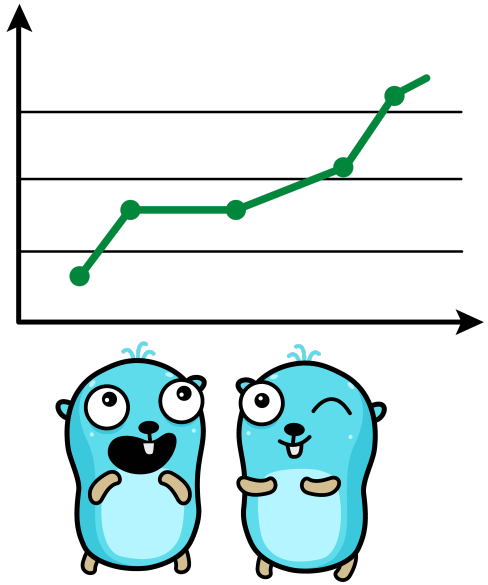
Benchmarking: First Steps

Go Ireland Meetup
6 February 2025

Jakub Jarosz

Agenda

- Testing vs Benchmarking
- Writing benchmarks
- Running benchmarks
- Interpreting results
- Gaming Time!



Design Principles

- Make it correct
- Make it readable
- Make it easy to understand
- ...
- Make it **fast** (enough)

Go Optimization Principle

Optimizing Golang code for performance is almost certainly a waste of your time, for several reasons:

- *Performance doesn't matter*
- *Go is fast*
- *Readability beats speed*

J. Arundel (https://bitfieldconsulting.com/posts/slower)

Benchmarking

- Speed ?
- Petrol ?
- Loops ?
- Why ?

Testing vs Benchmarking

Test

```
func TestMyFunc(t *testing.T) {
    // test logic
}
```

Benchmark

```
func BenchmarkMyFunc(b *testing.B) {
    for i := 0; i < b.N; i++ {
        MyFunc()
    }
}
```

Benchmarks

```
var tt = []string{"uno", "due", "tre"}

func BenchmarkBar(b *testing.B) {
    for i := 0; i < b.N; i++ {
        for _, tc := range tt {
            Bar(tc)
        }
    }
}
```

Running Benchmarks

- bench

```
go test -run none -bench .
```

```
BenchmarkFoo-8    19146404          62.46 ns/op
```

Running Benchmarks

- benchmem

```
go test -run none -bench . -benchmem
```

```
BenchmarkFoo-8    18934608        63.82 ns/op            80 B/op          3 allocs/op
```

Running Benchmarks

- cpu

```
go test -run none -bench . -cpu 2,4,6,8
```

```
BenchmarkIsIsogram-2          412042       2895 ns/op
BenchmarkIsIsogram-4          406560       2905 ns/op
BenchmarkIsIsogram-6          402758       2903 ns/op
BenchmarkIsIsogram-8          402938       2904 ns/op
```

Running Benchmarks

- benchtime

```
go test -run none -bench . -benchtime 2s -benchmem -cpu 2,4,6,8
```

```
BenchmarkIsIsogram-2        833965      2899 ns/op      1426 B/op          16 allocs/op
BenchmarkIsIsogram-4        813745      2894 ns/op      1426 B/op          16 allocs/op
BenchmarkIsIsogram-6        806353      2925 ns/op      1426 B/op          16 allocs/op
BenchmarkIsIsogram-8        804255      2928 ns/op      1426 B/op          16 allocs/op
```

ShareWith...

ShareWith v1

```
func ShareWith(name string) string {
    if name == "" {
        name = "you"
    }
    return "One for " + name + ", one for me."
}
```

ShareWith v2

```
func ShareWith(name string) string {
    if name == "" {
        name = "you"
    }
    return fmt.Sprintf("One for %s, one for me.", name)
}
```

ShareWith Benchmark

```
func BenchmarkShareWith(b *testing.B) {
    for i := 0; i < b.N; i++ {
        for _, tc := range tt {
            hello.ShareWith(tc.input)
        }
    }
}
```

ShareWith Benchmark

run benchmark

```
go test -run none -bench . -benchmem
```

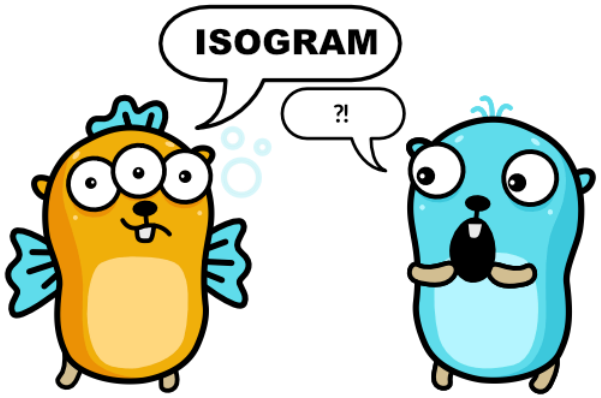
report

```
BenchmarkShareWith-8      18975690         62.86 ns/op             80 B/op          3 allocs/op
```

Demo

Let's play: Isogram

An isogram (also known as a "non-pattern word") is a word or phrase without a repeating letter, however spaces and hyphens are allowed to appear multiple times.



