AIM Qualifying Review Exam: Probability and Discrete Mathematics

January 7, 2023

There are five (5) problems in this examination.

There should be sufficient room in this booklet for all your work. But if you use other sheets of paper, be sure to mark them clearly and staple them to the booklet.

Problem 1

Use the pigeonhole principle on the ordinary long division algorithm to show that any rational number $\frac{p}{q}$, where $p < q$ and q does not divide any power of 10, has a decimal expansion that eventually repeats.

Solution

Ross 8, p. 83.

Consider the long division algorithm on p/q. Note that $p < q$, and by nature of the long division algorithm, each later division is of some non-negative integer remainder $r < q$ by q. A decimal expansion is infinite, but there are only q possible remainders, $0, 1, 2, \ldots, q-1$. By the pigeonhole principle, some remainder repeats. By the determinism of the long division algorithm, that generates a translation symmetry of the decimal expansion, and, so, a repetition.

Mathematical concepts: pigeonhole principle, arithmetic

Problem 2

Solve the following recurrence:

$$\begin{cases} h_n &= 4h_{n-1} + 4^n \\ h_0 &= 3. \end{cases}$$

Solution

Ross 42, p. 262.

The homogeneous relation $h_n = 4h_{n-1}$ has fully-general solution $C4^n$ for constant factor C. To find an inhomogeneous solution, note that, at iteration k, an inhomogeneous 4^k is newly introduced, but this combines with 4^{k-1} from the previous iteration that gets multiplied by the homogeneous coefficient 4, for another 4^k. This suggests that the present value of n inhomogenous contributions are each 4^n, for $n4^n$ in all.

So, try $(C+n)4^n$ and determine C to match the initial condition. At $n=0$, we need $C=3$. So, with $h_n=(3+n)4^n$, we get $h_{n-1}=(2+n)4^{n-1}$ and $4h_{n-1}+4^n=(2+n)4^n+4^n=(3+n)4^n$ is the unique solution.

Mathematical concepts: recurrence relations

Problem 3

(a) Consider the function

$$f(x)=\begin{cases} C(2x-x^3), & 0<x<\frac{5}{2} \\ 0, & \text{otherwise.}\end{cases}$$

Could f be a probability density function, possibly changing the value of $f(0)$ and $f(\frac{5}{2})$ if necessary? If so, find C. (Note: we are asking about f, not the cumulative distribution function $F(t)=\int_{-\infty}^{t} f(s)\,ds$.)

(b) Repeat for the following, in which an exponent has been changed from 3 to 2.

$$f(x)=\begin{cases} C(2x-x^2), & 0<x<\frac{5}{2} \\ 0, & \text{otherwise.}\end{cases}$$

Solution

Ross 5.3, page 227.
We need f to be non-negative and to have integral 1.

(a) For $x\to 0^+$, the term $2x>0$ and dominates, so we need $C>0$. At $x=\frac{5}{2}$, we have $2x-x^3=5-\frac{125}{8}<5-15<0$, so this cannot be a density function.

(b) At $x=\frac{5}{2}$, we have $2x-x^2=5-\frac{25}{4}<5-6<0$, so again the function cannot be a density.

Mathematical concepts: probability density functions

Problem 4

A player makes a sequence of bets that are independent, and each bet results in the player equally likely to win or to lose \$1. Let W denote the net winnings of a player using the strategy to stop playing immediately after their first win. Find:

(a) $P(W>0)$

(b) $P(W<0)$

(c) $E[W]$.

Solution

Ross 7.3, p. 373
The winnings are positive iff the first bet is a winner, with probability $\frac{1}{2}$. Winnings are zero with loss-win, which has probability $\frac{1}{2^2}=\frac{1}{4}$. Winnings are negative with the remaining probability, $1-\frac{1}{2}-\frac{1}{4}=\frac{1}{4}$.

For the expectation, condition on getting the first win after exactly $k\geq 0$ losses and take the infinite sum,

$$\sum_{k\geq 0}(1-k)\left(\frac{1}{2}\right)^{k+1}.$$

So the sum is $S = \frac{1}{2} + \frac{0}{4} - \frac{1}{8} - \frac{2}{16} - \frac{3}{32} - \cdots$

To sum, note that $\frac{S}{2} = \frac{1}{4} + \frac{0}{8} - \frac{1}{16} - \frac{2}{32} - \frac{3}{64} - \cdots$, so $S - \frac{S}{2} = \frac{1}{2} - (\frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots)$, or $S = 1 - (\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots)$. Repeating the computation $S - \frac{S}{2}$, we get $E[W] = S = 0$.

Alternatively (and out of scope), use the linearity of expectation and scrutinize the stopping condition as a stopping time of a martingale. Get that the expectation is the initial value of zero as an immediate conclusion of a more involved framework.

Mathematical concepts: expectation, foundations of probability, generating functions

Problem 5

A directed graph with nodes $\{v_1, v_2, \ldots, v_n\}$ is called *ordered* if

- Each arc goes from a node of lower index to higher index. That is, if $v_i \to v_j$ is an arc, then $i < j$.
- Each node except v_n has an out-arc.

The goal in this question is to solve the problem: *Given an ordered graph G, what is the length of the longest path (in number of arcs) from v_1 to v_n?*

(a) Show that the following algorithm will *not* work, by giving a counterexample graph.

```
set w = v_1
set L = 0
while there is an arc out of w
  choose arc w->v_j for minimal j
  set w = v_j
  set L = L + 1
end while
Return L as the length of the longest path
```

(b) Give a algorithm for this problem that runs in time polynomial in n. Hint: Try dynamic programming.

Solution

Kleinberg and Tardos 3, pp. 314–315.

For a counterexample, take the graph

$v_1 \to v_2 \to v_n$; $v_1 \to v_3 \to v_4 \to \cdots \to v_n$. The above greedy algorithm gives $L = 2$ for $v_1 \to v_2 \to v_n$ while the longest path is $v_1 \to v_3 \to v_4 \to \cdots \to v_n$ with length $L - 2$.

Note that there must be at least one path from v_1 to v_n since we can follow hypothesized out-arcs. But there need not be a path from v_1 to other v_i. A dynamic programming algorithm is as follows:

```
// L(j) stores the longest path length from  v_1 to v_j.
// L(j) = -infty means no path.  Include L(1) = 0.

L(1) = 0
for j = 2 to n
  L(j) = -infty   // provisionally; no path found yet
  for i = 1 to j - 1
                  // consider paths v_1->...->v_i->v_j
    if v_i -> v_j is present and L(i) + 1 > L(j)
    then          // -infty + 1 is still -infty:  no path
      L(j) = L(i) + 1
end if
Return L(n) as the length of the longest path
```

The runtime is $O(n^2)$ because the nested for-loops generate at most n iterations each, and each iteration takes time $O(1)$. The algorithm is correct from the definitions; the key observation is that, to compose the path `v_1->...->v_i->v_j`, we don't need the innards of `v_1->...->v_i`, only the length (including $-\infty$ as a possible length).

Mathematical concepts: Graph algorithms, dynamic programming