards[0], oprs[j]), oprs[k])));
            sprintf(buffer, "%d %c ( %d %c ( %d %c %d ))", card1, oprs[i], card2, oprs[j], card3, oprs[k], cards[0]);
            expression = convertToString(buffer);
            checkSolution(num, expression, solutions);

            // ((c0 op0 c1) op1 (c2 op2 c3))
            num = operation((operation(card1, card2, oprs[i])), (operation(card3, cards[0], oprs[j])), oprs[j]);
            sprintf(buffer, "( %d %c %d ) %c ( %d %c %d )", card1, oprs[i], card2, oprs[j], card3, oprs[k], cards[0])
            expression = convertToString(buffer);
            checkSolution(num, expression, solutions);

            cards.insert(cards.begin()+n, card3);
          }
          cards.insert(cards.begin()+m, card2);
        }
        cards.insert(cards.begin()+l, card1);
      }
    }
  }
}
```

```
    // Mencatat waktu algoritma bruteforce selesai
    auto stop = high_resolution_clock::now();

    // Menghitung durasi yang diperlukan dalam algoritma bruteforce;
    auto duration = duration_cast<microseconds>(stop - start);

    printSolution(solutions);
    cout << "Waktu eksekusi bruteforce: "
         << duration.count() << " microseconds" << endl;

    // Output ke file
    cout << "Apakah solusi ingin disimpan dalam file? (y/n)\n";
    while (true) {
      cin >> inputChar;
      if (inputChar == 'y') {
        saveToFile(solutions);
        return 0;
      } else if (inputChar == 'n') {
        return 0;
      }

      cout << "Masukan salah, masukkan huruf y atau n\n";
    }

}
```

C. Hasil

i. Input manual dan tidak disimpan

```
Pilih jenis input: (Ketikkan angkanya saja)
1. Manual
2. Acak
1
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
2 3 4 5
Kartu anda adalah: 2 3 4 5 Jumlah solusi: 20
(( 3 + 4 ) + 5) * 2
( 3 + ( 4 + 5 )) * 2
(( 3 + 5 ) + 4) * 2
( 3 + ( 5 + 4 )) * 2
(( 4 + 3 ) + 5) * 2
( 4 + ( 3 + 5 )) * 2
(( 4 + 5 ) + 3) * 2
( 4 + ( 5 + 3 )) * 2
(( 5 + 3 ) + 4) * 2
( 5 + ( 3 + 4 )) * 2
(( 5 + 4 ) + 3) * 2
( 5 + ( 4 + 3 )) * 2
(( 3 + 5 ) - 2) * 4
(( 5 + 3 ) - 2) * 4
(( 3 - 2 ) + 5) * 4
( 3 - ( 5 + 2 )) * 4
(( 5 - 2 ) + 3) * 4
( 5 - ( 3 + 2 )) * 4
( 3 - ( 2 - 5 )) * 4
( 5 - ( 2 - 3 )) * 4
Waktu eksekusi bruteforce: 5000 microseconds
Apakah solusi ingin disimpan dalam file? (y/n)
n
```

ii. Input manual dan disimpan

```
Pilih jenis input: (Ketikkan angkanya saja)
1. Manual
2. Acak
1
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
9 4 A K
Kartu anda adalah: 9 4 1 13 Jumlah solusi: 12
(( 9 * 4 ) + 1) - 13
(( 4 * 9 ) + 1) - 13
( 1 * ( 9 + 4 )) - 13
( 1 * ( 4 + 9 )) - 13
(( 9 * 4 ) - 13) + 1
( 9 * 4 ) - ( 13 + 1 )
(( 4 * 9 ) - 13) + 1
( 4 * 9 ) - ( 13 + 1 )
( 9 * 4 ) - ( 13 - 1 )
( 4 * 9 ) - ( 13 - 1 )
( 9 * 4 ) - ( 13 * 1 )
( 4 * 9 ) - ( 13 * 1 )
Waktu eksekusi bruteforce: 4946 microseconds
Apakah solusi ingin disimpan dalam file? (y/n)
y
Masukkan nama file output: Success
Output berhasil disimpan
```

test > Success.txt

