

**МИНОБРНАУКИ РОССИИ**  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра информатики и программирования

**АНАЛИЗ АЛГОРИТМА БОЙЕРА — МУРА**  
**ОТЧЕТ**

студентки 2 курса 211 группы  
направления 02.03.02 — Фундаментальная информатика и информационные  
технологии  
факультета компьютерных наук и информационных технологий  
Никитенко Яны Валерьевны

## СОДЕРЖАНИЕ

1	Текст программы .....	3
2	Анализ .....	5

## 1 Текст программы

```
// Функция для создания таблицы смещений по символам
vector<int> CreateBadCharTable(const string& pattern) {
    const int ALPHABET_SIZE = 256;
    vector<int> badCharTable(ALPHABET_SIZE, -1);

    for (int i = 0; i < pattern.size(); ++i) {
        badCharTable[static_cast<int>(pattern[i])] = i;
    }

    return badCharTable;
}

//

// Функция для поиска подстроки с использованием алгоритма Бойера — Мура
void BoyerMoore(const string& text, const string& pattern) {
    int m = pattern.size();
    int n = text.size();
    vector<int> badCharTable = CreateBadCharTable(pattern);

    int s = 0;
    bool first = true; // Флаг для определения первого вывода

    while (s <= n - m) {
        int j = m - 1;

        while (j >= 0 && pattern[j] == text[s + j]) {
            j--;
        }

        if (j < 0) {
            if (!first) {
                cout << " "; // Пробел между числами
            }
        }
    }
}
```

```

        cout << s;
        first = false;
        s += (s + m < n) ? m - badCharTable[static_cast<int>(text[s + m])] : 1;
    }
    else {
        s += max(1, j - badCharTable[static_cast<int>(text[s + j])]);
    }
}

// Если совпадений не найдено, вывод сообщение
if (first) {
    cout << "Совпадений не найдено";
}
}
//

```

## 2 Анализ

### Временная сложность алгоритма Бойера-Мура:

- **Предобработка:**  $O(m + |\Sigma|)$ , где  $|\Sigma|$  - размер алфавита
- **Худший случай:**  $O(n \cdot m)$
- **Лучший случай:**  $O(n/m)$
- **Средний случай на случайных текстах:**  $O(n)$

### Обоснование:

- **Предобработка:** Построение таблицы "плохих символов" требует  $O(|\Sigma|)$  памяти и  $O(m)$  времени
- **Худший случай:** Возникает, когда шаблон и текст имеют периодическую структуру (например, `text = "AAA...AAA"`, `pattern = "AAA"`). На каждой из  $O(n)$  позиций выполняется почти полное сравнение шаблона за  $O(m)$
- **Лучший случай:** Когда первый же символ текста не встречается в шаблоне, алгоритм сдвигается на  $m$  позиций за  $O(1)$ . Количество сдвигов:  $O(n/m)$
- **Средний случай:** На случайных текстах алгоритм демонстрирует сублинейное поведение. Эвристика "плохого символа" позволяет делать сдвиги в среднем близкие к  $m$ , что дает  $O(n)$  сравнений

### Практическая эффективность:

Несмотря на квадратичную худшую сложность, алгоритм Бойера-Мура часто оказывается быстрее алгоритмов с гарантированной линейной сложностью (как КМП) благодаря:

- Большим сдвигам шаблона
- Проверке символов справа налево
- Эффективной работе с естественными языками