ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ст. преподаватель |  |  |  | П. А. Степанов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| ОЦЕНКА КАЧЕСТВА ТЕСТОВОЙ БАЗЫ |
| по курсу: УПРАВЛЕНИЕ КАЧЕСТВОМ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4631 |  |  |  | С.А. Гришин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

**Задание**

В рамках лабораторной работы необходимо произвести функциональное тестирование кода методом белого ящика (всех ветвей)

Задание на лабораторную работу.

3. Оценить по отдельности и вместе покрытие тестами, разработанными в лабораторной работе номер 2 и 3.

4. Описать недостающие тесты

5. Выполнить инъекцию багов, оценить качество разработанных тестов.

**Вариант 9: Функция, производящая поиск заданного набора строк в текстовом файле.**

Поиск должен уметь находить любую строку из набора, при этом должен правильно обрабатывать переносы текста.

Использовать алгоритм: Ахо-Корасик.

**Решение**

Покрытие тестами во второй лабораторной работе были не полноценны, так как не покрывали одно из условий работы кода. Третья лабораторная работа покрывала и этот случай. После исправления удалось добиться 100% покрытия кода.

Инъекция багов:

1. Уберём условие отката при ошибке.

for curNode.child[v] == nil && curNode != m.root {

curNode = curNode.fail

}

И наши тесты упадут при на тесте, который добавил нам 100% покрытия кода.

--- FAIL: TestFullCover (0.00s)

panic: runtime error: invalid memory address or nil pointer dereference [recovered]

panic: runtime error: invalid memory address or nil pointer dereference

[signal 0xc0000005 code=0x0 addr=0x10 pc=0x4fdd82]

1. Изменим условие цикла, для добавления в результирующий массив

p = curNode

for p != m.root && p.count > 0 && !m.mark[p.index] {

m.mark[p.index] = true

for i := 0; i < p.count - 1; i++ {

ret = append(ret, p.index)

}

p = p.fail

}

И снова наши тесты не будут пройдены

--- FAIL: TestSimple (0.00s)

ahocorasick\_test.go:21: Got: [], want: [0 1].

**Вывод**

Таким образом простые тесты тоже позволяют отлавливать случайные ошибки. Чём больше покрытие тестами, тем лучше они отлавливают ошибки. Первую ошибки отловили тесты только из 3 лабораторной работы.