Extra Practice Problems for Quiz 2

Problem-1: True/False Questions

(a) A Nash Equilibrium is a state of a game such that a player can improve his or her expected utility by changing his/her strategy if and only if every other player can also improve his/her expected utility by changing his/her strategy.

Solution: False: It is not necessary that *all* other players must change his/her strategy)

(b) The matrix [[(1,-1), (-1, 1)], [(-1, 1), (1, -1)] has no pure-strategy or mixed-strategy Nash equilibria.

Solution: False: It has no pure-strategy Nash equilibria. However, by symmetry it clearly has a mixed-strategy Nash equilibrium $(\frac{1}{2}, \frac{1}{2})$.

(c) Given a 2 player zero sum game, if player A declares a deterministic strategy, and then player B responds with a deterministic strategy, the payoffs will be the same as if player B declares a deterministic strategy, and then player A responds with a deterministic strategy.

Solution: False: Analysis applies to randomized strategy.

(d) Consider the 2-player game where each player has 3 possible strategies and has the following payoff matrix:

Player B	Player A: 1	Player A: 2	Player A: 3
a	0 / 0	+1 / -1	-2 / +2
b	-1 / +1	0 / 0	+1 / -1
c	+2 / -2	-1 / +1	0 / 0

The mixed strategy where

i. A chooses one of the strategies 1, 2, 3 with probability 1/3 each.

ii. B chooses one of the strategies a, b, c with probability 1/3 each.

is a Nash equilibrium of the described game.

Solution: **False.** With these mixed strategies, the payoff for each players is chosen uniformly at random from the entries of the 3 by 3 payoff matrix. Due to symmetry, the expected payoff for each player is 0. However, player B has incentive to change their strategy to playing (a) with probability 1. (Similarly, A has incentive to change their strategy to playing (1) with probability 1.) In this case, the expected payoff is: $\frac{1}{3} \cdot 0 + \frac{1}{3} \cdot (-1) + \frac{1}{3} \cdot 2 = \frac{1}{3} > 0$

Problem-2: Monte Carlo Advantage

Alyssa P. Hacker is a space scientist at SpaceX where she works with various collections of space rocks. Her task is to figure out whether or not a given space rock contains any evidence of organic material. Unfortunately, the state of the art technology for detecting organic material in space rocks is not entirely reliable. This device is a bulky box where Alyssa can place a rock and get an output of whether or not the rock contains organic material.

The box has the following properties. If the rock contains organic material, it outputs YES with certainty. However if the rock does not contain organic material, it only outputs NO with probability $1/5$. Alyssa was discouraged at first, given that the box is not even reliable enough for a Monte-Carlo algorithm. But she quickly realized the box was actually much more capable than she thought.

(a) Using Alyssa's box, give a Monte-Carlo algorithm for detecting whether a rock contains organic material. Your algorithm should be correct with probability strictly greater than $\frac{1}{2}$. For full credit, your algorithm should make $O(1)$ queries to the box.

Solution: Place rock in box and if it outputs NO, then return NO.

If it outputs YES. Return NO with probability p and YES with probability $1 - p$ where we give a specific value for p.

With $p = 4/9$, we show that the algorithm always returns the correct answer with probability $5/9 > 1/2$. We do this analysis in cases.

Case 1: The answer is YES. In this case we know the box will never output NO. Therefore we output YES with probability $1 - p$.

Case 2: The answer is NO. In this case, the box outputs NO with probability $1/5$ and YES with probability $4/5$. Hence the probability that we output NO is $1/5 + (4/5) \cdot p$. We want this value to be at least a half, and for simplicity, we can set it equal to $1 - p$. We can solve $1/5 + (4/5) \cdot p = 1 - p$ to get $p = 4/9$.

Hence, we always output the correct answer with probability $1 - p = 5/9$ which is strictly greater than $1/2$, and only use 1 query to the box.

Why did we choose this value for p? We need a value $p < \frac{1}{2}$ so that we answer the first case (where the correct answer is YES) with probability $1 - p > 1/2$. But we also need that adding the extra $\frac{1}{5}$ advantage makes it greater than $\frac{1}{2}$ so that we answer the NO case with probability $> \frac{1}{2}$. That is we want $p < \frac{1}{2}$ and $\frac{1}{5} + \frac{4}{5} \cdot p > \frac{1}{2}$ which implies $p > \frac{3}{8}$. So any value $\frac{1}{2} > p > \frac{3}{8}$ works. Using $\frac{1}{5} + \frac{4}{5} \cdot p = 1 - p$ gives us the same probability of success in either case to make our analysis simpler.

(b) Alyssa wants to feel confident about her findings before she presents a report to NASA. Using k queries to the box, give an algorithm which allows her to detect organic material with a probability which depends on k. i.e. the success probability increases with k.

