EECS 376 Final Exam, Winter 2022

Instructions:

This exam is closed book, closed notebook. You may not provide your own scratch paper. No electronic devices are allowed. You may use one 8.5×11-inch study sheet (both sides). Make sure you are taking the exam at the time slot and the classroom you were assigned by the staff.

Any deviation from these rules will constitute an honor code violation. In addition, the staff reserves the right **not** to grade an exam taken in a violation of this policy.

The exam consists of 12 multiple choice questions, 4 short-answer questions, and 3 longer proofs. *You only need to answer two of the longer proof questions.* ***You are to cross out the one proof question you don't want graded.*** For the multiple choice questions, please fill-in the circle you select completely and clearly. ***Please print your UNIQNAME at the top of each page.***

For the short-answer and proof sections, please write your answers clearly in the spaces provided. If you run out of room or need to start over, you can use the blank pages at the end but you **MUST** make that clear in the space provided for the answer. The exam has 20 pages printed on both sides, including this page and two blank pages at the end.

You must leave all pages stapled together in their original order.

Sign the honor pledge below.

Honor pledge:
I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code. I will not discuss the exam with anyone before exam grades are released.
I attest that I am taking the exam at the time slot and the classroom I was assigned by the staff.

Signature: ______________________________

Name: ______________________________

Uniqname: ____________________

[**Markov's Inequality**] If Z is a non-negative random variable and $a > 0$, then

$$\Pr\left[Z \geq a\right] \leq \frac{\mathbb{E}[Z]}{a}.$$

[**Chernoff-Hoeffding**] If $X_1, X_2, ...X_n$ are are independent, identically distributed RVs w/ expectation μ, and range in [0,1] then, for any $k > 0$, the following holds for their sum:

$$\Pr\left[\sum_{i=1}^{n} X_i \geq \mu n + k\right] \leq e^{-2k^2/n}$$

Version C

UNIQNAME (print): ______________________________

Multiple Choice – 36 points

1. Let language $L = \{(A, x) \mid x \text{ is the minimum element of array } A\}$. Let L_2 be an NP-Hard language. The statement $L_2 \leq_T L$ is (always/sometimes/never) true.

   ◯ Always

   [✓ **Sometimes**]

   ◯ Never

   Solution: SOMETIMES

   L is decidable.

   Say L_2 is an NP-Complete language. It is in NP and is thus decidable. Any decidable language is Turing reducible to all languages, as the black box can simply be ignored. Thus, it is true that $L_2 \leq_T L$

   Say $L_2 = L_{HALT}$, which was proven to be NP-Hard. Because it is also undecidable, it cannot be the case that $L_2 \leq_T L$.

   We have shown that the statement is only sometimes true.

2. Using Markov's inequality, what is the upper bound on the probability that tossing a biased coin 40 times results in at least 33 tails if Pr(Heads) = 0.7?

   ◯ 20/33

   ◯ 28/33

   ◯ 23/40

   [✓ **12/33**]

   ◯ 33/28

   ◯ 28/40

   Solution: Probability of tails is $1 - 0.7 = 0.3$. Let X be a non-negative random variable which is the number of tails out of 40 tosses. Then $\mathbb{E}[X] = 0.3 * 40 = 12$. Then Markov's inequality gives $\Pr[X \geq 33] \leq 12/33$

3. ~~Which of the following languages are recognizable?~~

   i $L_{\text{Greater}} = \{M : \ |L(M)| > 1\}$

   ii $L_{\text{Small}} = \{M : \ (|L(M)| == 1) \text{ or } (|L(M)| == 0)\ \}$

   iii $L_{\text{Apple}} = \{M : M \text{ is a TM that accepts the input string “apple”}\}$.

   ◯ i and ii

   ◯ i, ii and iii

   ◯ ii and iii

✓ **i and iii**

○ only iii

Solution: L_{Greater} is recognizable.
Consider the following recognizer:

```
function R($M$)
    $count = 0$
    for $i = 1, 2, \ldots$ do
        for $j = 1, 2, \ldots, i$ do
            run one step of the execution of $M$ on $w_j$
            if $M$ accepts $w_j$ then
                $count = count + 1$
            if $count == 2$ then
                accept
```

$\overline{L_{\text{Greater}}} = L_{\text{Small}}$
L_{Greater} can shown to be undecidable via a Turing reduction from L_{ACC} or by Rice's Theorem.
$\overline{L_{\text{Small}}}$ is recognizable and undecidable. This means that L_{Small} is unrecognizable.

L_{Apple} is recognizable. Consider the following recognizer:

$R(\langle M \rangle, x)$: **run** M on "apple" and if M accepts, accept

4. There is a row of apple trees where each tree i has A[i] apples. If you pick up apples from a given tree then you can't pick up apples from any of its adjacent trees. For instance, If you pick up the apples from tree number 6 you can't pick apples from tree number 5 or tree number 7.
The function $T(n)$ returns the maximum number of apples you can pick up from n trees.
The base cases are $T(1) = \text{A}[1]$ as we only have one tree and $T(0) = 0$.
What is the recurrence relation for $T(n)$?

