Laporan Tugas 1 Bahasa dan Otomata Kelas KOM A

Andreandhiki R.P (23/517511/PA/22191) - Andrian Danar P. (23/513040/PA/21917)

See More on Github

Deskripsi Permasalahan

Tugas ini meminta implementasi sebuah Deterministic Finite Automaton (DFA) yang mensimulasikan operasi sebuah vending machine penjual minuman kaleng. Vending machine ini memiliki tiga jenis minuman dengan harga yang berbeda: Minuman A (Rp3.000), Minuman B (Rp4.000), dan Minuman C (Rp6.000). Mesin hanya menerima uang dalam pecahan tertentu, yaitu Rp1.000, Rp2.000, Rp5.000, dan Rp10.000, dengan batas maksimal nominal uang yang dapat dimasukkan sebesar Rp10.000.

Permasalahan utama yang perlu diselesaikan meliputi:

- Pemrosesan Uang:
  - Mesin harus dapat memproses input uang yang dimasukkan oleh pengguna dan mengubah state sesuai dengan transisi DFA yang telah ditentukan.
  - Jika jumlah uang yang dimasukkan tepat sesuai harga salah satu minuman, tombol ON untuk minuman tersebut harus menyala, menandakan bahwa minuman bisa dibeli.
  - Jika uang yang dimasukkan kurang, mesin harus menolak pembelian.
- Fitur Penukaran Uang:
  - Mesin harus mampu menukar uang pecahan besar (Rp2.000, Rp5.000, atau Rp10.000) menjadi pecahan Rp1.000 jika diperlukan.
- Fitur Kembalian (Bonus):
  - Jika uang yang dimasukkan lebih dari harga minuman, mesin harus dapat menghitung dan mengembalikan uang kembalian kepada pengguna.
- Validasi Input:
  - Program harus memastikan bahwa input yang dimasukkan oleh pengguna valid, yaitu berupa pecahan uang yang diterima atau pilihan minuman (A, B, C).
- Representasi DFA:
  - DFA harus direpresentasikan dalam bentuk file konfigurasi (.txt) yang mencakup states, alphabet, start state, accepting states, dan transisi.
  - Program harus membaca file konfigurasi ini dan mensimulasikan operasi vending machine berdasarkan DFA yang telah didefinisikan.
- Output yang Jelas:
  - Program harus menampilkan lintasan state DFA selama proses transaksi.
  - Output harus mencerminkan status transaksi (ACCEPTED atau REJECTED) beserta alasan jika ditolak (uang kurang atau lebih).
  - Untuk fitur bonus, output harus mencantumkan jumlah uang kembalian yang diberikan.

Permasalahan ini mengharuskan pemahaman mendalam tentang konsep DFA, termasuk bagaimana state, transisi, dan input alphabet bekerja, serta kemampuan untuk mengimplementasikannya dalam bentuk program yang interaktif dan user-friendly.

Notasi DFA
(States, Alphabet, Start, State, Accepting States, Transition table, Diagram)

States:

S0, S1000, S2000, S3000, S4000, S5000, S6000, S7000, S8000, S9000, S10000, Reject, DispenseA_0, DispenseA_1000, DispenseA_2000, DispenseA_3000, DispenseA_4000, DispenseA_5000, DispenseA_6000, DispenseA_7000, DispenseB_0, DispenseB_1000, DispenseB_2000, DispenseB_3000, DispenseB_4000, DispenseB_5000, DispenseB_6000, DispenseC_0, DispenseC_1000, DispenseC_2000, DispenseC_3000, DispenseC_4000

Alphabet:

1000, 2000, 5000, 10000, A, B, C

Start State:

S0

Accept States:

DispenseA_0, DispenseA_1000, DispenseA_2000, DispenseA_3000, DispenseA_4000, DispenseA_5000, DispenseA_6000, DispenseA_7000, DispenseB_0, DispenseB_1000, DispenseB_2000, DispenseB_3000, DispenseB_4000, DispenseB_5000, DispenseB_6000, DispenseC_0, DispenseC_1000, DispenseC_2000, DispenseC_3000, DispenseC_4000

