Kang Shentu – SEC01 (NUID 001569432)

> Big Data System Engineering with Scala Spring 2022

Before Assignment

1. Based on the messages provide by Yiqing Jackie Huang, I downloaded the given page and uploaded to GitHub. Also, I fixed these 3 links.

```
<dl>
    <dt>CSYE7200</dt>
    <dd><a href="http://newton.neu.edu/uploads/32397.pdf">Big Data System Engineering wi
    <dt>INFO6205</dt>
    <dd><a href="http://newton.neu.edu/uploads/31859.pdf">Program Structure and Algorithr
    <dt>INFO6205</dt>
    <dd><a href="http://newton.neu.edu/uploads/33189.pdf">Program Structure and Algorithr
    <dt>Schedule</dt>
    <dd><a href="http://www1.coe.neu.edu/~rhillyard/Schedule.pdf">My weekly schedule Spr
</dl>
```

2. Here is the GitHub Link. Clicked Me

Assignment No. 7

Crawler

wget(u: URL)

```
  def wget(u: URL): Future[Seq[URL]] = {
    // Hint: write as a for-comprehension, using the method
createURL(Option[URL], String) to get the appropriate URL for relative links
    // 16 points.
    def getURLs(ns: Node): Seq[Try[URL]] = (ns \\ "a").map(n => {
      createURL(Option(u), n \@ "href")
    })
```

```
    def getLinks(g: String): Try[Seq[URL]] = {
      val ny = HTMLParser.parse(g) recoverWith { case f => Failure(new
RuntimeException(s"parse problem with URL $u: $f")) }
      for (n <- ny; z <- MonadOps.sequence(getURLs(n))) yield z
    }
    // Hint: write as a for-comprehension, using getURLContent (above) and
getLinks above. You might also need MonadOps.asFuture
    // 9 points.
    for (c <- getURLContent(u); us <- MonadOps.asFuture(getLinks(c))) yield us

  }
```

wget(us: Seq[URL])

```
  def wget(us: Seq[URL]): Future[Seq[Either[Throwable, Seq[URL]]]] = {
    val us2 = us.distinct take 10
    // Hint: Use wget(URL) (above). MonadOps.sequence and Future.sequence are
also available to you to use.
    // 15 points. Implement the rest of this, based on us2 instead of us.
    Future.sequence(us2.map(u => {
      MonadOps.sequence(wget(u))
    }))
  }
```

MonadOps

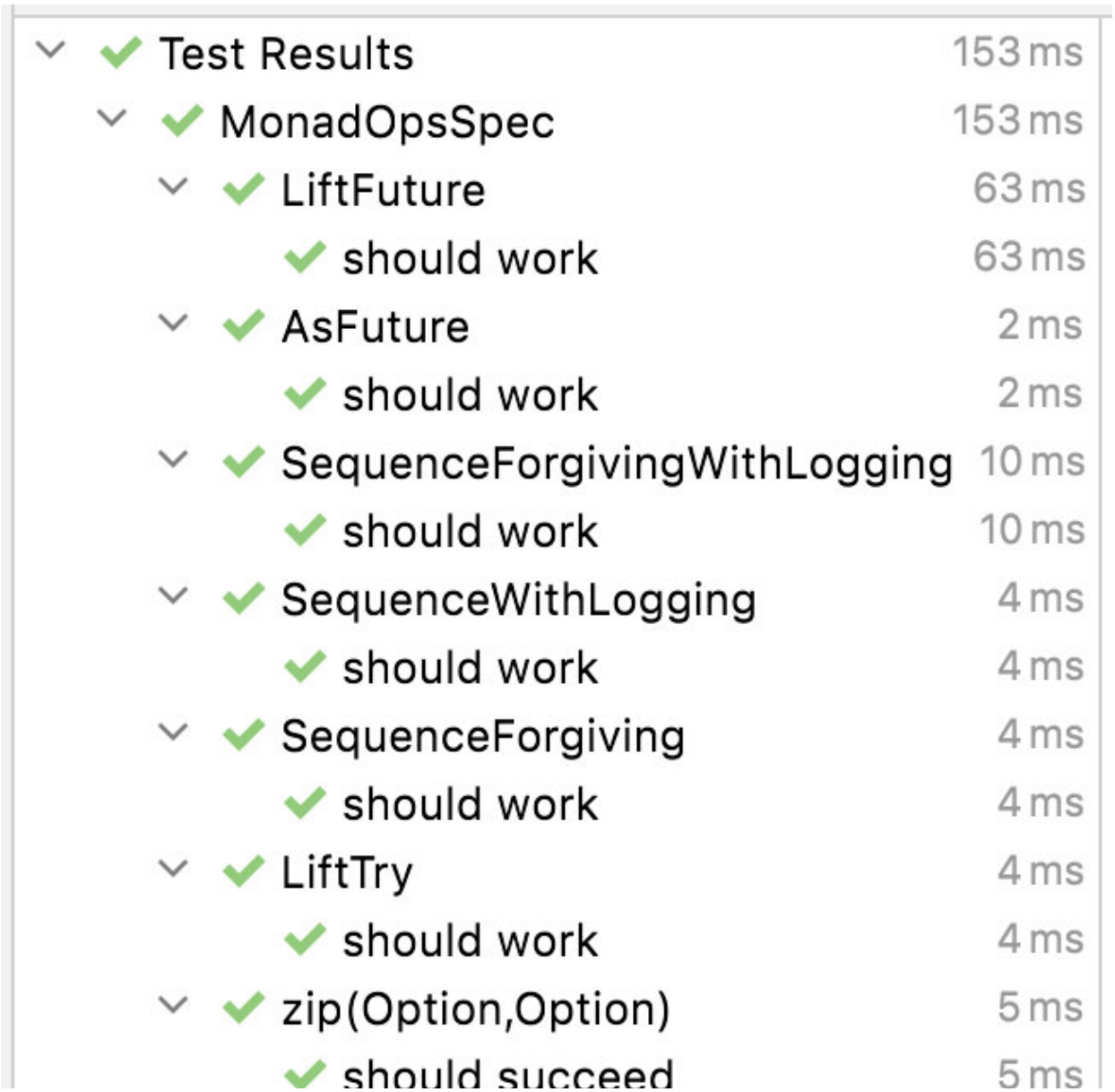
mapFuture(xfs: Seq[Future[X]])