Solution: Place the rock in the box k times and obtain an independent result for reach test. If any of the k results are NO, output NO. Otherwise output NO with probability p and YES with probability $1 - p$ as before, where we set a different value for p this time.

Case 1: If the answer is YES, then similarly as before, we output YES with probability $1-p$. Case 2: If the answer is NO, then the box will output all k values to be YES with probability $(4/5)^k$. Which means with probability $1-(4/5)^k$, at least one of the results from the box will be NO. Hence, we want p such that $(1-(4/5)^k)+(4/5)^k \cdot p = 1-p$. We can solve this to get $p = \frac{1}{1+(5/4)^k}$.

Thus with probability $1 - p = \frac{1}{(4/5)^k+1}$, our algorithm always produces the correct answer. Furthermore, as k increases, our algorithm's success probability also increases.

Problem-3: Broadcast Channel

A set of up to n processors attempt to communicate over a network. The communication process is deemed successful if *any* of the processors manages to broadcast its information (since the successful processor can then lead the remainder of the communication process). However, the only means of communication is through a common broadcast channel. At any given time step (we assume the time is discrete), any subset of the processors can attempt to communicate through the channel by sending a message. The channel operates as follows:

- If *none* of the processors attempts to send a message, then all processors receive a special "none" message.
- If *only one* of the processors attempts to send a message, then all the processors receive that message, and the communication process is deemed successful.
- If *two or more* processors attempt to send a message, then all the processors receive a special "collision" message.

Suppose that the number of processors is at least $\frac{n}{2}$. Design a randomized protocol that, if followed by all processors, will result in successful communication. The expected number of time steps used by the protocol should be $O(1)$. You may assume all processors know the upper bound n and the lower bound $\frac{n}{2}$.

Solution: We assume $n > 1$, since otherwise the problem is trivial. The algorithm is as follows: at each time step, each processor sends its message with probability $1/n$. If exactly one of the processors manages to broadcast its message, the whole process stops. Otherwise, the protocol is repeated in the next time step.

The analysis is as follows: we will prove that the expected number of time steps used by the protocol is constant. Since each processor sends a message with probability $1/n$, the number of messages sent in each time step follows the binomial distribution. In particular, if there are k processors, the probability that exactly one message is sent at a given time step is

$$P = \binom{n}{k}\frac{1}{n}(\frac{n-1}{n})^{k-1} \geq 1/2 \cdot (1-1/n)^{k-1} \geq 1/2 \cdot (1-1/n)^{n}$$

Since $(1-1/n)^n \to 1/e$ as $n \to \infty$ and $(1-1/n)^n > 0$ for $n > 1$, it follows that $(1-1/n)^n \geq \delta$ for some absolute constant $\delta > 0$ (in fact, we can take $\delta = 1/4$). This implies that $P \geq \delta/2$. The expected number of time steps used by the protocol is at most $1/P \leq 2/\delta = O(1)$.

Problem-4: Testing Polynomial Products

An instance of the *Testing Polynomial Products (TPC)* decision problem is a quadruple of univariate polynomials (A, B, C, D). We say that (A, B, C, D) is a yes-instance if $AB = CD$, and a no-instance otherwise.

Consider the following probabilistic algorithm for solving the problem. On input polynomials (A, B, C, D): Choose uniformly at random an integer y from a set of integers $\{1, \ldots, m\}$. Compute the product $L = A(y)B(y)$ and the product $R = C(y)D(y)$ If $R = L$ outputs "PASS", otherwise output "FAIL".

(a) Analyze the probability that the algorithm is correct as a function of m and n (where the degree of any of the polynomials is at most n). Namely, it passes yes-instances and fails no-instances.

Solution: The product of two degree $\leq n$ polynomials has degree at most $2n$. Therefore, the probability that for a random x, A(x)B(x) = C(x)D(x) is at most $2n/m$. This is the prbability that the algorithm returns PASS when they are not equal.

(b) How large should we make m to make the probability of correctness at least $1 - \frac{1}{\log n}$.

Solution: From part (a), we know that the probability that the algorithm returns PASS when A(x)B(x) $\neq$ C(x)D(x) is at most $2n/m$. Thus, if we set m = $2nlogn$, then the probability that the algorithm returns PASS when the products are not equal will be at most 1/logn. This will help us achieve the bound that the algorithm outputs FAIL with probability at least
$1 - 1/\log n$.

Problem-5: Streaming

(a) You are given an input stream $x_1, x_2, x_3, \ldots, x_n$ where each x_i is an n bit integer. You want to compute the sum of the pairwise products of the integers, but since space is limited, you are only allowed to use $O(n)$ bits of storage.

