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).

Assignment 4

Sheryl Mathew

November 03, 2018

1. Question 1

Step 1: In Depth First Search, find all the simple paths in the directed graph specified by the finite state controller M by marking all the nodes that we visit to ensure that we do not traverse cycles.

Step 2: If there is a simple path from the first to the last node which satisfies the bad condition, then there is a bad walk in the graph

Step 3: Find all the SCC's using Tarjan's algorithm and find the simple cycles from the obtained SCC's

Step 4: Find the output = min(1=2+…+n) based on all the given constraints using Linear Programming using Simplex method

Step 5: If the output is zero, then there is a bad walk in the graph. If the output is not zero then there is no bad walk in the graph

2. Question 2

Step 1: Construct a table which will hold the occurrence of each value of a die. For each throw of the die enter which value has occurred. Perform the experiment for 1000 times.

For example, if in the first throw 5 occurred the table entry will be as follows

1	2	3	4	5	6
0	0	0	0	1	0

Step 2: Count each column of the table and store the value in $d_1, d_2 \ldots d_6$. Let D be the total number of rolls i.e. $D = d_1 + d_2 + \cdots d_6$

Step 3: Find the expected number of times each side should come up i.e. the total number of rolls divided by number of sides. $d_{exp} = \frac{D}{6}$

Step 4: Calculate the below for each side k of the die where k = 1 to 6:

$$x_k^2 = \frac{(d_k - d_{\exp})^2}{d_{exp}}$$

Step 5: Sum all the above values to get: $x = x_1^2 + x_2^2 + \cdots + x_6^2$

Step 6: From the Critical Values of Chi Square distribution look up the value for v = 5

Upper-tail critical values of chi-square distribution with ν degrees of freedom

	Probability less than the critical value				
ν	0.90	0.95	0.975	0.99	0.999
1	2.706	3.841	5.024	6.635	10.828
2	4.605	5.991	7.378	9.210	13.816
3	6.251	7.815	9.348	11.345	16.266
4	7.779	9.488	11.143	13.277	18.467
5	9.236	11.070	12.833	15.086	20.515
6	10.645	12.592	14.449	16.812	22.458

Step 7: He we select 5% level of significance so the critical value will be 11.07. So if the value of $x \leq 11.07$ then the die is fair else the die is not. To measure how unfair the die is we calculate $|x - 11.07|$

3. Question 3

Step 1: Find all the possible sequences of S using a permutation algorithm. For every sequence found update totalSequence variable by 1

Step 2: Perform chi square test by comparing each permutation sequence with the original sequence. If the value found from the chi square test is less than the critical value then increment verificationCount variable by 1.

Step 3: If for a sequene the verificationCount =2 which signifies that the sequence is both independent and identically distributed than increment isIidCount variable by 1

Step 4: After performing step 2 and 3 for all the sequences from Step 1, the likelihood of being IID is $Likelihood = \frac{isIidCount}{totalSequenceCount}$

Reference: IID Testing in SP 800 90B [Meltem Sonmez Turan]

4. Question 4

Step 1: Construct an Adjacency matrix A_1 for G_1.

Step 2: Calculate the eigen values for A_1. Let them be $\lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$

Step 3: We can count the total number of paths of length k by considering $1^T A_1 1$ where 1 is an all 1s vector of size n. Let ϕ_i be the eigen vector associated with λ_i. Let a_i be a constant. Then the total number of paths of length k is

$$(\sum_i a_i \phi_i^T)\, A_1\, (\sum_i a_i \phi_i) = (\sum_i a_i \phi_i^T)\, (\sum_i a_i \lambda_i^k \phi_i) = (\sum_i a_i^2 \lambda_i^k)$$

Step 4: We can assume that the graph has an edge which implies that $\lambda_n > 0$.

$$\lim_{k\to\infty} \frac{1^T A_1 1}{\lambda_n^k} = \lim_{k\to\infty} \frac{a_i^2 \lambda_i^k}{\lambda_n^k} = a_n^2$$

Step 5: From Perron-Frobenius theorem we can assume that $|\lambda_i| < \lambda_n$ for $i \neq n$. This shows that λ_n gives the number of walks of length k.

Step 6: Repeat Steps 1 to 5 for G_2.

Step 7: For G_1we get λ_1 and for G_2we get λ_2. If $\lambda_1 > \lambda_2$ then the number of paths in G_1 is greater than the number of paths in G_2

Reference: Eigenvalues and Structures of Graphs [Steven Kay Butler]