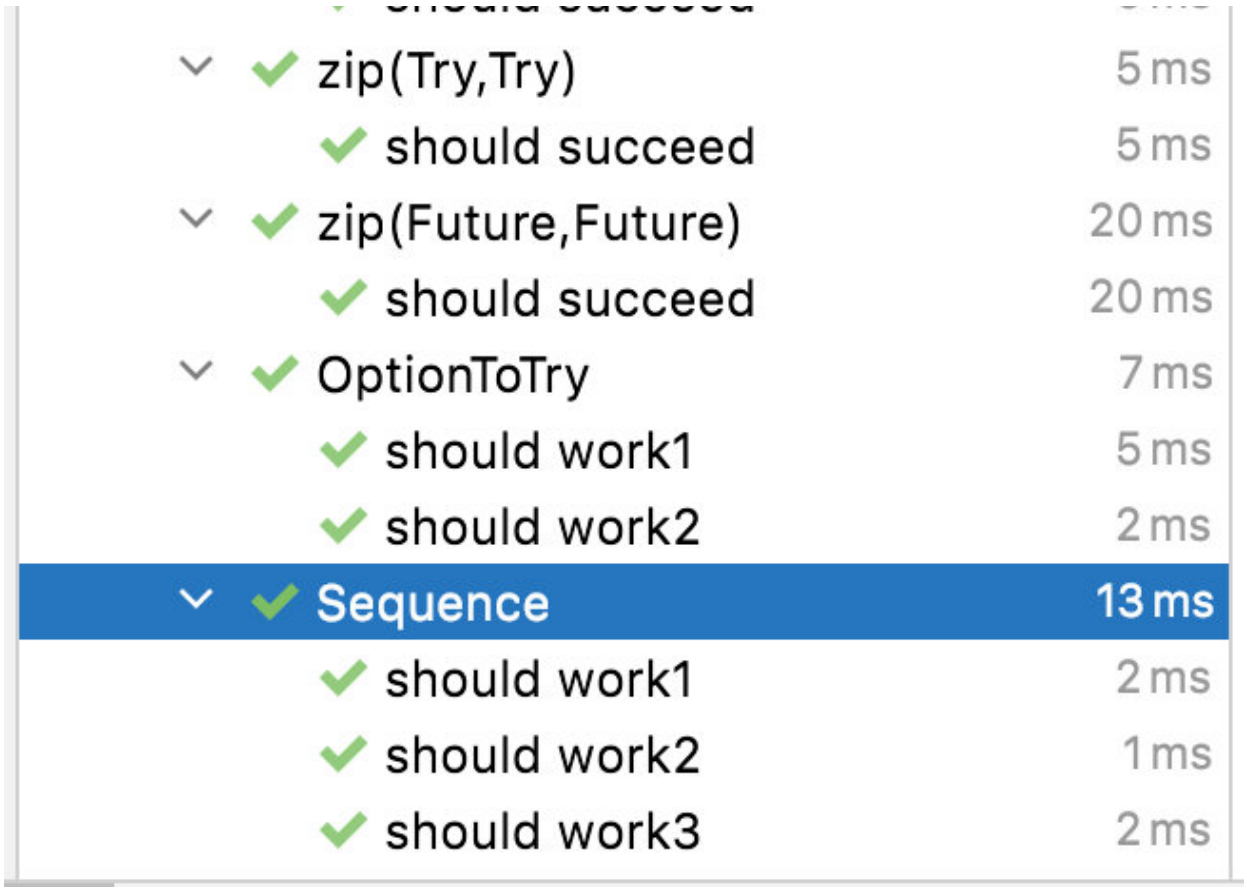
```
  // 6 points.
  def mapFuture[X](xfs: Seq[Future[X]])(implicit executor: ExecutionContext):
Seq[Future[Either[Throwable, X]]] =
    for (x <- xfs) yield sequence(x) // TO BE IMPLEMENTED
```