Transition Table

Current State	Input	Next State
S0	1000	S1000
S0	2000	S2000
S0	5000	S5000
S0	10000	S10000
S0	A/B/C	Reject
S1000	1000	S2000
S1000	2000	S3000

S1000	5000	S6000
S1000	10000	Reject
S1000	A/B/C	Reject
S2000	1000	S3000
S2000	2000	S4000
S2000	5000	S7000
S2000	10000	Reject
S2000	A/B/C	Reject
S3000	1000	S4000
S3000	2000	S5000
S3000	5000	S8000
S3000	10000	Reject
S3000	A	DispenseA_0
S3000	B/C	Reject
S4000	1000	S5000
S4000	2000	S6000
S4000	5000	S9000
S4000	10000	Reject
S4000	A	DispenseA_1000
S4000	B	DispenseB_0
S4000	C	Reject
S5000	1000	S6000
S5000	2000	S7000

S5000	5000	S10000
S5000	10000	Reject
S5000	A	DispenseA_2000
S5000	B	DispenseB_1000
S5000	C	Reject
S6000	1000	S7000
S6000	2000	S8000
S6000	5000/10000	Reject
S6000	A	DispenseA_3000
S6000	B	DispenseB_2000
S6000	C	DispenseC_0
S7000	1000	S8000
S7000	2000	S9000
S7000	5000/10000	Reject
S7000	A	DispenseA_4000
S7000	B	DispenseB_3000
S7000	C	DispenseC_1000
S8000	1000	S9000
S8000	2000	S10000
S8000	5000/10000	Reject
S8000	A	DispenseA_5000
S8000	B	DispenseB_4000
S8000	C	DispenseC_2000

S9000	1000	S10000
S9000	2000/5000/10000	Reject
S9000	A	DispenseA_6000
S9000	B	DispenseB_5000
S9000	C	DispenseC_3000
S10000	1000/2000/5000/10000	Reject
S10000	A	DispenseA_7000
S10000	B	DispenseB_6000
S10000	C	DispenseC_4000
Reject	1000/2000/5000/10000/A/B/C	Reject

Diagram:
https://github.com/Andhiki/dfa-vendingmachine/blob/main/Diagram.pdf

Penjelasan Implementasi Program

Libraries:

```
#include <iostream>
#include <fstream>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <vector>
#include <algorithm>
#include <sstream>
using namespace std;
```

Library digunakan untuk:

- I/O (iostream, fstream)

- Penyimpanan data (unordered_map, unordered_set, vector)
- Parsing file (sstream)

Hash Pair:

```
struct hash_pair {
    template <class T1, class T2>
    size_t operator()(const pair<T1, T2>& p) const {
        return hash<T1>{}(p.first) ^ hash<T2>{}(p.second);
    }
};
```

Dibutuhkan untuk menggunakan pair<string, string> sebagai key di unordered_map. Digunakan untuk merepresentasi transisi DFA

DFA Class

```
class DFA {
private:
    unordered_set<string> states;
    unordered_set<string> alphabet;
    string start_state;
    unordered_set<string> accept_states;
    unordered_map<pair<string, string>, string, hash_pair> transitions;
    string current_state;
    vector<string> path;
    unordered_map<string, int> prices = {{"A", 3000}, {"B", 4000}, {"C", 6000}};
    int balance = 0;
```

Variabel yang digunakan:

- unordered_set<string> states;
  - Kumpulan state (e.g., S0, S1000, Reject)
- unordered_set<string> alphabet;
  - Input valid (1000, 2000, A, B, C)
- string start_state;
  - State awal (S0)
- unordered_set<string> accept_states;
  - State penerima (e.g., DispenseA_0)
- unordered_map<pair<string, string>, string, hash_pair> transitions;
  - Fungsi transisi
- string current_state;
  - State saat ini
- vector<string> path;
  - Riwayat state yang dilalui

