Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

Отчет

по лабораторной работе №2

«Сравнительный анализ методов поиска подстроки в строке»

Выполнил:
студент группы 213 Файтельсон А.А.

Проверил:
ассистент кафедры ПОиАИС Овсянников А.В.

Курск, 2024

Цель работы: Изучение методов поиска подстроки в строке и приобретение навыков в проведении сравнительного анализа различных методов.

Постановка задачи:

1. Изучить временные характеристики алгоритмов.
2. Изучить методы подстроки в строке.
3. Программно реализовать 2 метода в соответствии с вариантом:

Таблица 1 – Индивидуальный вариант

Вариант 8	Метод 1.	8. Поиск подстроки в строке с помощью суффиксного массива
	Метод 2.	2. Алгоритм Рабина-Карпа

1. Разработать и программно реализовать средство для проведения экспериментов по определению временных характеристик алгоритмов.
2. Провести эксперименты по определению временных характеристик алгоритмов поиска подстроки в строке.
3. Определить время исполнение алгоритма с помощью бенчмарков.
4. Определить порядок функций временной сложности алгоритмов.

Алгоритмы решения задачи в текстуальном виде.

8. Поиск подстроки в строке с помощью суффиксного массива

Для решения задачи поиска подстроки в строке можно использовать суффиксный массив, который является мощным инструментом для работы со строками. Суффиксный массив — это массив всех суффиксов строки, отсортированных в лексикографическом порядке. После построения суффиксного массива задачу поиска подстроки можно решить за логарифмическое время с использованием бинарного поиска.

2. Алгоритм Рабина-Карпа

Алгоритм Рабина-Карпа — это эффективный метод для поиска подстроки в строке, основанный на технике хеширования. Этот алгоритм используется для быстрого поиска всех вхождений подстроки в строку с помощью сравнения хеш-значений вместо прямого сравнения подстрок. Это позволяет сократить количество сравнений, что делает его полезным для решения задачи поиска подстроки в строке.

Основная идея заключается в том, чтобы вычислить хеш-значение искомой подстроки и сравнивать его с хеш-значениями всех подстрок в исходной строке той же длины. Если хеш-значения совпадают, проводится точное посимвольное сравнение. Благодаря хешированию этот метод имеет лучшее среднее время работы по сравнению с простыми методами, особенно в случаях, когда требуется искать несколько вхождений подстроки.

Результаты сравнительного анализа алгоритмов.
(в наносекундах на операцию)

```
❯ go test -bench=.
goos: linux
goarch: amd64
pkg: laba
cpu: Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz
BenchmarkFindSubStringWithSuffixArray/input:_"that"__-4          4065          289828 ns/op
BenchmarkFindSubStringWithSuffixArray/input:_"I've"__-4          4089          303644 ns/op
BenchmarkFindSubStringWithSuffixArray/input:_"I_am"__-4          3931          305339 ns/op
BenchmarkFindSubStringWithSuffixArray/input:_"no"__-4            4044          297399 ns/op
BenchmarkFindSubStringWithSuffixArray/input:_"on"__-4            3819          294295 ns/op
```

Рисунок 1 – Бенчмарк функции поиска подстроки с помощью суффиксного массива

```
BenchmarkFindSubStringWithRabinKrapin/input:_"that"__-4          44547          26968 ns/op
BenchmarkFindSubStringWithRabinKrapin/input:_"I've"__-4          44239          26922 ns/op
BenchmarkFindSubStringWithRabinKrapin/input:_"I_am"__-4          44437          26908 ns/op
BenchmarkFindSubStringWithRabinKrapin/input:_"no"__-4            42904          28408 ns/op
BenchmarkFindSubStringWithRabinKrapin/input:_"on"__-4            39422          29446 ns/op
PASS
ok      laba    14.462s
```

Рисунок 2 – Бенчмарк функции поиска подстроки с помощью алгоритма Рабина Крапина

По бенчмарку можно сделать вывод, что алгоритм Рабина-Крапина работает быстрее.

