

MATHEMATICS AND STATISTICS RESEARCH COMPETITION

QUESTION 8

JIAMU LI & FRANK TANG & EDWARD WANG

JULY 24, 2022

CONTENTS

Problem 1	2
Problem 2	5
Problem 3	6
Code Implementation	8

A particle generator is emitting two types of particles (called X and Y) into a long tube. The particles will line up in order after entering the tube. Initially, the tube is empty. At each shot, either an X- or Y-particle is randomly emitted into the tube with equal probability. Different shots are assumed to be independent from each other. Suppose that n shots have been emitted.

PROBLEM 1

What is the probability that no two X-particles are next to each other?

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 n th 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:

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

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

$$|Y|Y|\dots|Y|Y| \tag{2}$$

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

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

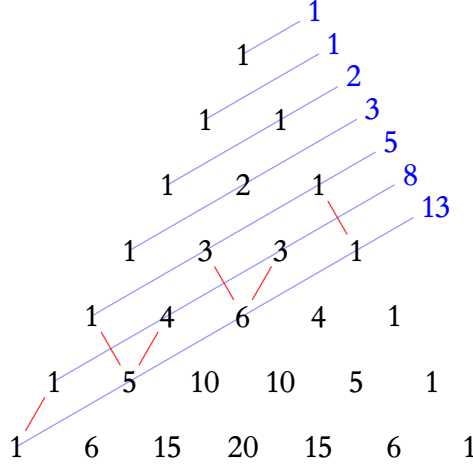


Figure 1

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

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

CLAIM 1.2. *We claim that*

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

where F_n is the n th Fibonacci number.

Proof. Figure 1 showcases a way to obtain the identity from Pascal's triangle. We will present an algebraic proof below.

Recall that the Fibonacci numbers are defined as follows:

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

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 Fibonacci 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

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

Hence $f(n)$ must be equivalent to F_{n+2} . □

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. □

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. □

PROBLEM 2

Compute the average number of particles for $n = 2, 3, 4$.

Can you find the pattern and establish an explicit formula for general n ?

THEOREM 2. *The average number of particles after n shots is*

$$0.75n + 0.25.$$

Proof. Let the average number of particles after n shots be T_n . Obviously, $T_1 = 1$. Next, consider the chance that after firing a shot, the number of particles *doesn't* decrease. This occurs only in the event that the last particle in the tube is an X-particle, and when the particle emitted is also an X-particle. Since both events have a probability 0.5, the probability that both occur is simply 0.25. Hence, the probability that the number of particles *does* increase after firing a shot is $1 - 0.25 = 0.75$. Thus the expected number of particles increases by 0.75 after each shot, giving us the recursion

$$T_{n+1} = T_n + 0.75.$$

Since $T_1 = 1$, we arrive at the formula

$$T_n = 0.75n + 0.25. \quad \square$$

Using this, we can easily compute the average number of particles when $n = 2, 3, 4$:

n	T_n
1	1
2	1.75
3	2.5
4	3.25

PROBLEM 3

Suppose that at each shot, an X-particle is emitted with probability $p \in (0, 1)$.

Under the same assumption as above, when n is very large, do you think the proportion of X-particles in the tube will eventually stabilise at a certain number? Why/why not?

If so, can you compute this number explicitly?

THEOREM 3. *The limiting ratio of X-particles to total particles in the tube, where the probability of emitting an X-particle is $p \in (0, 1)$, as the number of shots approaches infinity is*

$$\frac{p}{1+p}.$$

Proof. We use similar logic as done in Theorem 2 to arrive at a formula for the expected number of particles after n shots when the probability is p .

CLAIM 3.1. *The average number of particles after n shots when the probability of emitting an X-particle is $p \in (0, 1)$ is*

$$(1 - p^2)n + p^2.$$

Proof. Again, let the average number of particles after n shots be T_n . The number of particles *doesn't* increase when firing an X-particle and the last particle is also an X-particle, which has a probability of p^2 of happening. Hence, the probability that the number of particles *does* increase is $1 - p^2$, meaning that the expected number of particles increases by $1 - p^2$ after each shot, and thus we obtain

$$T_{n+1} = T_n + (1 - p^2).$$

Since $T_1 = 1$, the formula for T_n is

$$T_n = (1 - p^2)n + p^2. \quad \square$$

We can now find the expected number of X-particles in the tube. Since the expected number of Y-particles in the tube is simply $(1 - p)n$, because the probability of emitting a Y-particle is $(1 - p)$ at each shot, the expected number of X-particles is

$$\begin{aligned} \#X\text{-particles} &= \# \text{Total particles} - \#Y\text{-particles} \\ &= T_n - (1 - p)n \\ &= (1 - p^2)n + p^2 - (1 - p)n. \end{aligned}$$