- unordered_map<string, int> prices;
    - Harga minuman (A: 3000, B: 4000, C: 6000)
- int balance;
    - Saldo terkumpul

Fungsi-fungi penting:

- reset(): Mengembalikan mesin ke state awal (S0).

```
void reset() {
    current_state = start_state;
    balance = 0;
    path.clear();
    path.push_back(start_state);
}
```

- process_input(): Memproses input dan mengupdate state DFA.

```
bool process_input(string input) {
    if (!is_valid_input(input)) {
        return false;
    }

    auto key = make_pair(current_state, input);
    if (transitions.find(key) != transitions.end()) {
        current_state = transitions[key];
        path.push_back(current_state);

        if (isdigit(input[0])) {
            balance += stoi(input);
            // Jika saldo melebihi 10.000, reset otomatis
            if (balance > 10000) {
                current_state = "Reject";
                path.push_back(current_state);
                print_path();
                cout << "Saldo melebihi batas 10.000. Status: REJECTED. Mesin akan
                direset." << endl;
                reset();
                return false;
            }
        }
        return true;
    }
    return false;
}
```

- process_drink(): Menangani pembelian minuman dan kembalian.

```
void process_drink(string drink) {
    if (prices.count(drink)) {
        if (balance >= prices[drink]) {
            print_path();
            cout << "Minuman " << drink << " dapat dibeli. Status: ACCEPTED." << endl;
            if (balance > prices[drink]) {
                cout << "Kembalian: " << (balance - prices[drink]) << endl;
            }
            reset();
        } else {
            print_path();
            cout << "Uang tidak cukup. Status: REJECTED." << endl;
            reset();
        }
    }
}
```

Membaca DFA dari file .txt

1. Membuka file vending_dfa.txt

```
DFA load_dfa_from_file(const string& filename) {
    ifstream file(filename);
    if (!file.is_open()) {
        cerr << "Error: Could not open DFA file" << endl;
        exit(1);
    }
```

2. Membaca file .txt untuk states, alphabet, start, accept, dll

```
while (getline(file, line)) {
    // Remove comments
    line = line.substr(0, line.find('#'));
    // Trim whitespace
    line.erase(remove_if(line.begin(), line.end(), ::isspace), line.end());
    if (line.empty()) continue;

    if (line.find("States:") == 0) {
        stringstream ss(line.substr(7));
        string state;
        while (getline(ss, state, ',')) {
            if (!state.empty()) states.insert(state);
        }
    }
    else if (line.find("Alphabet:") == 0) {
        stringstream ss(line.substr(9));
        string symbol;
        while (getline(ss, symbol, ',')) {
            if (!symbol.empty()) alphabet.insert(symbol);
        }
    }
    else if (line.find("Start:") == 0) {
        start_state = line.substr(6);
    }
    else if (line.find("Accept:") == 0) {
        stringstream ss(line.substr(7));
        string state;
        while (getline(ss, state, ',')) {
            if (!state.empty()) accept_states.insert(state);
        }
    }
    else if (line.find("Transitions:") == 0) {
        continue;
```

3. Return state, alphabet, transitions, dll yang ditemukan

```
return DFA(states, alphabet, start_state, accept_states, transitions);
```

Main method:

- Initialize DFA menggunakan function load_dfa_from_file

```
int main() {
    DFA vending_machine = load_dfa_from_file("vending_dfa.txt");
```

- While loop utama yang menerima dan menunggu input user

```
while (true) {
    cout << "Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): ";
    cin >> input;

    if (input == "exit") break;

    if (!vending_machine.is_valid_input(input)) {
        cout << "Input tidak valid! Harap masukkan 1000, 2000, 5000, 10000, A, B, atau C" << endl;
        continue;
    }

    if (vending_machine.process_input(input)) {
        if (vending_machine.is_accepted()) {
            string drink = input;
            vending_machine.process_drink(drink);
        }
        else if (vending_machine.is_reject()) {
            vending_machine.print_path();
            cout << "Transaksi ditolak! Mesin akan direset." << endl;
            vending_machine.reset();  // Manual reset after rejection
        }
        else {
            if (isdigit(input[0])) {
                cout << "Saldo saat ini: " << vending_machine.get_balance() << endl;
                vending_machine.show_available_drinks();
            }
        }
    }
    else {
        cout << "Transisi tidak valid untuk state saat ini" << endl;
    }
}
```