Листинг программы

main.go

```
package main

import (
	"index/suffixarray"
	"math"
	"slices"
)

func FindSubStringWithSuffixArray(s string, subs string) []int {
	index := suffixarray.New([]byte(s))
	offset := index.Lookup([]byte(subs), -1)
	// fmt.Print("Indexes of substring - ", offset)
	return offset
}

func FindSubStringWithRabinKrapin(s string, sub string) []int {
	M := len(sub)
	N := len(s)
	d := 26
	q := math.MaxInt64
	var i, j int
	p := 0 // hash value for subtern
	t := 0 // hash value for text
	h := 1 // the value of h would be "pow(d, M-1) % q"
	var ans []int
	// Calculate the value of h as "pow(d, M-1) % q"
	for i = 0; i < M-1; i++ {
		h = (h * d) % q
	}

	// Calculate the hash value of subtern and first window of text
	for i = 0; i < M; i++ {
		p = (d*p + int(sub[i])) % q
		t = (d*t + int(s[i])) % q
	}

	// Slide the subtern over text one by one
	for i = 0; i <= N-M; i++ {

		// Check the hash values of current window of text and subtern
		if p == t {
			// Check characters one by one
			for j = 0; j < M; j++ {
				if s[i+j] != sub[j] {
```

```
                        break
                    }
                }

                // If p == t and subtern matches
                if j == M {
                    ans = append(ans, i)
                    // fmt.Printf("subtern found at index %d\n", i)
                }
            }

            // Calculate hash value for the next window of text
            if i < N-M {
                t = (d*(t-int(s[i])*h) + int(s[i+M])) % q

                // We might get negative value of t, convert it to positive
                if t < 0 {
                    t = (t + q)
                }
            }
        }
        slices.Reverse(ans)
        return ans
}
func main() {}

// func TestFindSubStringWithSuffixArray(t *testing.T) {
//     assert.Equal(t, FindSubStringWithSuffixArray("Banana", "ana"), []int{3, 1}, "They should be equal")
// }
```

main_test.go

```
package main

import (
    "fmt"
    "testing"

    "github.com/stretchr/testify/assert"
)

func TestFindSubStringWithK(t *testing.T) {
    assert.Equal(t, FindSubStringWithSuffixArray("Banana", "ana"), []int{3, 1}, "They should be equal")
    assert.Equal(t, FindSubStringWithSuffixArray("Bananananana", "ana"), []int{9, 7, 5, 3, 1}, "They should be equal")
```

```
	assert.Equal(t, FindSubStringWithSuffixArray("Do you like what do you see?", "what"), []int{12}, "They should be equal")
	assert.Equal(t, FindSubStringWithSuffixArray("Do you like what do you see?", "My name is, What?"), ([]int)(nil), "They should be equal")
}

func TestFindSubStringWithRabinKrapin(t *testing.T) {
	assert.Equal(t, FindSubStringWithRabinKrapin("Banana", "ana"), []int{3, 1}, "They should be equal")
	assert.Equal(t, FindSubStringWithRabinKrapin("Bananananana", "ana"), []int{9, 7, 5, 3, 1}, "They should be equal")
	assert.Equal(t, FindSubStringWithRabinKrapin("Do you like what do you see?", "what"), []int{12}, "They should be equal")
	assert.Equal(t, FindSubStringWithRabinKrapin("Do you like what do you see?", "My name is, What?"), ([]int)(nil), "They should be equal")
}

func BenchmarkFindSubStringWithSuffixArray(b *testing.B) {
	for _, v := range TestSubString {
		b.Run(fmt.Sprintf("input: %q \n", v.input), func(b *testing.B) {
			for i := 0; i < b.N; i++ {
				FindSubStringWithSuffixArray(TestString, v.input)
			}
		})
	}
}

func BenchmarkFindSubStringWithRabinKrapin(b *testing.B) {
	for _, v := range TestSubString {
		b.Run(fmt.Sprintf("input: %q \n", v.input), func(b *testing.B) {
			for i := 0; i < b.N; i++ {
				FindSubStringWithRabinKrapin(TestString, v.input)
			}
		})
	}
}
```

help.go

```
//STRING for testing
package main

var TestString = `Ever on and on, I continue circling
With nothing but my hate and the carousel of agony
Till slowly I forget and my heart starts vanishing
And suddenly I see that I can't break free, I'm
```