We then divide this by the total number of particles to obtain the desired proportion and then take the limit as $n \rightarrow \infty$, getting

$$\lim_{n \rightarrow \infty} \frac{(1 - p^2)n + p^2 - (1 - p)n}{(1 - p^2)n + p^2} = \lim_{n \rightarrow \infty} \frac{(1 - p^2) + \frac{p^2}{n} - (1 - p)}{(1 - p^2) + \frac{p^2}{n}}$$

By the algebraic limit theorem, the terms with a denominator of n become negligibly small, and so we obtain

$$\begin{aligned}\lim_{n \rightarrow \infty} \frac{(1 - p^2) + 0 - (1 - p)}{(1 - p^2) + 0} &= \lim_{n \rightarrow \infty} \frac{p - p^2}{1 - p^2} \\ &= \lim_{n \rightarrow \infty} \frac{p(1 - p)}{(1 + p)(1 - p)} \\ &= \lim_{n \rightarrow \infty} \frac{p}{1 + p}\end{aligned}$$

Since the limit no longer has any terms containing n , we can remove the limit, getting the final result

$$\frac{p}{1 + p}. \quad \square$$

CODE IMPLEMENTATION

```
1  #include <iostream>
2  #include <thread>
3  #include <future>
4  #include <random>
5  #include <string>
6  #include <vector>
7
8  using namespace std;
9
10 random_device rd;
11 mt19937 rng(rd());
12
13 const char particles[] = "XY";
14
15 unsigned int num_threads = thread::hardware_concurrency();
16
17 string gen_tube(int length, double p) {
18     discrete_distribution<int> pick{p, 1 - p};
19     string tube;
20     for (int i = 0; i < length; i++) {
21         tube += particles[pick(rng)];
22     }
23     return tube;
24 }
25
26 string annihilate(string tube) {
27     string output = "";
28     for (int i = 0; i < tube.length() - 1; i++) {
29         if (tube[i] != tube[i + 1] && tube[i] == 'X')
30             output += tube[i];
31         else if (tube[i] != 'X')
32             output += tube[i];
33     }
34     output.push_back(tube.back());
35     return output;
36 }
37
38 bool check_consec_x(string tube) {
39     for (int i = 0; i < tube.length() - 1; i++) {
40         if (tube[i] == tube[i + 1] && tube[i] == 'X')
41             return 0;
42     }
43     return 1;
44 }
45
46 double prob_no_two_consec_after_n(int n, int runs) {
47     int count = 0;
48     for (int i = 0; i < runs; i++) {
49         if (check_consec_x(gen_tube(n, 0.5)))
50             count++;
51     }
52     return double(count) / runs;
53 }
54
55 double average_len_after_n(int n, int runs) {
```



```

56     long long count = 0;
57     for (int i = 0; i < runs; i++) {
58         count += annihilate(gen_tube(n, 0.5)).length();
59     }
60     return count / double(runs);
61 }
62
63 double threaded_avg_len_after_n(int n, int runs) {
64     double avg = 0;
65     vector<future<double>> threads;
66     for (int i = 0; i < num_threads; i++) {
67         threads.push_back(
68             async(launch::async, average_len_after_n, n, runs / num_threads));
69     }
70     for (auto &t : threads) {
71         avg += t.get();
72     }
73     return avg / num_threads;
74 }
75
76 double proportion_x(int runs, double p) {
77     string tube = "";
78     vector<future<string>> threads;
79     for (int i = 0; i < num_threads; i++) {
80         string small_tube = gen_tube(runs / num_threads, p);
81         threads.push_back(async(launch::async, annihilate, small_tube));
82     }
83     for (auto &t : threads) {
84         tube += t.get();
85     }
86     int len = tube.length();
87     return (len - (1 - p) * runs) / len;
88 }
89
90 int main() {
91
92     unsigned long long runs = 1000000;
93
94     cout << "q1\n";
95     for (int i = 1; i <= 10; i++) {
96         cout << "n = " << i << " prob = " << prob_no_two_consec_after_n(i, runs) << '\n';
97     }
98     cout << "q2\n";
99     for (int i = 1; i <= 10; i++) {
100         cout << "n = " << i << " avg length = " << threaded_avg_len_after_n(i, runs) << '\n';
101     }
102     cout << "q3\n";
103     for (double p = 0; p <= 10; p++) {
104         cout << "p = " << p/10 << " proportion = " << proportion_x(runs, p/10) << '\n';
105     }
106
107     return 0;
108 }

```