Exercise 1

1. Flip a fair coin twice. What is the probability that you get two heads (HH)? What is the probability that you get heads followed by tails (HT)? Are these probabilities the same?
2. Flip a fair coin repeatedly until you get heads and tails in a row (HT). What is the probability that it takes n flips to win?
3. Flip a fair coin repeatedly until you get two heads in a row (HH). What is the probability that it takes n flips to win? (*suggestion*: go all the way up to $n = 8$ before making conclusions).
4. Based on the answers to points 2 and 3, is the probability of a *large* value of n equal in the two cases? If it is not, which probability is the highest?
5. Player A and B play the following game: they flip a coin repeatedly until either HH occurs (A wins) or HT occurs (B wins). Is the game fair (i.e., are the two players equally likely to win)?

Exercise 2

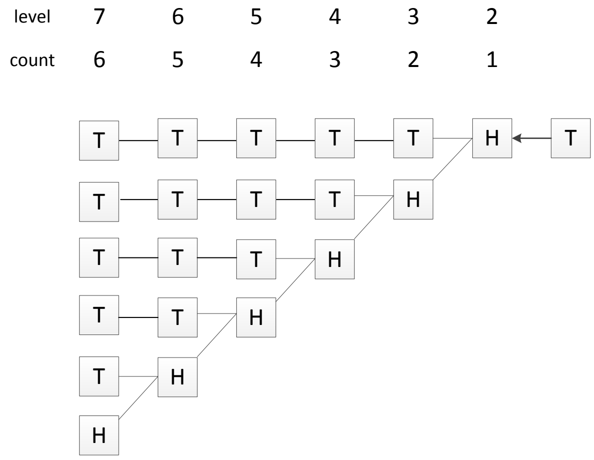
Consider a system where *messages* arrive (exponentially, at a rate λ) and *packets* are buffered and served (exponentially, at a rate μ). Each message carries two packets. The system has enough memory to store two packets, and will reject a message unless it can store both packets.

1. Model the system as a queuing system and draw the CTTC (or transition-rate diagram)
2. Compute the stability condition, the SS probabilities and the mean number of packets in the system
3. Compute the probability that a message is lost, and, from the latter, the mean *rate* of accepted packets
4. Compute the system throughput (in packets per second)
5. Compute the z-transform of the number of packets in the system and, using the latter, the mean and the variance of the number of packets in the system.

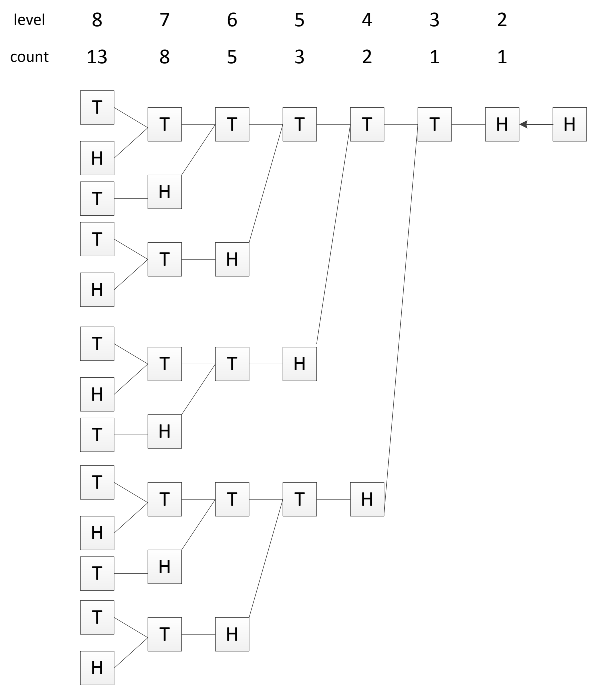
Exercise 1 - Solution

1. The two probabilities are obviously the same, and each one is $\left(\frac{1}{2}\right)^2 = \frac{1}{4}$

2. This is clearly a uniform probability model. The number of n-sequences is 2^n, whereas the number of *good* n-sequences can be computed by constructing a tree. The topmost two levels of the tree are HT, and every time a level j is added, only feasible sequences are generated (i.e., you only add a child T to a parent T, whereas you add both children H and T to a parent H). The tree is in the figure besides. The number of *good* n-sequences increases by one at each step, i.e., it grows linearly and is equal to $n-1$. Hence:
$P\{N_{HT} = n\} = \frac{n-1}{2^n}, n \geq 2$