Slipping through the cracks of a dark eternity
With nothing but my pain and the paralyzing agony
To tell me who I am! Who I was!
Uncertainty enveloping my mind
Till I can't break free and

Maybe it's a dream, maybe nothing else is real
But it wouldn't mean a thing if I told you how I feel
So I'm tired of all the pain, all the misery inside
And I wish I could live feeling nothing but the night
You could tell me what to say, you could tell me where to go
But I doubt that I would care, and my heart would never know
If I make another move, there'll be no more turning back
Because everything would change, and it all would fade to black

Will tomorrow ever come? Will I make it through the night?
Will there ever be a place for the broken in the light?
Am I hurting? Am I sad? Should I stay or should I go?
I've forgotten how to tell, did I ever even know?
Can I take another step? I've done everything I can
All the people that I see, they will never understand
If I find a way to change, if I step into the light
Then I'll never be the same and it all will fade to white

Ever on and on, I continue circling
With nothing but my hate and the carousel of agony
Till slowly I forget and my heart starts vanishing
And suddenly I see that I can't break free, I'm

Slipping through the cracks of a dark eternity
With nothing but my pain and the paralyzing agony
To tell me who I am! Who I was!
Uncertainty enveloping my mind
Till I can't break free and

Maybe it's a dream, maybe nothing else is real
But it wouldn't mean a thing if I told you how I feel
So I'm tired of all the pain, all the misery inside
And I wish I could live feeling nothing but the night
You could tell me what to say, you could tell me where to go
But I doubt that I would care, and my heart would never know
If I make another move, there'll be no more turning back
Because everything would change, and it all would fade to black

If I make another move, if I take another step
Then it all would fall apart

There'd be nothing of me left
If I'm crying in the wind, if I'm crying in the night
Will there ever be a way? Will my heart return to white?
Can you tell me who you are? Can you tell me where I am?
I've forgotten how to see
I've forgotten if I can
If I opened up my eyes, there'd be no more going back
'Cause I'd throw it all away and it all would fade to black

Her carnal temptation
Took over my mind
Back then
Running from the fallout
Has turned my whole life
To mayhem

[Chorus]
I am the storm that is escaping
Evading
Parental obligations
I am the changer of my name
Life in flames
I am in debt
Just try and find me, you won't take a cent!
Forsakened child abandoned
A house and home I won't provide
Grow up in poverty
Without a father at your side

[Verse 2]
Inherit genetics
They're all that you're getting
'Cause I've run away
I'm keeping what's mine
You're welcome to whine
But I won't pay
You might also like
ไผกะได้ (Anyone)
FIIXD, BIRDMANKKC & 1LIFE
MONEY (ฟังแล้วรวย)
Ple Irin
เด็กอินเตอร์ (Dek Inter)
YOUNGOHM
I'm never gonna pay the rent for
This kid whose mother I can't remember

A parent should be warm and tender
And I just can't be that kind of mentor

[Chorus]
I am the storm that is escaping
Evading
Parental obligations
I am the changer of my name
Off the grid
Hiding from debt
Just try and find me, you won't take a cent!
Forsakened child abandoned
A house and home I won't provide
Grow up in poverty
Without a father at your side
Disappeared into the night
No information left behind
Commitment repelling me
Sorry, I just don't have the time!

[Bridge]
Lurking in the shadows under veil of night
False identity staying out of sight
Dancing through my daily life with no paper trail
Detectives and P.I.'s have all tried and failed
A derelict that reeks of booze and cigarette ashes
Who's never registered to vote and never paid taxes
I only deal in cash, bury it when I make it
So if you want some alimony, come try and take it
You will not take
What I've earned
Secret trauma
Childhood yearned
My dad also
Never was there
Never admit
That I
Cared

[Instrumental Breakdown]

[Chorus 2]
Bury the guilt deep within!
Never still, there's no going home!
Scatter those feelings on the wind!

Because this cash is mine and mine alone!

```
  `

var TestSubString = []struct {
        input string
}{{input: "that"}, {input: "I've"}, {input: "I am"}, {input: "no"}, {input: "on"}}
```