Design a deterministic streaming algorithm that uses $O(n)$ bits of storage and outputs the sum of pairwise products, i.e. $\sum_{1 \le i < j \le n} x_i x_j$.

Example: Given the input stream $[2, 3, 1, 4]$, the output should be:

$$(2 \times 3) + (2 \times 1) + (2 \times 4) + (3 \times 1) + (3 \times 4) + (1 \times 4) = 35$$

Solution: Solution 1: Keep track of the sum and the pairwise products up to an index m.

Suppose we have kept track of the above two values, so that we have $\sum_{i=1}^{m} x_i$ and $\sum_{1 \le i < j \le m} x_i x_j$. Then, when we process the integer x_{m+1}, we can update the pairwise products using

$$\sum_{1 \le i < j \le m+1} x_i x_j = \sum_{1 \le i < j \le m} x_i x_j + x_{m+1}(\sum_{i=1}^{m} x_i)$$

Afterwards, we can easily update the sum using we can update the sum easily using

$$\sum_{i=1}^{m+1} x_i = \sum_{i=1}^{m} x_i + x_{m+1}$$

The streaming sum is at most the sum of n k-bit numbers. The streaming pairwise product sum is at most the sum of $O(n^2)$ $2k$-bit numbers. The largest this number can be is $O(2^{2k}n^2)$. Thus, the storage needed is $O(\log(2^{2k}n^2)) = O(\log(2^{2k}) + \log(n^2)) = O(k + \log n)$.

Solution 2: Keep track of the first and second moments as each integer is processed.

Note that $(\sum_{i=1}^{n} x_i)^2 = \sum_{i=1}^{n} x_i^2 + 2\sum_{1 \le i < j \le n} x_i x_j$. Thus, if we keep track of the first and second moments, we can find $\sum_{1 \le i < j \le n} x_i x_j = ((\sum_{i=1}^{n} x_i)^2 - \sum_{i=1}^{n} x_i^2)/2$.

The first moment is at most the sum of n k-bit numbers. The second moment is at most the sum of n 2k-bit numbers. Thus, the storage needed is $k + \log n + 2k + \log n = O(k + \log n)$.

(b) Sarah has an $n \times n$ matrix A, and she wants to find the inverse A^{-1} (recall: $A^{-1}A = I$). She finds code on the Internet that claims it does the job. She runs the code on A and gets the output B. However, Sarah is not sure that the code is correct. Thus, she decides to test whether $B = A^{-1}$.

Provide a Monte Carlo algorithm that runs in $O(n^2)$ time for this job and outputs the correct answer with probability at least $3/4$. Prove the correctness of your algorithm.

Solution:

MATRIX-INVERSE-TEST(A, B)

```
1  for i = 1 to 2
2      Pick a vector x ∈ {0,1}^n uniformly at random.
3      y = A(Bx)
4      if y ≠ x
5          REJECT
6  ACCEPT
```

It is clear that if $B = A^{-1}$, then AB is the identity matrix. Thus, y is equal to x and the algorithm always outputs ACCEPT.

Suppose $B \neq A^{-1}$. Let $C = AB - I$. It is not hard to see $Cx = y - x$. Now, we show that given that C is not a zero matrix, Cx will be non-zero with probability at least a half. Let $c_{i,j}$ be one of the non-zero entries of C. Let z be the i-th row of C: $< c_{i,1}, c_{i,2}, \ldots, c_{i,n} >$. Consider any vector $x \in \{0,1\}^n$. Let x' be equal to x except the j-th bit is filliped. It is clear that $|z.x' - z.x| = c_{i,j}$ which is not zero. Thus, at least one of the Cx or Cx' are non-zero. If we pick x uniformly at random, Cx with probability a half will be non-zero. Thus, $x \neq y$ and the algorithm outputs REJECT with probability at least a half. Thus, the probability of not rejecting in both iteration is at most $1/4$. Thus, with probability $3/4$, the algorithm outputs the correct answer.

Problem-6: Graph Completeness

Alice is given a very large undirected graph $G = (V, E)$ with $n = |V|$. We say that a graph is complete if it contains an edge between all $\binom{n}{2}$ pairs of vertices. We say that a graph is ϵ-far from complete if it is missing at least $\epsilon\binom{n}{2}$ edges in order to make it complete.

Alice is given query access to G. Namely, for any u and v vertices in V, Alice can issue queries (u, v) to find out whether $(u, v) \in E$ or $(u, v) \notin E$.

(a) **[14 points]** Help Alice design a Property Testing algorithm with parameter ϵ which makes $O(1/\epsilon)$ queries to G and satisfies the following:

- If G is complete, the algorithm outputs ACCEPT with probability 1.
- If G is ϵ-far from being complete, the algorithm outputs REJECT with probability at least 2/3.

