Assignments for week 6

Anders Degn Lapiki, Jacob Kjærulff Furberg, Jonas Ishøj Nielsen

01/10/2020

Contents

1 6.1

2 6.2 Dictionary

2.1 6.2.1 - green

Done

2.2 6.2.2 - green

Done

2.3 6.2.3 - green

There are 63 such words.

2.4 6.2.4 - green

Done

2.5 6.2.5 - green

There are 161 such words.

2.6 6.2.6 - green

Didn't bother adding mark 6.
Took time with system.currenttimemilis, the parallel version finished in 45 ms approximately and the non-parallel version finished in 90 ms approximately.
To use mark6 one would have to e.g. instead of printing return count for mark6 to use to prevent dead data.

2.7 6.2.7 - green

```
Min: 1
Max: 24
Average: 9.569126612007494
```

2.8 6.2.8 - Yellow

```
readWords(filename).collect(Collectors.groupingBy(s-> s.length()))
.forEach((s,lst)-> System.out.println(""+s+" "+lst.size()));
```

```
1 52
2 160
3 1420
4 5272
```

```
5 10230
6 17706
7 23869
8 29989
9 32403
10 30878
11 26013
12 20462
13 14939
14 9765
15 5925
16 3377
17 1813
18 842
19 428
20 198
21 82
22 41
23 17
24 5
```

2.9 6.2.9 - yellow

Done

2.10 6.2.10 - yellow

-=2, a=199554, b=40433, c=103440, d=68191, e=235331, f=24165, g=47094, h=64356, i=201032, j=3167, k=16158, l=130463, m=70680, n=158743, o=170692, p=78163, q=3734, r=160985, s=139542, t=152831, u=87353, v=20177, w=13864, x=6932, y=51681, z=8460

2.11 6.2.11 - red

Done, it took 18 seconds on a 4 core computer.

```
readWords(filename)
    .collect(Collectors.groupingBy(s -> letters(s))).values()
    .stream().filter(g -> g.size() > 1).count()
```

2.12 6.2.12 - red

Appears to be way way slower, ours didn't stop computing before we lost patience.

2.13 6.2.13 - red

Done, and it took now 4.5 seconds.

3 6.3 Primes

3.1 6.3.1 - red

Can be seen in `primeNumberTheorem.java`.

```
Arrays.parallelSetAll(arr, i -> isPrime(i) ? 1 : 0);
```

3.2 6.3.2 - red

Can be seen in `primeNumberTheorem.java`.

```
Arrays.parallelPrefix(a, (v1,v2)->v1+v2 );
```

3.3 6.3.3 - red

Can be seen in `primeNumberTheorem.java`.

```
for (int i = N/10; i < N; i += N/10) {
    var ratio = arr[i] / (i/Math.log(i));
    System.out.println(ratio);
}
```

```
0
    NaN
1000000
    1.0844899477790797
2000000
    1.080408961485814
3000000
    1.0778734863718156
4000000
    1.076082563904751
5000000
    1.0751590132527897
6000000
    1.073907637242721
7000000
    1.073235665308496
8000000
    1.0724661949361998
9000000
    1.0719440865480063
```

```
10000000
    1.0711747889618228
```