Cara kerja:

1. Menunggu input user (uang/beli minuman)
2. Ketika vending machine aktif, tunjukkan saldo dan minuman apa saja yang bisa dibeli dengan saldo tersebut
3. Ketika user memberi input, cek apakah input valid (termasuk salah satu alphabet)
    a. Jika valid, cek apabila sudah sampai Reject state (sink state) atau Accept state
        i. Jika sampai state Reject:
            1. Cetak lintasan DFA
            2. Tolak transaksi dan
            3. Reset vending machine
        ii. Jika sampai state Accept/berhasil membeli minuman

1. Cetak lintasan DFA
2. Kabari pengguna, bahwa Minuman X dapat dibeli dan Status: ACCEPTED.
3. Berikan kembalian jika ada
4. Reset vending machine

iii. Jika masih belum sampai Reject/Accept State, maka ulangi dari step 2

4. Program akan berhenti ketika user mengetik ‘exit’

Contoh input dan output

1. Contoh Input 1 (Pembelian berhasil dengan jumlah uang pas)

```
❯ ./vm
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 2000
Saldo saat ini: 2000
ON: Tidak ada minuman yang tersedia
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 1000
Saldo saat ini: 3000
ON: Minuman A
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): A
Lintasan DFA: S0 → S2000 → S3000 → DispenseA_0
Minuman A dapat dibeli. Status: ACCEPTED.
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C):
```

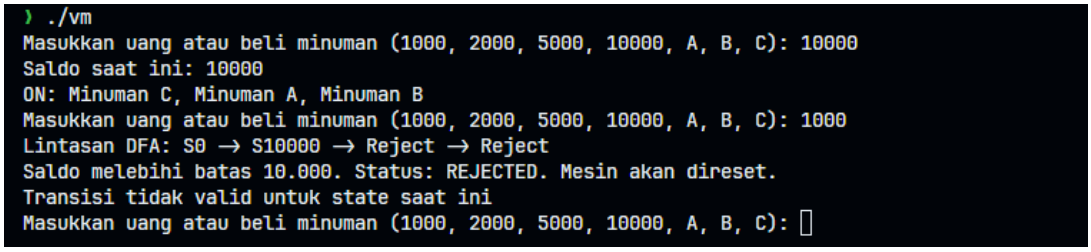
2. Contoh Input 2 (Pembelian gagal karena uang kurang)

```
❯ ./vm
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 1000
Saldo saat ini: 1000
ON: Tidak ada minuman yang tersedia
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 1000
Saldo saat ini: 2000
ON: Tidak ada minuman yang tersedia
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): B
Lintasan DFA: S0 → S1000 → S2000 → Reject
Uang tidak cukup. Status: REJECTED. Transaksi ditolak dan Mesin akan direset.
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C):
```

3. Contoh Input 3 Bonus (Pembelian dengan jumlah uang lebih dari harga minuman, dengan fitur kembalian)

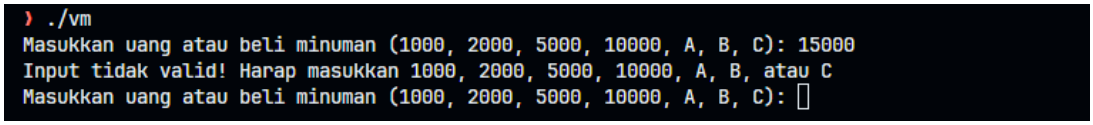
```
❯ ./vm
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): 5000
Saldo saat ini: 5000
ON: Minuman A, Minuman B
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C): B
Lintasan DFA: S0 → S5000 → DispenseB_1000
Minuman B dapat dibeli. Status: ACCEPTED.
Kembalian: 1000
Masukkan uang atau beli minuman (1000, 2000, 5000, 10000, A, B, C):
```