2**Note:** The following inequality may be helpful: $(1 + x) \leq e^x$ for any x. **Solution:**

Version 2

COMPLETENESS-TESTER(G, ϵ)

```
1  m = 2ε^{-1}
2  for i = 1, ... m
3       Pick random u, v ∈ V uniformly at random.
4       if (u, v) ∉ E
5            REJECT
6  ACCEPT
```

It is not hard to see that if G is complete, we always accept.

Now suppose G is ϵ-far from being complete. That means the probability of picking a random u, v such that $(u, v) \notin E$ is at least ϵ. Therefore, the probability that all (u, v) are in E is at most $(1-\epsilon)^m \leq e^{-\epsilon m} \leq 1/3$ for $m = 2/\epsilon$.

Therefore, with probability at least $2/3$, the algorithm will reject.

(b) **[6 points]** Modify your algorithm from part (a) to reduce the error probability $1/3$ to less than $1/\sqrt{n}$ while keeping the number of queries sublinear.

Note: You can do this part without solving part (a).

Solution: From part (a), we have an algorithm that has an error probability of at most $1/3$. We can simply repeat the algorithm $O(\log_3 \sqrt{n})$ to get the error probability to be at most $(1/3)^{\log_3 \sqrt{n}} = 1/\sqrt{n}$. Which means the algorithm is correct with probability at least $1 - 1/\sqrt{n}$.

This is still sublinear since the total number of queries is now $(2/\epsilon) \log_3 \sqrt{n} = O(\log n)$.

Problem-7: [20 points] $\frac{3}{4}$-SAT

In the $\frac{3}{4}$-SAT problem, you are given n boolean variables x_i which can be set to TRUE or FALSE. You are also given m clauses C_j composed of a logical OR of three "literals," i.e. a variable x_i or its negation $\neg x_i$. A clause is "satisfied" if it evaluates to TRUE. A "$\frac{3}{4}$-satisfying assignment" is an assignment of values to variables such that at least $\frac{3m}{4}$ clauses are satisfied.

Below is an example of a clause and a table of assignments that satisfy the clause.

$$C_j = (x_a \vee \neg x_b \vee x_c)$$

x_a	x_b	x_c
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0
0	0	1
0	1	1

(a) [5 points] Design an $O(n)$ time Monte-Carlo algorithm that satisfies $7m/8$ clauses in expectation. You may assume clauses do not have duplicate variables (a literal or its negation).

Hint: Try a random assignment.

Solution: Our algorithm is to independently assign each variable x_i to be 0 or 1 uniformly at random. Since there are three different variables in each clause, this gives it a 1/8 probability of being in any one of the 8 possible configurations. And since only 1 out of the 8 configurations could set the clause to be false, each clause is satisfied with probability $7/8$.

Hence, by linearity of expectation, $7m/8$ clauses will be satisfied in expectation.

(b) [10 points] Show that your algorithm satisfies at least $3m/4$ clauses with probability at least 1/2.

Hint: Use Markov Bound.

Solution: Let U be the number of unsatisfied clauses. From part (a), we know that the expectation of U is $\mathbb{E}[U] = m/8$. Since U is a nonnegative random variable, we can apply Markov Bound to upper bound the probability that $U > \frac{m}{4}$.

$$Pr[U > \frac{m}{4}] \leq \frac{\mathbb{E}[U]}{m/4} = \frac{m/8}{m/4} = \frac{1}{2}$$

Therefore, with probability at least 1/2, the algorithm satisfies at least $3m/4$ clauses.

(c) [5 points] Use parts (a) and (b) to design a Monte Carlo randomized algorithm that runs in $O((m+n)\log n)$ time and outputs a correct $\frac{3}{4}$-satisfying assignment with probability at least $1 - \frac{1}{n^2}$.

Note: You may assume the results from parts (a) and (b).

Solution: Given an assignment, we can count in $O(m)$ time, how many clauses are satisfied with that assignment. Hence, our new algorithm is to repeat our algorithm from part (a) (randomly assign values to each variable) a maximum of $2\log n$ times, until at least $3m/4$ clauses are satisfied.

First, since our algorithm terminates after $2\log n$ iterations, and each iteration takes $O(n)$ time to randomly assign variables and $O(m)$ time to count the number of satisfied clauses, it runs in deterministic $O((n+m)\log n)$ time.

The probability that none of the iterations satisfy at least $3m/4$ clauses is at most $(1/2)^{2\log n} = \frac{1}{n^2}$. Therefore, with probability at least $1 - \frac{1}{n^2}$, at least one of the iterations will be a $3/4$ satisfying assignment.