UMMC

Jiamu Li & Frank Tang & Edward Wang

July 17, 2022

1 Question 1

Theorem 1. *The probability that no two X-particles are next to each other after n shots is given by*

$$\frac{F_{n+2}}{2^n},$$

where F_n is the nth Fibonacci number.

Proof. The probability we require can be calculated by dividing the total number of ways to arrange the contents of the tube such that there are no consecutive X-particles, by the total number of arrangements of the tube. That is to say:

$$\Pr(\text{No consecutive X-particles}) = \frac{\#\text{Arrangements w/o consecutive X-particles}}{\#\text{Total arrangements}}$$

Claim 1.1. *The number of arrangements with no consecutive X-particles is*

$$\sum_{k=0}^{n} \binom{n-k+1}{k}.$$

Proof. Consider a tube with n particles in it. Let the number of X-particles be equal to k, and the number of Y-particles be equal to $n-k$. Consider the tube without the X-particles, consisting solely of Y-particles in a line:

$$\underbrace{\text{YY...YY}}_{n-k} \tag{1}$$

Now consider the 'gaps' between these Y-particles, indicated by a bar (|):

$$|\text{Y}|\text{Y}|...|\text{Y}|\text{Y}| \tag{2}$$

Notice that there are exactly $n-k+1$ 'gaps'. Clearly, if we were to only place X-particles in the gaps, then there would never be any consecutive X-particles. This can be done in a total of

$$\binom{n-k+1}{k}$$

ways. However, we must consider this for any number of X-particles k, so we arrive at the sum

$$\#\text{Arrangements with no consecutive X-particles} = \sum_{k=0}^{n} \binom{n-k+1}{k}. \qquad \square$$

Claim 1.2. *We claim that*

$$\sum_{k=0}^{n} \binom{n-k+1}{k} = F_{n+2},$$

where F_n is the nth Fibonnaci number.

Proof. Recall that the Fibonnaci numbers are defined as follows:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \qquad n > 1 \end{aligned}$$

Let $f(x) := \sum_{k=0}^{x} \binom{x-k+1}{k}$. It is sufficient to prove that $f(1) = F_3 = 2$, $f(2) = F_4 = 3$, and that $f(n) = f(n-1) + f(n-2)$, which would then imply the result by definition of the Fibonnaci numbers.

It is obvious that $f(1) = \binom{2}{0} + \binom{1}{1} = 2$, which is equal to F_3. Next, $f(2) = \binom{3}{0} + \binom{2}{1} + \binom{1}{2} = 3$. Notice that we define $\binom{n}{k} = 0$ when $n < k$, as it is impossible to choose k things from a set with elements less than k.

We proceed to prove that $f(n) = f(n-1) + f(n-2)$, where $n > 2$.

Using the fact that $\binom{n}{0} = 1$, we rewrite $f(n)$ using Pascal's identity and linearity as

$$f(n) = 1 + \sum_{k=1}^{n} \binom{n-k+1}{k} = 1 + \sum_{k=1}^{n} \binom{n-k}{k} + \sum_{k=1}^{n} \binom{n-k}{k-1}.$$

Next, we simplify, getting

$$\begin{aligned} f(n) &= \sum_{k=0}^{n} \binom{n-k}{k} + \sum_{k=1}^{n} \binom{n-k}{k-1} \\ &= \sum_{k=0}^{n-1} \binom{n-k}{k} + \cancel{\binom{n-n}{k}} + \sum_{k=1}^{n} \binom{n-k}{k-1} \\ &= \sum_{k=0}^{n-1} \binom{n-k}{k} + \sum_{k=0}^{n-1} \binom{n-k-1}{k} \\ &= \sum_{k=0}^{n-1} \binom{n-k}{k} + \sum_{k=0}^{n-2} \binom{n-k-1}{k} + \cancel{\binom{n-(n-1)-1}{n-1}} \\ &= f(n-1) + f(n-2). \end{aligned} \qquad \square$$

Claim 1.3. *The number of total arrangements of a tube with n particles is*

$$2^n.$$

Proof. Each particle in the tube can be either an X-particle or a Y-particle, meaning there are 2 choices for each of the n particles. Hence, there are a total of 2^n arrangements. $\square$

Hence by dividing the number of arrangements where there are no consecutive X-particles by the total number of arrangements, we arrive at the formula

$$\frac{F_{n+2}}{2^n}$$

which gives the desired probability. $\square$

```cpp
#include <chrono>
#include <future>
#include <iostream>
#include <random>
#include <string>
#include <thread>
#include <vector>

using namespace std::chrono;

std::random_device rd;
std::mt19937 rng(rd());

const char particles[] = "XY";

unsigned int num_threads = std::thread::hardware_concurrency();

std::string gen_tube(int length) {
  std::uniform_int_distribution<int> pick(0, 1);
  std::string tube;
  for (int i = 0; i < length; i++) {
    tube += particles[pick(rng)];
  }
  return tube;
}

std::string annihilate(std::string tube) {
  std::string res = "";
  for (char &c : tube) {
    if (res.size() && res.back() == c && c == 'X')
      res.pop_back();
    else
      res.push_back(c);
  }
  return res;
}

bool check_consec_x(std::string tube) {
  for (int i = 0; i < tube.length() - 1; i++) {
    if (tube[i] == tube[i + 1])
      return 0;
  }
  return 1;
}

double prob_consec(int length, int runs) {
  long count = 0;
  for (int i = 0; i < runs; i++) {
    if (check_consec_x(gen_tube(length)))
      count++;
  }
  return count/double(runs);
}

int main() {
  auto start = high_resolution_clock::now();

```

```
  int runs = 10000000;

  // std::vector<std::future<long>> threads;
  //
  // for (int i = 8; i < num_threads+8; i++) {
  //   threads.push_back(std::async(std::launch::async, count_check, i, runs));
  // }
  // for (auto &t : threads) {
  //   std::cout << t.get() << '\n';
  // }

  for (int i = 8; i < 16; i++) {
    std::cout << prob_consec(i, runs) << '\n';
  }

  auto stop = high_resolution_clock::now();
  auto elapsed = duration_cast<milliseconds>(stop - start);
  std::cout << elapsed.count() << "ms\n";
}
```