```
Jumlah solusi: 12
(( 9 * 4 ) + 1) - 13
(( 4 * 9 ) + 1) - 13
( 1 * ( 9 + 4 )) - 13
( 1 * ( 4 + 9 )) - 13
(( 9 * 4 ) - 13) + 1
( 9 * 4 ) - ( 13 + 1 )
(( 4 * 9 ) - 13) + 1
( 4 * 9 ) - ( 13 + 1 )
( 9 * 4 ) - ( 13 - 1 )
( 4 * 9 ) - ( 13 - 1 )
( 9 * 4 ) - ( 13 * 1 )
( 4 * 9 ) - ( 13 * 1 )

```

iii. Input acak dan tidak disimpan

```
Pilih jenis input: (Ketikkan angkanya saja)
1. Manual
2. Acak
2
Kartu anda adalah: 3 Q 10 J
Jumlah solusi: 8
(( 3 + 10 ) - 11) * 12
(( 10 + 3 ) - 11) * 12
( 3 - ( 10 + 11 )) * 12
(( 3 - 11 ) + 10) * 12
( 10 - ( 3 + 11 )) * 12
(( 10 - 11 ) + 3) * 12
( 3 - ( 11 - 10 )) * 12
( 10 - ( 11 - 3 )) * 12
Waktu eksekusi bruteforce: 6039 microseconds
Apakah solusi ingin disimpan dalam file? (y/n)
n
```

iv. Input acak disimpan

```
Pilih jenis input: (Ketikkan angkanya saja)
1. Manual
2. Acak
2
Kartu anda adalah: 9 2 10 Q
Jumlah solusi: 10
( 10 - 9 ) * ( 2 + 12 )
( 10 - 9 ) * ( 12 + 2 )
( 10 - 9 ) * ( 2 - 12 )
( 10 - 9 ) * ( 12 - 2 )
( 2 - ( 10 * 9 )) * 12
(( 10 - 9 ) * 2) * 12
( 10 - 9 ) * ( 2 * 12 )
(( 10 - 9 ) * 12) * 2
( 12 - ( 10 * 9 )) * 2
Waktu eksekusi bruteforce: 2997 microseconds
Apakah solusi ingin disimpan dalam file? (y/n)
y
Masukkan nama file output: Berhasil
Output berhasil disimpan
```

test > Berhasil.txt

```
1   Jumlah solusi: 10
2   ( 10 - 9 ) * ( 2 + 12 )
3   ( 10 - 9 ) * ( 12 + 2 )
4   ( 10 - 9 ) * ( 2 - 12 )
5   ( 10 - 9 ) * ( 12 - 2 )
6   ( 2 - ( 10 * 9 )) * 12
7   (( 10 - 9 ) * 2) * 12
8   ( 10 - 9 ) * ( 2 * 12 )
9   (( 10 - 9 ) * 12) * 2
10  ( 10 - 9 ) * ( 12 * 2 )
11  ( 12 - ( 10 * 9 )) * 2
12
```

v. Kasus tidak ada solusi

```
Pilih jenis input: (Ketikkan angkanya saja)
1. Manual
2. Acak
1
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
K K K K
Kartu anda adalah: 13 13 13 13 Tidak ada solusi
Waktu eksekusi bruteforce: 3990 microseconds
Apakah solusi ingin disimpan dalam file? (y/n)
n
```

vi. Input salah

```
Pilih jenis input: (Ketikkan angkanya saja)
1. Manual
2. Acak
1
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
2 3 4
Masukan terlalu sedikit, masukkan 4 kartu
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
2 3 4 5 6
Masukan terlalu banyak, masukkan 4 kartu
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
h 3 4 5
Karakter masukan tidak benar, masukkan input dengan benar!
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
1 2 3 4
Untuk angka satu, masukkan huruf A!
Masukkan 4 kartu:
<card 1> <card 2> <card 3> <card 4>
A A A A
Kartu anda adalah: 1 1 1 1 Tidak ada solusi
Waktu eksekusi bruteforce: 4988 microseconds
Apakah solusi ingin disimpan dalam file? (y/n)
n
```

D. Lampiran

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	✓	

Tautan repositori: