Probabilistic Algorithms

CS4050 – Assignment 3

Ryan Flageolle

**Description**

This assignment we were asked to write 5 separate programs, each of which deal with a probabilistic algorithm, and statistically analyze the results. Below are my findings and data from each program written in Scala.

**Computing Pi**

From my implementation:

The approximation of 𝜋, throwing 1000 darts is: 3.156

The approximation of 𝜋, throwing 10000 darts is: 3.1384

The approximation of 𝜋, throwing 100000 darts is: 3.14528

The approximation of 𝜋, throwing 1000000 darts is: 3.142104

The approximation of 𝜋, throwing 100000000 darts is: 3.14169764

As we can see by throwing more darts in each simulation, we increase both the precision and accuracy of our estimation of pi.

**Testing for Primes**

From my implementation:

The number: 2448, is prime. Tested against 10 numbers.

The number: 2448, is prime. Tested against 100 numbers.

The number: 2448, is not prime. Tested against 1000 numbers.

The number: 2448, is not prime. Tested against 10000 numbers.

The number: 2448, is not prime. Tested against 100000 numbers.

The composite number: 242508, was identified as composite. Tested against 10 numbers.

The composite number: 242508, was identified as composite. Tested against 100 numbers.

The composite number: 242508, was identified as composite. Tested against 1000 numbers.

The composite number: 242508, was identified as composite. Tested against 10000 numbers.

The composite number: 242508, was identified as composite. Tested against 100000 numbers.

Again, as we add more numbers to test against we can see that we can more precisely guess a random prim number as well as identifying whether a large composite number is actually composite.

**Searching Arrays**

From my implementation:

Through 1000 trials - Random Search takes on average 987 attempts to guess the correct value.

From this output we can see that randomly searching the array until the correct value is found on average takes less than 20% of the allotted 5000 attempts.

**Monte Carlo Integration**

From my implementation:

Where f(x) = (cos(3πx)+4sin(x)+5)

Darts (after 100000 trials): The estimated integral of f(x) = 57.373

Mean (after 100000 trials): The estimated integral of f(x) = 57.30056235669802

Trapezoid (after 100000 trials): The estimated integral of f(x) = 57.35688611017637

As we can see using the darts method, we tend to over-estimate our integral, whereas with the mean method we tend to under-estimate our integral.

**8 Queens Problem**

For this problem, the algorithm I implemented worked too quickly to accurately record the time it took to run. Because of this I was unable to determine which value for k is best.

However, I did find that the best value of k should be no larger than k – (0.25 \* k), this is because the time it takes to randomly generate the correct position for the almost full board is extremely large.