Isogram v1

```
func IsIsogram(word string) bool {

    letters := make(map[rune]bool)

    for _, l := range word {
        letter := unicode.ToLower(l)
        if letter == '-' || unicode.IsSpace(letter) {
            continue
        }
        if letters[letter] {
            return false
        }
        letters[letter] = true
    }
    return true
}
```

Isogram v2

```
func IsIsogram(word string) bool {

    seen := make(map[rune]bool)
    word = strings.ToLower(word)

    for _, l := range word {
        if !unicode.IsLetter(l) {
            continue
        }
        _, ok := seen[l]
        if ok {
            return false
        }
        seen[l] = true
    }
    return true
}
```

Isogram v3

```
func IsIsogram(word string) bool {

    word = strings.ToUpper(word)

    for i, c := range word {
        if c == ' ' || c == '-' {
            continue
        }
        for j := i + 1; j < len(word); j++ {
            if word[i] == word[j] {
                return false
            }
        }
    }
    return true
}
```

Isogram v4

```
func IsIsogram(word string) bool {

    word = strings.ToLower(word)

    for i := 0; i < len(word); i++ {
        for j := i + 1; j < len(word); j++ {
            if word[i] == ' ' || word[j] == '-' {
                continue
            }
            if word[i] == word[j] {
                return false
            }
        }
    }
    return true
}
```

DEMO

Let's play: Scrabble

Scrabble v1

```
var points = [26]int{
    1, 3, 3, 2, 1, 4, 2, 4, 1, 8, 5, 1, 3,
    1, 1, 3, 10, 1, 1, 1, 1, 4, 4, 8, 4, 10,
}

func Score(word string) int {
    sum := 0
    workString := strings.ToUpper(word)

    for _, set := range workString {
        sum += points[set-65]
    }
    return sum
}
```

Scrabble v2

```
func Score(word string) int {
    letterScores := map[rune]int{
        'A': 1, 'E': 1, 'I': 1, 'O': 1, 'U': 1, 'L': 1, 'N': 1, 'R': 1, 'S': 1, 'T': 1,
        'D': 2, 'G': 2,
        'B': 3, 'C': 3, 'M': 3, 'P': 3,
        'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
        'K': 5,
        'J': 8, 'X': 8,
        'Q': 10, 'Z': 10,
    }
    score := 0
    for _, letter := range word {
        score += letterScores[rune(unicode.ToUpper(letter))]
    }
    return score
}
```

Scrabble v3

```
var letterScores = map[rune]int{
    'A': 1, 'E': 1, 'I': 1, 'O': 1, 'U': 1, 'L': 1, 'N': 1, 'R': 1, 'S': 1, 'T': 1,
    'D': 2, 'G': 2,
    'B': 3, 'C': 3, 'M': 3, 'P': 3,
    'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
    'K': 5,
    'J': 8, 'X': 8,
    'Q': 10, 'Z': 10,
}

func Score(word string) int {
    score := 0
    for _, letter := range word {
        score += letterScores[rune(unicode.ToUpper(letter))]
    }
    return score
}
```

Comparing Benchmarks - Benchstat

Comparing Benchmarks - Benchstat

Running benchmarks

```
go test -run none -bench . -benchmem -count=10 > ../v1.txt
go test -run none -bench . -benchmem -count=10 > ../v2.txt
```

Running **benchstat**

```
benchstat v1.txt v2.txt
pkg: github.com/qba73/scrabble

        |    v1.txt    |               v2.txt                |
        |    sec/op    |   sec/op      vs base               |
Score-8   3029.5n ± 1%   298.4n ± 1%  -90.15% (p=0.000 n=10)


        |   v1.txt     |               v2.txt                |
        |  allocs/op   | allocs/op   vs base                  |
Score-8   63.000 ± 0%   9.000 ± 0%  -85.71% (p=0.000 n=10)
```

Benchstat (https://pkg.go.dev/golang.org/x/perf/cmd/benchstat)

Preventing Compiler Optimizations

```
var Res bool

func BenchmarkIsIsogram(b *testing.B) {
    var got bool
    for i := 0; i < b.N; i++ {
        for _, tc := range tt {
            got = isogram.IsIsogram(tc.input)
        }
    }
    Res = got
}
```

There is much more...

- Quiet environment
- Managing benchmark loop
- Managing benchmark timer
- Improving benchmark accuracy

Upcoming book

Get the book when it's out (https://jarosz.dev/article/performance-tuning-and-benchmarking/).

Thank you

Jakub Jarosz
jakub@jarosz.dev (mailto:jakub@jarosz.dev)
https://jarosz.dev (https://jarosz.dev)
@qba73 (http://twitter.com/qba73)