○ $T(n) = \max(T(n-1), \text{A}[n])$

○ $T(n) = \max(T(n-1), T(n-2))$

○ $T(n) = \max(\text{A}[n] + T(n-1), \text{A}[n] + T(n-2))$

✓ $T(n) = \mathbf{max}(T(n-1), \mathbf{A}[n] + T(n-2))$

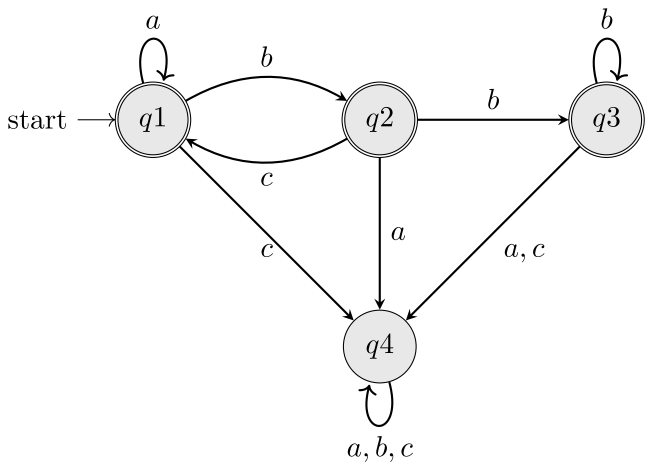
○ $T(n) = \max(\text{A}[n], \text{A}[n+1] + \text{A}[n-1])$

Solution: The correct answer is $T(n) = \max(T(n-1), \text{A}[n] + T(n-2))$.

At tree n if you pick the apples from that tree you get A[n] apples. You can't pick apples from tree $n-1$ due to the adjacency constraint, but all the preceding $n-2$ trees are available to you so you can take the maximum considering the first $n-2$ trees, that is $T(n-2)$. So one option is $\mathrm{A}[n]+T(n-2)$

Another option is not picking apples from the n'th tree and having all $n-1$ preceding trees available. In which case you could take the maximum number from the $n-1$ trees which is $T(n-1)$.
We will take the best of these options:
So $T(n) = \max(\mathrm{A}[n]+T(n-2), T(n-1))$

5. Consider the following DFA, what language does it decide?



[✓ $(a^*|bc)^+a^*b^*$]
◯ $(a^+|bc)^+a^+b^+$
◯ $(a^*|b^*c^*)^+a^*b^*$
◯ $(a^+|bc)^*a^+b^+$
◯ $(a^*|bc)^*(a^+|b^+)$

Solution: Deriving the exact regular expression from the graph is hard though possible, but there is a clever way to do it. Notice from the graph that state $q1$ is an accepting state, meaning that the DFA must accept empty string. For options 2 and 4, they do not accept empty string. Also, input like "*bcc*" should be valid for option 3, but it is rejected by our DFA. The correct option must be 1.

6. Given RSA public key (e=7, n=33), what should be the *signature* for message m=4?

[✓ **31**]
◯ 23
◯ 19

- ○ 25
- ○ 11

Solution: To calculate the signature of a message, we need to find the private key first. Given $n = 33 = 3 \times 11$, we compute totient function $\phi(n) = (3-1) * (11-1) = 20$. Let the private key be d, then $7d + 20k = 1$. Doing an extended Euclidean algorithm on gcd(7, 20), we find that $3 \times 7 - 20 = 1$, so $d = 3$. Finally we take $m^d \equiv 4^3 \equiv 64 \equiv 31 \pmod{33}$, so choice 1 is correct.

7. What is $5^{146} \pmod{13}$?

- ○ 8
- ○ 3
- ✓ **12**
- ○ 5
- ○ 4

Solution: Since 13 is prime we know that $a^{12} \equiv 1 \pmod{13}$ by Fermat's little theorem. Therefore $5^{146} \pmod{13} \equiv 5^{144} * 5^2 \pmod{13} \equiv (5^{12})^{12} * 5^2 \pmod{13} \equiv 5^2 \pmod{13} \equiv 25 \pmod{13} \equiv 12$

UNIQNAME (print): ____________________

8. Let B-3N be a divide-and-conquer algorithm. On an input of size n, B-3N splits the input into 5 subproblems of size $\frac{n}{3}$. Splitting the input and recombining the results takes $O(3n+1)$ time. Using the Master Theorem, what is the tightest time complexity of B-3N?

   [✓ $O(n^{\log_3 5})$]
   ◯ $O(n)$
   ◯ $O(n \log n)$
   ◯ $O(n^5)$
   ◯ The Master Theorem cannot be used to find the time complexity of B-3N.

