ЛАБОРАТОРНА РОБОТА №1

“Перевірка включення однієї регулярної мови в іншу. Перевірка належності слова до мови, що акцептується заданим автоматом.”

Виконали студенти:

1 курсу магістратури, групи ПЗС-1

Артюхов Яків

Бондаренко Ярослав

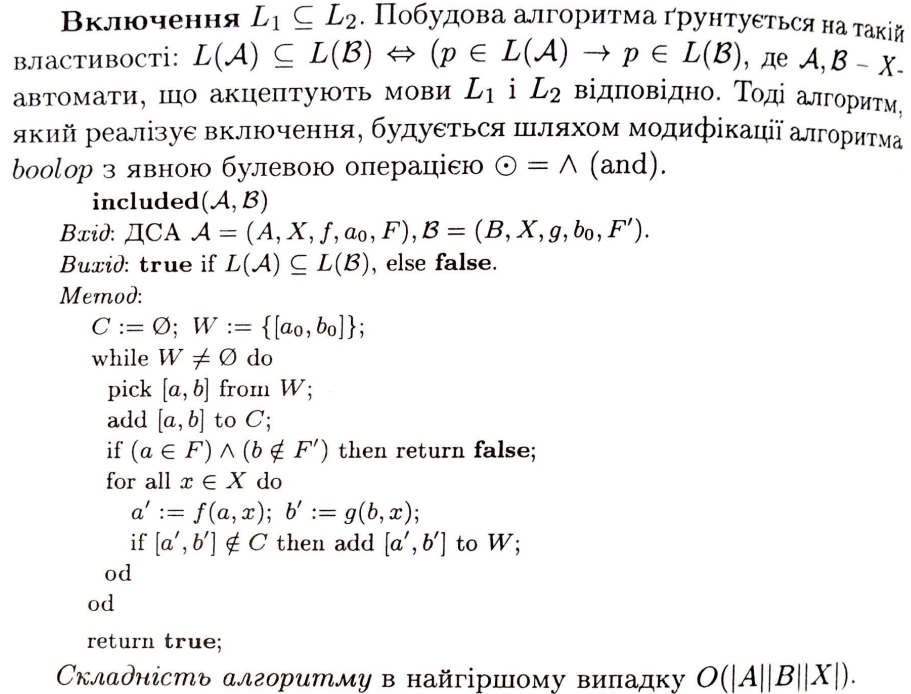
Грицишен Святослав

Іванченко Максим

Козій Євгеній

### Задача 1. Перевірка включення однієї регулярної мови в іншу.

###### Алгоритм



###### Код алгоритму (мовою Python)

def \_contains\_fsm(self, fsm):

        C = []

        W = [(self.start\_state, fsm.start\_state)]

        while W:

            a, b = W.pop(0)

            C.append((a, b))

            if a in self.final\_states and b not in fsm.final\_states:

                return False

            for x in self.alphabet:

                \_a = self.transit(a, x)

                \_b = fsm.transit(b, x)

                if \_a is None and \_b is not None:

                    return False

                if \_a is None or \_b is None:

                    continue

                if (\_a, \_b) not in C:

                    W.append((\_a, \_b))

        return True

###### Код алгоритму (мовою C++)

bool contains\_fsm(FiniteStateMachine fsm) {

        std::set<std::pair<int, int>> W;

        std::set<std::pair<int, int>> C;

        W.insert(std::make\_pair(start\_state, fsm.start\_state));

        while (!W.empty()) {

            auto a\_b = \*W.begin();

            W.erase(W.begin());

            C.insert(a\_b);

            if (final\_states.count(a\_b.first) && final\_states.count(a\_b.second))

                return false;

            for (auto const &x: alphabet) {

                int \_a = transit(a\_b.first, x);

                int \_b = fsm.transit(a\_b.second, x);

                if (\_a == -1 && \_b != -1) return false;

                if (\_a == -1 || \_b == -1) continue;

                if (C.find(std::make\_pair(\_a, \_b)) == C.end()) W.insert(std::make\_pair(\_a, \_b));

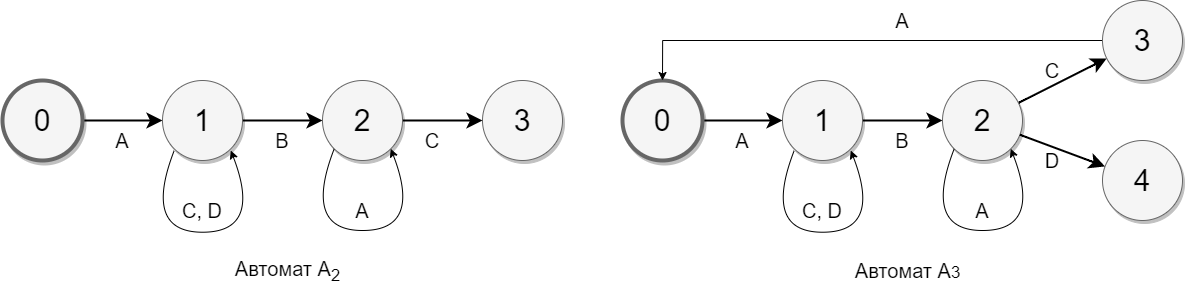
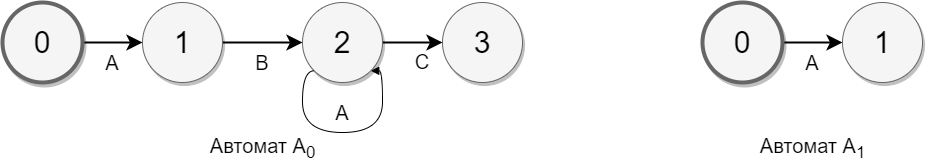
            }

