CS525DA
HW#4

DUE: Tuesday, October 30

1. (6 points) You want a cup of coffee, your department has a policy that the person taking the last cup must make a new pot, there's one cup of coffee left in the pot, and you **don't** want to make a new pot (you feel that you've made more than your share of coffee). Instead, you adopt the *Vulture Strategy* of returning every 5 minutes to test if someone else has made a new pot. Your testing (carried out while observing others make coffee) reveals a probability p=1/10 of a new pot being made (by someone besides you) in any 5 minute interval, independent of how long the pot has been empty.
(a) What is the expected amount of time you will wait until you can drink coffee, and what is the variance of this value?
(b) What is the probability of having to wait more than 1 hour for your coffee?

2. (8 points) An instance of the NP-hard *Minimum Cover Problem* has input $U=\{1,2,...,n\}$ and a set $S=\{S_1,S_2,...,S_m\}$ of subsets of U satisfying $\bigcup_{S_i \in S} S_i = U$. The output is a minimal number of members of S whose union is U. That is, we seek $T \subseteq S$ of minimum cardinality such that $\bigcup_{S_i \in T} S_i = U$. For example, if $U=\{1,2,3,4,5,6\}$ and $S=\{\{1,4,6\},\{1,6\},\{1,4,5\},\{1,2,5\},\{3\}\}$, then a minimum cover is $T=\{\{1,4,6\},\{1,2,5\},\{3\}\}$.

Consider the *Greedy Algorithm* which selects elements of S with maximal overlap with uncovered elements of U

$T = \varnothing$
while $U \neq \varnothing$ **do**
  Select an $S_i \in S$ to maximize $S_i \cap U$
  $S \leftarrow S \setminus \{S_i\}$
  $T \leftarrow T \cup \{S_i\}$
  $U \leftarrow U \setminus S_i$

(*a*) Show that the *Greedy Algorithm* is not guaranteed to give an optimal answer.
(*b*) Assume that if the optimal answer uses m_{opt} sets. Show that after the greedy algorithm chooses m_{opt} sets for T then $\frac{\left|\bigcup_{S_i \in T} S_i\right|}{n} \geq 1 - \frac{1}{e}$. That is, if U can be covered by the union of m_{opt} sets of S, then after the *Greedy Algorithm* puts m_{opt} sets in T, it may not have covered all of U, but it must have covered at least a fraction 1-1/e of U. Hint: $1/e \geq \left(1 - 1/m_{opt}\right)^{m_{opt}}$.

3. (7 points) For $n \geq 1$, the *n*-queens problem asks for $Q(n)$, the number of ways to place n nonattacking queens on an $n \times n$ chessboard, where two queens attack each other if they are on the same row, column or diagonal. The URL

http://www.research.att.com/cgi-bin/access.cgi/as/njas/sequences/eisA.cgi?Anum=000170

shows the first few values of $Q(n)$ to be:

n	$Q(n)$
1	1
2	0
3	0
4	2
5	10
6	4
7	40
8	92
9	352
10	724
11	2680
12	14200
13	73712
14	365596
15	2279184
16	14772512
17	95815104
18	666090624
19	4968057848
20	39029188884
21	314666222712
22	2691008701644
23	24233937684440

Write and test a probabilistic program to estimate $Q(n)$ for arbitrary n. Show and discuss the results (accuracy and variance) of your tests.

CS525DA
HW#4 SOLUTIONS

1. (a) Let p=1/10, and let random variable X denote the number of slots you wait until finding a fresh new pot. That is, the probability of using exactly k slots, $k \geq 1$, is $(1-p)^{k-1} p$. So X is distributed according to a geometric distribution with probability of success p. The computation of $E[X]$ may be found in many probability texts (for example, in equation C.31 on pg. 1112 of our text), but I like the recurrence which uses the memoryless aspect of geometrically distributed random variables. That is, with probability p only 1 slot is used, and with probability 1-p a new pot is not made the first interval and we must expect to wait $1+E[X]$ time slots. So $E[X] = p*1+(1-p)(1+E[X])$. Solving for $E[X]$ yields $E[X]=1/p$, and on average you will wait 50 minutes for a new pot of coffee.
(b) The probability of waiting more than 1 hour is the probability of starting with 12 failures, which is $(1-p)^{12} = 0.9^{12} =$ 0.2824295365

By the way, many sources derive $V[X] = (1-p)/p^2 = 90$, and the standard deviation is $\sqrt{V[X]} = \sqrt{90} = 9.486832980$

2. (*a*) The *Greedy Algorithm* fails for input instance {1,2,3,4,5,6} and $S = \{\{1,2,3,4\},\{1,2,5\},\{3,4,6\},\{5\},\{6\}\}$. The optimal answer is $\{\{1,2,5\},\{3,4,6\}\}$, but the greedy algorithm would construct a T with three members.
(*b*) There must be some member of S which at least the fraction $1/m_{opt}$ of any subset of U (because some set of at most m_{opt} members of S covers the subset of U). After m_{opt} iterations, the uncovered fraction of U is at most $\left(1-\frac{1}{m_{opt}}\right)^{m_{opt}}$, and the covered fraction is at least $1-\left(1-\frac{1}{m_{opt}}\right)^{m_{opt}} \geq 1-\frac{1}{e}$.