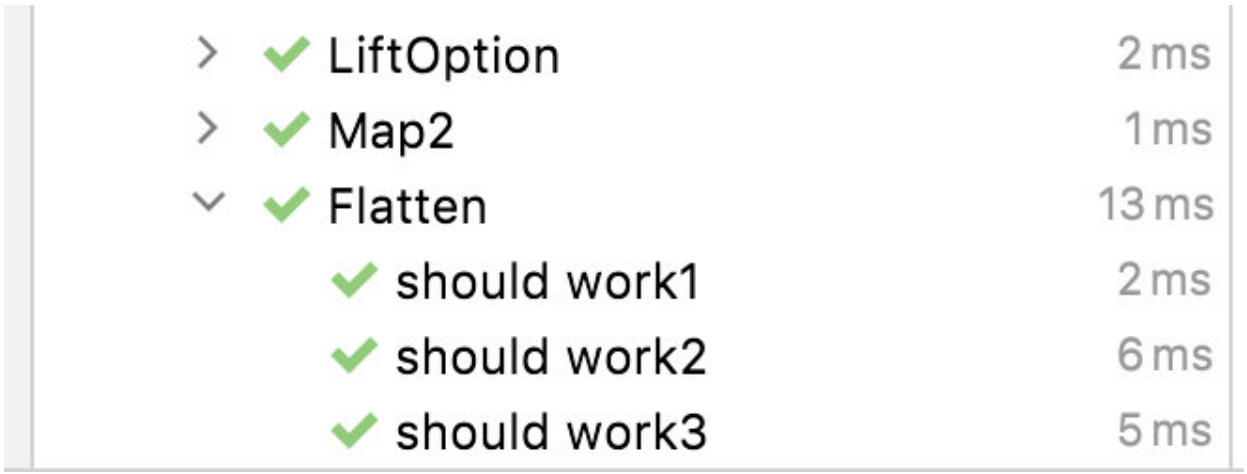
sequence(xe: Either[Throwable, X])

```
  // 7 points.
  def sequence[X](xe: Either[Throwable, X]): Option[X] = xe.toOption // TO BE
IMPLEMENTED
```

Unit Test




zip(Try,Try) 5 ms
should succeed 5 ms
zip(Future,Future) 20 ms
should succeed 20 ms
OptionToTry 7 ms
should work1 5 ms
should work2 2 ms
Sequence 13 ms
should work1 2 ms
should work2 1 ms
should work3 2 ms


LiftOption 2 ms
Map2 1 ms
Flatten 13 ms
should work1 2 ms
should work2 6 ms
should work3 5 ms

Updated on Unit Test

```
val localURL = "https://raw.githubusercontent.com/arron-rgb/CSYE7200/Spring2022/indexSafe

"crawler(Seq[URL])" should s"succeed for $goodURL, depth 2" taggedAs Slow in {
  val args = List(s"$localURL")
  val uys = for (arg <- args) yield Try(new URL(arg))
  val usft = for {us <- MonadOps.sequenceForgiving(uys)} yield WebCrawler.crawler( depth =
  val usf = MonadOps.flatten(usft)
  whenReady(usf, timeout(Span(60, Seconds))) { s => Assertions.assert(s.length == 34) }
}
```

Database SciView