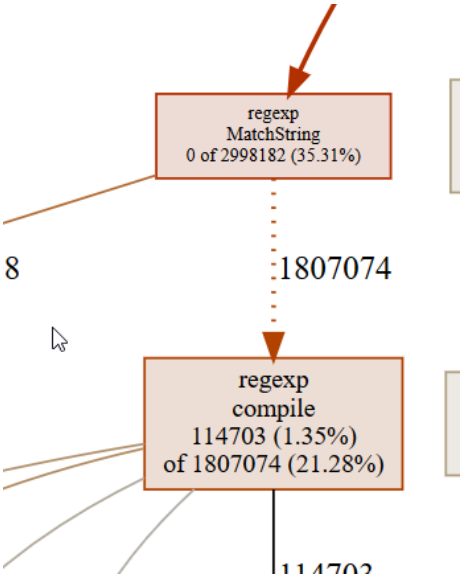
1.Вынесены из функции все регулярки

```
var r  = regexp.MustCompile("@")
var rMSIE  = regexp.MustCompile("MSIE")
var rAndroid  = regexp.MustCompile("Android")
```

чтобы каждый раз их не компилить в функции в цикле. Потому что на графе эта операция занимает много ресурсов. После вынесения занимает небольшую часть, которую даже не видно на графе.



2. Чтобы не накапливать данные в лишнюю переменную foundUsers в цикле, они сразу выводятся.

3. Замена двух повторяющихся одинаковых циклов

```
for _, browserRaw := range browsers {...}
```

одним.

4. Все функции fmt.Fprinln заменены на out.Write(). т.к fmt принимают interface{} и работают с ним с помощью рефлекта

5. Добавлен анмаршалинг с помощью easyjson`а с помощью структуры

```
type User struct {

  Browsers []string
  Company string
  Country string
  Email string
  Job string
```

  Name string
  Phone string
}

6.Цикл считывания юзеров в слайс из файла

for i, line := **range** lines {…}

объединен с циклом итерации по юзерам.

7.Вместо регулярок добавлены функции поиска по строке из пакета strings.

```
goos: windows
goarch: amd64
BenchmarkSlow-4                10         134340750 ns/op        320462288 B/op    284123 allocs/op
BenchmarkFast-4               300           4291751 ns/op         3966056 B/op       8640 allocs/op
PASS
```

Бенчмарк до этого.

```
goarch: amd64
BenchmarkSlow-4                10         143516140 ns/op        320461780 B/op    284122 allocs/op
BenchmarkFast-4               500           3777898 ns/op         3962374 B/op       8440 allocs/op
PASS
ok      _/D_/GitRep/5/99_hw/optimization        4.375s
```

Бенчмарк после этого. Итоговый