        }

        return true;

    }

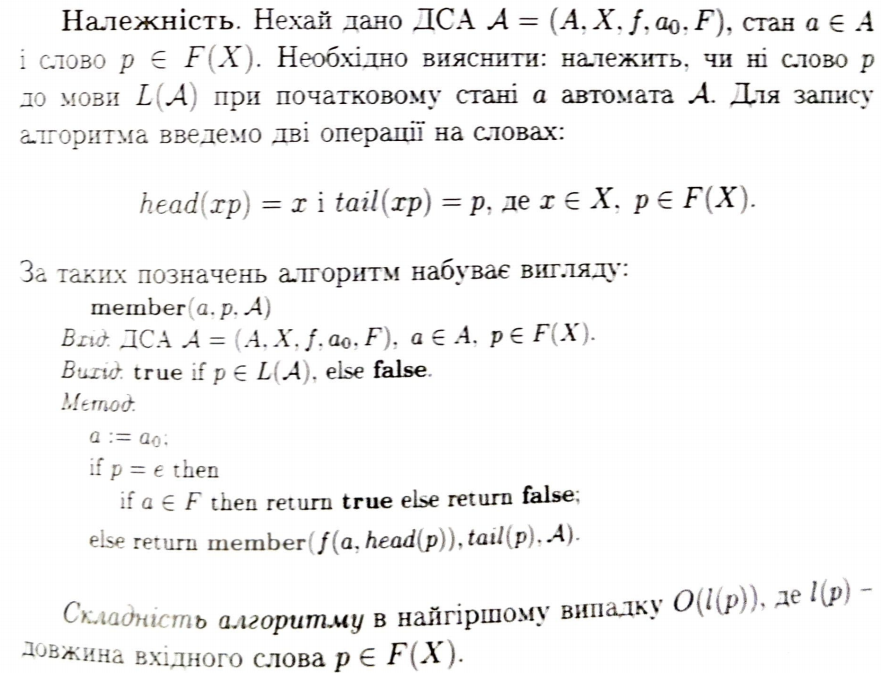
###### Результати роботи

Для дослідження роботи цього алгоритму було перевірено його роботу з перевірки включення мови автомату X в мову автомату Y. Було створено кілька автоматів. 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A1 в А0 | A1 в А3 | A0 в А3 | A2 в А3 |
| С++ | 4.1 мкс | 4.5 мкс | 5.1 мкс | 5.4 мкс |
| Python | 3.9 мкс | 4.2 мкс | 4.6 мкс | 4.8 мкс |

### Задача 2. Перевірка належності слова до мови, що акцептується заданим автоматом.

###### Алгоритм



###### Код алгоритму (мовою Python)

def \_contains\_word(self, word):

        cur\_state = self.start\_state

        while word:

            cur\_state = self.transit(cur\_state, word[0])

            if cur\_state is None: return False

            word = word[1:]

        if cur\_state in self.final\_states: True

        return False

###### Код алгоритму (мовою C++)

bool contains\_word(std::string word) {

        int cur\_state = start\_state;

        while (!word.empty()) {

            cur\_state = transit(cur\_state, word[0]);

            if (cur\_state == -1) return false;

            word = word.substr(1);

        }

        if (final\_states.count(cur\_state)) return true;

        return false;

    }

###### Результати роботи

Для перевірки ефективності роботи цього алгоритму було перевірено його роботу з перевірки належності слів “abc”, «abaaaaaaaaac» та «aba(50 разів)c» до мови, що акцептується вищезгаданими автоматами.

Середній час роботи алгоритму:

|  |  |  |
| --- | --- | --- |
|  | C++ | Python |
| А0 | | |
| abc | 2 мкс | 0.9 мкс |
| abaaaaaaaaac | 7 мкс | 3 мкс |
| aba(50 разів)c | 30 мкс | 12.3 мкс |
| А2 | | |
| abc | 2 мкс | 1 мкс |
| abaaaaaaaaac | 8.5 мкс | 3 мкс |
| aba(50 разів)c | 37.8 мкс | 12.8 мкс |
| А3 | | |
| abc | 2 мкс | 1.2 мкс |
| abaaaaaaaaac | 9 мкс | 3.3 мкс |
| acdcbaaaacaacdcdbaaad | 16 мкс | 5.65 мкс |
| aba(50 разів)c | 40 мкс | 13.3 мкс |