Code Jam 2016 - World Finals

Analysis: Family Hotel

Let P(**N**, **K**) denote the probability that room **K** is occupied when the NO VACANCY sign goes up. This is essentially what the problem asks for, except that it is requested in a special format. We will get to that later.

Small input

We first present an algorithm for the small input. There are **N**-1 ways to accommodate the first family, each with probability 1/(**N**-1). Suppose we assign to them adjacent rooms i and i+1. If either is equal to **K**, we are done. Otherwise, **K** falls either in the interval [1, i-1] or in the interval [i+2, **N**]. Take the former case without loss of generality. What happens in the interval [i+2, **N**] is now irrelevant. Hence the problem reduces to finding P(i-1, **K**). Working out the math, we get the following recurrence relation:

- P(1, 1) = 0, and
- P(**N**, **K**) = 1/(**N**-1) [μ(**K**+1 ≤ **N**) + μ(**K**-1 ≥ 1) + $\Sigma_{\mathbf{K}+1 \le i \le \mathbf{N}-1}$ P(i-1, **K**) + $\Sigma_{1 \le i \le \mathbf{K}-2}$ P(**N**-i-1, **N**-**K**+1)], where 1 ≤ **K** ≤ **N**, 2 ≤ **N**, and μ(condition) = 1 if condition holds and 0 otherwise.

Setting aside the issues of precision and the special format requirements of the output, we can use dynamic programming to compute P(**N**, **K**) in time O($\mathbf{N}^3$). This is too slow even for the small case. However, by defining S(n, k) = $\Sigma_{k \le i \le n}$ P(i, k), the recurrence simplifies as follows:

P(**N**, **K**) = 1/(**N**-1) [μ(**K**+1 ≤ **N**) + μ(**K**-1 ≥ 1) + S(**N**-2, **K**) + S(**N**-2, **N**-**K**+1)].

The above recurrence can be used for an O($\mathbf{N}^2$)-time algorithm, which is sufficient for the small input.

Modular arithmetic

Before proceeding to a faster solution for the large input, let us address the output format. Let M = 10^9+7 be the prime specified in the problem statement. For a rational solution p/q, we are asked to output the unique integer y such that 0 ≤ y < M and y*q = p (mod M). In terms of the field (Z_M, +, *), we are simply looking for p/q (mod M) = p * q^{-1}, where the latter (i.e., the inverse of q modulo M) equals q^{M-2} by Euler's theorem (or Fermat's little theorem). This computation requires only logarithmically many operations on integers less than M^2. In fact, all the arithmetic can be carried out in this field, hence there is no need for large integers or any floating-point operations. Note that the division operation in the recurrence is well-defined since we never divide by a multiple of M.

Large input

To solve the large input, we observe the following:

Claim: P(**N**, **K**) = 1 - F(**K**) * F(**N**-**K**+1), where F(q) = 1-P(q, 1) denotes the probability of the left-most room being unoccupied at the end of the day.

Proof: This can be proved rigorously via induction with the base cases being when **K** ∈ {1, **N**}, and using the recurrence relation. A more elegant proof, though, is as follows. Suppose room K is left unoccupied until the end. In that case we essentially have two independent hotels, one

formed with rooms from 1 to **K**, and the other with rooms from **K** to **N**. The first hotel gets those guests that are assigned to rooms between (1,2) and (**K**-1,**K**), and the second hotel gets those guests that are assigned to rooms between (**K**,**K**+1) and (**N**-1,**N**). It's easy to see that those two "sub-streams" of guests that are going to each hotel are also uniformly and independently distributed. The only difference between having such two hotels and one big hotel is that room K could be occupied twice in the two hotel case, but since we're considering the case where it's left unoccupied, this does not happen. Now since the two hotels are independent, we just need to multiply the probabilities that room **K**, which is a border room in both hotels, is left unoccupied. ■

The recurrence for P(**N**, **K**) has a simpler form for **K**=1, as follows:

- P(1, 1) = 0, and
- P(**N**, 1) = 1/(**N**-1) [1 + S(**N**-2, 1)] for **N** ≥ 2.

Since the two one-dimensional arrays can be filled at the same time, there is an O(**N**)-time algorithm to compute all values of P(**N**, 1), hence F(**N**), which leads to a constant-time operation for each P(**N**, **K**) query in the input.

Final remarks

For the curious, the above recurrence yields the following solution for F(q):

$F(q) = \Sigma_{0 \le i \le q-1} (-1)^i \, 1/i!$, for $q \ge 1$.

This has a combinatorial meaning, as well. Consider a permutation π of 1, …, **N**-1, where π_i denotes the time at which we try to assign pair of rooms i and i+1 (if both are vacant). Convince yourself that the number of such permutations producing a certain outcome of room assignments is proportional to that outcome's probability. Now F(q) is equal to the probability that the length of the decreasing sequence at the beginning of π be even. We calculate this via Principle of Inclusion-Exclusion.

Since the claim above is the crux of the solution for large input, it is worth noting that some contestants got the idea by simply looking at the table of 1-P(**N**, **K**) for small values of **N** and **K**.