Cpt S 515 Homework #4

No late homework!

1. I have k, for some k, water tanks, $T_1, \cdots, T_k$ (which are identical in size and shap), whose water levels are respectively denoted by nonnegative real variables $x_1, \cdots, x_k$. Without loss of generality, we assume that x_i equals the amount of water that is currently in T_i. Initially, all the tanks are empty; i.e., $x_i = 0$; $1 \leq i \leq k$. I have m pumps $p_1, \cdots, p_m$, that pump water into tanks. More precisely, a pump instruction, say, P_{A,c_1,c_2}, where $A \subseteq \{T_1, \cdots, T_k\}$, is to pump the same amount of water to each of the tank T_i with $i \in A$ (so water levels on other tanks not in A will not change), where the amount is anywhere between c_1 and c_2 (including c_1 and c_2, of course we have assumed $0 \leq c_1 \leq c_2$). For instance, $P_{\{T_2,T_5\},1.5,2.4}$ means to pump simontaneously to T_2 and T_5 the same amount of water. However, the amount can be anywhere between 1.5 and 2.4. Suppose that we execute the instruction twice, say:

$P_{\{T_2,T_5\},1.5,2.4}$;
$P_{\{T_2,T_5\},1.5,2.4}$.

The first $P_{\{T_2,T_5\},1.5,2.4}$ can result in 1.8 amount of water pumped into T_2 and T_5, respectively, and the second $P_{\{T_2,T_5\},1.5,2.4}$ can result in 2.15 amount of water pumped into T_2 and T_5, respectively. That is, the amount of water can be arbitrary chosen inside the range specified in the instruction, while the choice is independent between instructions.

Now, let M be a finite state controller which is specified by a directed graph where each edge is labeled with a pump instruction. Different edges may be labedl with the same pump instruction and may also be labeled with different pump instructions. There is an initial node and a final node in M. Consider the following condition $Bad(x_1, \cdots, x_k)$:

$$x_1 = x_2 + 1 = x_3 + 2 \wedge x_3 > x_4 + 0.26.$$

A walk in M is a path from the initial to the final. I collect the sequence of pump instructions on the walk. If I carefully assign an amount (of water pumped) for each such pump instruction and, as a result, the water levels $x_1, \cdots, x_k$ at the end of the sequence of pump instruction satisfy $Bad(x_1, \cdots, x_k)$, then I call the walk is a bad walk. Such a walk intuitively says that there is an undesired execution of M.

Design an algorithm that decides whether M has a bad walk. (Hint: first draw an example M where there is no loop and see what you can get. Then, draw an M that is with a loop and see what you get. Then, draw an M that is with two nested loops and see what you get, and so on.)

2. The word *bit* comes from Shannon's work in measuring the randomness in a fair coin. However, such randomness measurement requires a probability distribution of the random variable in consideration. Suppose that a kid tosses a dice for 1000 times and hence he obtains a sequence of 1000 outcomes

$$a_1, a_2, \cdots, a_{1000}$$

where each a_i is one of the six possible outcomes. Notice that a dice may not be fair at all; i.e., the probability of each outcome is not neccessarily $\frac{1}{6}$. Based on the sequence only, can you design an algorithm to decide how "unfair" the dice that the kid tosses is.

3. In below, a sequence is a sequence of event symbols where each symbol is drawn from a known finite alphabet. For a sequence $\alpha = a_1 \cdots a_k$ that is drawn from a known finite set S of sequences, one may think it as a sequence of random variables $x_1 \cdots x_k$ taking values $x_i = a_i$, for each i. We assume that the lengths of the sequences in the set S are the same, say n. In mathematics, the sequence of random variables is called a stochastic process and the process may not be i.i.d at all (independent and identical distribution). Design an algorithm that takes input S and outputs the likehood on the process being i.i.d.

4. Let G_1 and G_2 be two directed graphs and v_1, u_1 be two nodes in G_1 and v_2, u_2 be two nodes in G_2. Suppose that from v_1 to u_1, there are infinitely many paths in G_1 and that from v_2 to u_2, there are infinitely many paths in G_2 as well. Design an algorithm deciding that the number of paths from v_1 to u_1 in G_1 is "more than" the number of paths from v_2 to u_2 in G_2, even though both numbers are infinite (but countable).