3. We repeat the same argument as before and find that the number of *good* n-sequences is different. The highest two levels of the tree are HH, and every time a level j is added, you only add a child T to a parent H, whereas you add both children H and T to a parent T. The tree is in the figure below. By counting the number of nodes at each level, one immediately gets that the number of *good* n-sequences is the $n-1$ number in the Fibonacci sequence S_{n-1}. Therefore:
$P\{N_{HH} = n\} = \frac{S_{n-1}}{2^n}, n \geq 2$



4. Given the above, we can observe that $S_{n-1} > n-1$ starting from $n = 6$ (The fact that the Fibonacci sequence is superlinear is well known). Therefore, the probability of observing long sequences is *larger* if you bet on HH.

5. Counterintuitively, the game *is* fair, despite the fact that you observe longer sequences more often with HH than with HT. This can be proved by observing that the *only* n-sequence that can go on indefinitely without either player winning is {TT...TT} (every other sequence has at least one H in the middle, hence leads to one of the two winning). As soon as *one* H appears, the winner is decided by the next flip. So, there is only *one* sequence that leads to the decisive flip, and it is {TTT…TT}H. After that, both players have the same 50% chance to win.

Exercise 2 – solution

It is expedient to use the number of *packets* as a state characterization. This way, the system has a finite memory, equal to 2, and only admits arrivals in state 0. The CTTC is the following:

The SS equations are:

$$p_0 \cdot \lambda = p_1 \cdot \mu$$
$$p_1 \cdot \mu = p_2 \cdot \mu$$
$$p_2 \cdot \mu = p_0 \cdot \lambda$$

And the normalization condition is:

$$p_0 + p_1 + p_2 = 1$$

Using two out of three SS equations (one is linearly dependent on the other two) and the normalization, one straightforwardly obtains:

$$p_1 = p_2 = \frac{\lambda}{2\lambda + \mu}$$
$$p_0 = \frac{\mu}{2\lambda + \mu}$$

The system is always stable, as are all a finite-memory ones. The mean number of packets in the system is:

$$E[N] = 0 \cdot p_0 + 1 \cdot p_1 + 2 \cdot p_2 = \frac{3\lambda}{2\lambda + \mu}$$

The loss probability is the probability that the system is in states 1 or 2. Therefore, it is:

$$p_L = p_1 + p_2 = \frac{2\lambda}{2\lambda + \mu}$$

The rate at which the system accepts packets is instead:

$$\lambda_{pkt} = 2 \cdot \lambda \cdot (1 - p_L) = \frac{2\lambda \cdot \mu}{2\lambda + \mu}$$

The throughput (in packets per second) is obviously $\gamma = \lambda_{pkt}$. The definition yields:

$$\gamma = \mu \cdot (p_1 + p_2) = \mu \cdot \frac{2\lambda}{2\lambda + \mu}$$

Which is obviously the same expression.

By definition, we have $P(z) = \sum_{k=0}^{+\infty} p_k \cdot z^k$, i.e.:

$$P(z) = \frac{\mu}{2\lambda + \mu} + (z + z^2) \cdot \frac{\lambda}{2\lambda + \mu} = \frac{\lambda \cdot z^2 + \lambda \cdot z + \mu}{2\lambda + \mu}$$

We compute:

$$P'(z) = \frac{2\lambda \cdot z + \lambda}{2\lambda + \mu}, \qquad P''(z) = \frac{2\lambda}{2\lambda + \mu}$$

And we know from the theory that:

$$E[N] = P'(1) = \frac{3\lambda}{2\lambda + \mu}$$

$$Var(N) = P''(1) + P'(1) - P'(1)^2 = \frac{2\lambda}{2\lambda + \mu} + \frac{3\lambda}{2\lambda + \mu} - \left(\frac{3\lambda}{2\lambda + \mu}\right)^2 = \frac{\lambda^2 + 5\lambda \cdot \mu}{(2\lambda + \mu)^2}$$