**Scenario:**

add some test coverage to the project so that if any future modifications affect the optimisations, they'll show up as failing tests - because the algorithm is deterministic and running against known test data, this could be as detailed as counting the number of instructions executed for a given set of test inputs. In other words, I'm not looking to measure algorithm performance using timers - I'm interested in actually testing the algorithm's internal behaviour instead of just the output.

how would you apply red->green->refactor when performance is a critical requirement?

1. Write pinning tests to catch regressions, for what you plan to change and other methods that may slow down as a result of your changes.

2. Write a performance test that fails.

3. Make performance improvements, running all tests frequently.

4. Update your pinning tests to more closely pin the performance.

**Write pinning tests**

Create a helper method like this to time what you want to pin.

private TimeSpan Time(Action toTime)

{

var timer = Stopwatch.StartNew();

toTime();

timer.Stop();

return timer.Elapsed;

}

Then write a test that asserts your method takes no time:

[Test]

public void FooPerformance\_Pin()

{

Assert.That(Time(()=>fooer.Foo()), Is.LessThanOrEqualTo(TimeSpan.FromSeconds(0));

}

When it fails (with the actual time elapsed in the failure message), update the time with something slightly more than the actual time. Rerun and it will pass. Repeat this for other functions whose performance you might impact with your changes, ending up with something like this.

[Test]

public void FooPerformance\_Pin()

{

Assert.That(Time(()=>fooer.Foo()), Is.LessThanOrEqualTo(TimeSpan.FromSeconds(0.8));

}

[Test]

public void BarPerformance\_Pin()

{

Assert.That(Time(()=>fooer.Bar()), Is.LessThanOrEqualTo(TimeSpan.FromSeconds(6));

}

**Write a failing performance test**

I like to call this kind of test a "baiting test". It's just the first step of a pinning test.

[Test]

public void FooPerformance\_Bait()

{

Assert.That(Time(()=>fooer.Foo()), Is.LessThanOrEqualTo(TimeSpan.FromSeconds(0));

}

Now, work on performance improvements. Run all the tests (pinning and baiting) after each tentative improvement. If you are successful, you'll see the time going down in the failure output of the baiting test, and none of your pinning tests will fail.

When you are satisfied with the improvements, update the pinning test for the code you changed, and delete the baiting test.