МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМ. Б.ХМЕЛЬНИЦЬКОГО

**Факультет обчислювальної техніки, інтелектуальних та управляючих систем**

**Теорія алгоритмів**

Лабораторна робота №2

Машина Тюрiнга

Виконала

студентка групи ЗКН-21

Кузьмінова Марія

2023

Варіант 2

Завдання:

Зображення, що містить текст, Шрифт, квитанція, алгебра

Автоматично згенерований опис

Виконання:

class TuringMachine {

constructor(alphabet, transitions) {

this.alphabet = alphabet;

this.transitions = transitions;

this.tape = [];

this.headPosition = 0;

this.currentState = '';

this.output = '';

}

setInitialState(state) {

this.currentState = state;

}

setTape(input) {

this.tape = input.split('');

}

run() {

while (true) {

const currentSymbol = this.tape[this.headPosition];

const transition = this.transitions[this.currentState][currentSymbol];

if (!transition) {

break;

}

const { writeSymbol, moveDirection, nextState } = transition;

this.tape[this.headPosition] = writeSymbol;

this.headPosition += moveDirection === 'R' ? 1 : -1;

this.currentState = nextState;

this.output += writeSymbol;

}

}

getOutput() {

return this.output;

}

}

// Приклад використання

const alphabet = ['0', '1']; // Алфавіт

const transitions = {

q0: {

'0': { writeSymbol: '1', moveDirection: 'R', nextState: 'q1' },

'1': { writeSymbol: '0', moveDirection: 'R', nextState: 'q2' },

},

q1: {

'0': { writeSymbol: '0', moveDirection: 'R', nextState: 'q1' },

'1': { writeSymbol: '1', moveDirection: 'R', nextState: 'q1' },

},

q2: {

'0': { writeSymbol: '1', moveDirection: 'L', nextState: 'q0' },

'1': { writeSymbol: '0', moveDirection: 'L', nextState: 'q2' },

},

};

const tm = new TuringMachine(alphabet, transitions);

tm.setInitialState('q0'); // Початковий стан

tm.setTape('0000000'); // Вхідна стрічка

tm.run(); // Запускаємо машину Тюрінга

console.log(tm.getOutput()); // Виводимо непусте слово

**Пояснення до коду:**

Цей код реалізує клас ‘TuringMachine’, який представляє машину Тюрінга. Машина Тюрінга - це абстрактна обчислювальна модель, яка складається з набору правил переходу і працює на основі руху головки по стрічці символів.

Основні методи класу ‘TuringMachine’:

1. Конструктор: Приймає алфавіт і переходи як параметри і ініціалізує внутрішні змінні, такі як алфавіт, переходи, стрічка на стрічці, позиція головки, поточний стан та вивід.
2. ‘setInitialState(state)’: Встановлює початковий стан машини.
3. ‘setTape(input)’: Встановлює вхідну стрічку, розбиваючи її на окремі символи та зберігаючи їх у масиві.
4. ‘run()’: Запускає виконання машини Тюрінга. У циклі він отримує поточний символ зі стрічки за допомогою позиції головки, знаходить відповідний перехід для поточного стану та символу, оновлює стрічку, позицію головки, поточний стан та вивід відповідно до переходу. Цикл продовжується, поки є визначений перехід для поточного стану та символу. Коли немає визначеного переходу, виконання зупиняється.
5. ‘getOutput()’: Повертає вивід машини Тюрінга.

Приклад використання:

У цьому прикладі машина Тюрінга працює на двох символах - '0' та '1'. Є три стани: 'q0', 'q1' та 'q2'. Переходи задаються у вигляді об'єкта, де ключі - це стани, а значення - об'єкти, що визначають переходи для кожного символу алфавіту.

Після ініціалізації машини Тюрінга і встановлення початкового стану та вхідної стрічки, викликається метод ‘run()’, який починає виконання машини Тюрінга. Після закінчення виконання викликається метод ‘getOutput()’, який повертає вивід машини Тюрінга.

У цьому конкретному прикладі, машина Тюрінга заміняє '0' на '1' та '1' на '0', зсуваючи головку вправо досягнення кінця стрічки. Вивідом машини Тюрінга є непусте слово, яке складається з символів, що були записані на стрічці під час виконання. У цьому випадку, вихід буде '1000001', оскільки машина Тюрінга замінила перший символ на '1', другий символ залишилася незмінним, а всі інші символи були змінені на '0'.

**Додаткове завдання**

**Завдання:**

**Зображення, що містить текст, Шрифт, квитанція

Автоматично згенерований опис**

**Виконання:**

function turingMachine(tape, transitions, initialState, finalStates, speed) {

let state = initialState;

let position = 0;

while (!finalStates.includes(state)) {

const symbol = tape[position];

if (!(state in transitions) || !(symbol in transitions[state])) {

console.log('No transition found for current state and symbol.');

break;

}

const { newState, newSymbol, direction } = transitions[state][symbol];

const transitionsString = `Transitions: (${state}, ${symbol}) -> (${newState}, ${newSymbol}, ${direction})`;

const tapeString = 'Tape: ' + tape.slice(0, position).join('') + '[' + symbol + ']' + tape.slice(position + 1).join('');

console.log(transitionsString);

console.log(tapeString);

console.log();

tape[position] = newSymbol;

if (direction === 'R') {

position++;

} else if (direction === 'L') {

position--;

}

state = newState;

sleep(speed);

}

if (finalStates.includes(state)) {

console.log('Final state reached. Turing machine halted.');

}

}

function sleep(ms) {

return new Promise((resolve) => setTimeout(resolve, ms));

}

// Приклад використання

const tape = ['0', '1', '1', '0', '1', '0', '1', '1', '0'];

const transitions = {

q0: {

'0': { newState: 'q1', newSymbol: '1', direction: 'R' },

'1': { newState: 'q0', newSymbol: '0', direction: 'L' },

},

q1: {

'0': { newState: 'q1', newSymbol: '0', direction: 'R' },

'1': { newState: 'q1', newSymbol: '1', direction: 'R' },

},

};

const initialState = 'q0';

const finalStates = ['q1'];

const speed = 1000; // Пауза між ітераціями у мілісекундах

turingMachine(tape, transitions, initialState, finalStates, speed);

**Пояснення до коду:**

У цьому прикладі ми використовуємо масив ‘tape’ для представлення вхідної стрічки. Таблиця переходів ‘transitions’ представлена у вигляді об'єкта, де ключі - стани, а значення - об'єкти, що містять новий стан, новий символ і напрямок руху головки. Задані початковий стан ‘initialState’ і кінцеві стани ‘finalStates’. Параметр ‘speed’ визначає паузу між кожною ітерацією у мілісекундах.

**Результат коду:**

**Зображення, що містить текст, Шрифт, знімок екрана, ряд

Автоматично згенерований опис**