4. Contoh Input Tambahan 1 (Saldo lebih dari 10.000)

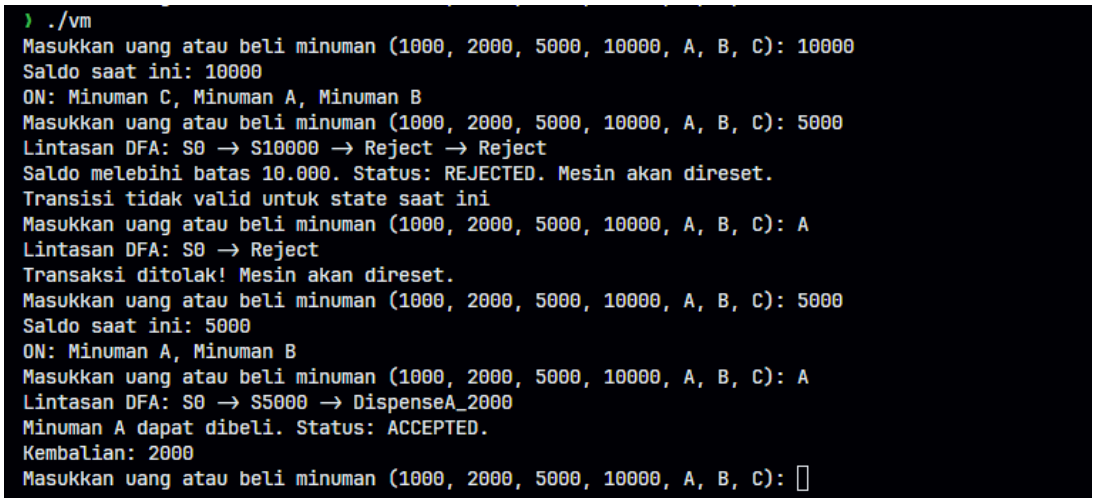


   Saldo akan di reset menjadi 0.

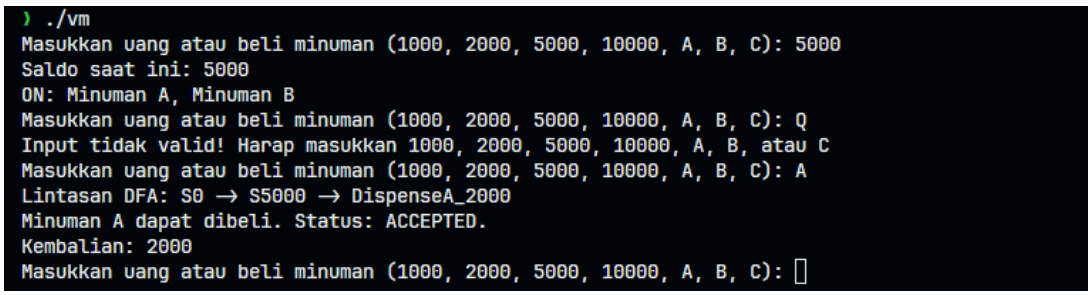
5. Contoh Input Tambahan 2 (Input uang lebih dari 10.000)



6. Contoh Input Tambahan 2 (Membeli minuman setelah reset otomatis akibat saldo berlebih)



7. Contoh Input Tambahan 3 (Kombinasi angka dan huruf untuk input)



Detail Kontribusi masing-masing anggota tim

1. Andreandhiki
    a. Membuat notasi DFA (state, alphabet, transition, tabel transition, diagram)
    b. Mengimplementasi DFA ke kode
    c. Membuat laporan
2. Andrian
    a. Membuat notasi DFA (state, alphabet, transition, tabel transition, diagram)
    b. Mengimplementasi DFA ke kode
    c. Membuat laporan

Kedua anggota tim memberi kontribusi pada semua aspek tugas ini dan setuju bahwa kedua anggota memberi kontribusi yang seimbang.