Assignment2

Screenshot of Test Result

文本

描述已自动生成

1. a) In MyLazyList, it uses two cases to distinguish the situation of empty list and nonempty list. At the same time, the scala lazylist uses state to go to two different parts to continue the lazy part, it doesn’t make two cases.

b) The difference is because the MyLazyList extends the LazyListLike. With ListLike, we need to make a judgement to check whether the elements match the situation of LazyList. But the scala LazyList doesn’t need this step.

1. This is to evaluate the lazyTail when it is invoked. We need this function for that we can save the values of the lazyTail so we can use this value later.
2. line 131: case MyLazyList(h, f) => inner(rs :+ h, f())

line 383: def continually[X](x: => X): ListLike[X] = MyLazyList(x, () => continually(x))

line 408: def from(start: Int, step: Int): ListLike[Int] = MyLazyList(start, () => from(start + step, step))

1. There’s no mutable variables and mutable collections in MyLazyList.
2. The zip method is used to make two elements combined to a tuple.

def zip[Y](ys: ListLike[Y]): ListLike[(X, Y)] = ys match {

case MyLazyList(y, g) => MyLazyList((x, y), () => lazyTail() zip g())

case \_ => EmptyList

}

For two streams(not empty), this make the first element from each stream together into a touple (x, y) as the head, and then make the tail of them “zip” one by one to make a new ListLike stream.

1. Because we cannot know the length of a lazylist because it is indefinite. The length of the lazylist is decided by us because we can create new elements in lasylist at any time.