CS200 - Problem Set 11

1. Randomized search without replacement is a popular randomized algorithm. The algorithm is as follows. You have an unordered array A of elements, and you would like to find the element x. In other words, you'd like to find the index i where $A[i] = x$ (if x is in the array.) At the first round of the algorithm, you randomly pick an index i_1 and check if $A[i_1] = x$. If it is, you return i_1. If not, you put i_1 into an excluded list, and never check that index again. In the next round, you pick an index i_2 at random among those that have not been excluded and see if $A[i_2] = x$. If it is, you return i_2. If not, you put i_2 into the excluded list, and never check that index again. Repeat until you find x, or have checked all elements.

   (a) Suppose you have an array of length 3 of integers, the number you are looking for is at position 3 (index value 3). Create a tree of possibilities to describe all elements of the sample space and use the tree to describe the sample space.

   (b) What is the probability of each element of the sample space?

   (c) What is the probability that you have *at least* j rounds of the algorithm for $j = 1, 2, 3$?

   (d) We can instead draw a tree of possibilities where we don't include all possible outcomes, bust instead just include two outcomes for each round: either we find x or we don't find x in that round. Redraw the tree, and use it to and figure out the probability that you have at least j rounds, for $j = 1, 2, 3$. (It should be the same as before.)

2. **[6 points]** Suppose you have a weighted coin such that the probability of heads is 0.3 and the probability of tails is 0.7.

   (a) What is the sample space of 10 flips? (Please do not draw a tree, instead simply describe the set.)

   (b) What is the probability of getting at least 3 heads, if you flip the coin 10 times? In this case, drawing out the full tree would be too much work, but you can think about how many elements of the sample space we care about using counting rules, and then calculate the probability of each of those outcomes as if you had constructed the full tree.

3. **[6 points]** Let $S = \mathbb{Z} \times \mathbb{Z}$. Let $R \subseteq S \times S$ be the equivalence relation $R = \{((a_1, b_1), (a_2, b_2)) : a_1 \times b_1 = a_2 \times b_2\}$. Please describe an equivalence class of R using set-builder notation.

4. **[11 points]** Consider an undirected, unweighted graph G with no self-loops on n vertices where every vertex has degree at least $n/2$. Prove that there is a path from every vertex to every other vertex. In other words, prove that the graph is connected, and not split up into disconnected components.

5. **[6 points]** In class, we showed that $T(n)$ runtime of an algorithm with recurrence relation $T(1) = O(1)$, $T(n) = aT(n/b) + O(n^d)$ is

$$T(n) = n^d \sum_{k=0}^{\log_b n} \left(a/b^d\right)^k. \tag{1}$$

Use the formula for a geometric series to evaluate this function, and then use big-O notation to get an expression for three cases: $a = b^d$, $a > b^d$ and $a < b^d$.