Solution: The time complexity of B-3N can be represented by the recurrence relation $T(n) = 5 \cdot T(\frac{n}{3}) + O(n)$. Applying the Master Theorem, we find that $T(n) \in O(n^{\log_3 5})$.

9. ~~Let L_1 and L_2 be recognizable languages. Then $L_1 \setminus \overline{L_2}$ is (always/sometimes/never) recognizable.~~

~~*Recall:* "\" denotes set difference. $A \setminus B = \{x : x \in A \text{ and } x \notin B\}$.~~

   [✓ **Always**]
   ◯ Sometimes
   ◯ Never

Solution: $x \in L_1 \setminus \overline{L_2}$ iff $x \in L_1 \cap \overline{\overline{L_2}} = L_1 \cap L_2$. Recognizability is closed under intersection, so we know that $L_1 \setminus \overline{L_2}$ is always recognizable.

10. Consider a language A that is NP-Complete. For some arbitrary language L, if A $\leq_p$ L then L is (always/sometimes/never) NP-Complete.

   ◯ Always
   [✓ **Sometimes**]
   ◯ Never

Solution: Sometimes. L can be NP-Hard and not NP-C. For example, we showed that $SAT \leq_p L_{halt}$ in discussion 9 worksheet problem 3, and we know that $L_{halt} \notin NP$ because L_{halt} is not decidable. L can also be NP-Hard and NP-C. As an example, we can reduce any NP-C language to itself.

11. ~~Which of the following languages can be shown to be undecidable by a direct application of Rice's Theorem? (Select all that apply)~~

   ◯ $L = \{(\langle M \rangle, x) \mid M(x) \text{ halts on the } 376^{th} \text{ step}\}$
   ◯ $L = \{\langle M \rangle \mid L(M) \subseteq \sum^*\}$
   ◯ $L = \{x \mid x \text{ is a prime number}\}$

[✓ $L = \{\langle M\rangle \mid M$ **accepts an even number of inputs**$\}$]

◯ $L = \{\langle M\rangle \mid M$ accepts the input '376' in 101 steps$\}$

Solution: The fourth choice
To be able to show a language is undecidable using a direct application of the Rice's Theorem, there has to be a semantic property $\mathbb{P}$ that is nontrivial.

- The "input" to the language is not in the proper format for Rice's Theorem to apply. Also, it's decidable.
- While this language does have the semantic property $\mathbb{P}$ where $\mathbb{P} = \{L \subseteq \Sigma^* : L \subseteq \sum^*\}$. Notice that this set includes all recognizable languages which means $\mathbb{P}$ is trivial. Thus, this language is decidable.
- The "input" to the language is not in the proper format for Rice's Theorem to apply. Also it's trivially decidable.
- This language has a semantic property $\mathbb{P}$.
  $\mathbb{P} = \{L \subseteq \Sigma^* : |L| \text{ is even}\}$
  The language corresponding to this property is $L_{\mathbb{P}} = \{M : |L(M)| \text{ is even}\}$
  $\mathbb{P}$ is also nontrivial, because we can construct two recognizable languages L and L' where L accepts an even number of inputs (reject everything) and where L' accepts an odd number of inputs (accept only one string).
- This language is decidable.

12. Which of the following statements is **false**?

◯ Vertex-Cover is efficiently reducible to the Knapsack problem.

◯ All languages that can be efficiently verified are decidable.

◯ If P = NP, then all NP-complete problems are in coNP.

[✓ **If NP $\neq$ coNP, there is at least one NP-hard language that is in coNP.**]

Solution: The fourth answer is false. Only if $NP = coNP$, the NP-complete languages would be in coNP (the intersection between NP-Hard and coNP is not empty). However, if $NP \neq coNP$, if any NP-hard problem is also in coNP, since all problems in NP could be reduced to it, all problems in NP would be in coNP, which means that $NP \subset coNP$. As coNP is the complement of NP, $coNP \subseteq NP$. It would be $NP = coNP$. By a proof of contradiction, we have arrived at the conclusion that if $NP \neq coNP$, it's impossible to have any NP-hard language that is in coNP as well.

The first answer is true. Knapsack problem is an NP-hard problem, therefore all languages that are in NP could be reduced to Knapsack problem. Vertex-Cover is an NP-complete problem, therefore it could be reduced to the Knapsack problem.

The second answer is true. All languages that are in NP should be decidable.

The third answer is true. If P = NP, NP = coNP. Therefore all NP-complete problems would be in coNP as well.

UNIQNAME (print): ______________________

Shorter Answer – 34 points

1. **Encryption**

   (10 points) Sarah and Matt wish to communicate securely and so they need to obtain a shared secret key k, but they can only communicate over an *nonsecure* channel! So, they turn to the Diffie-Hellman protocol to establish their secret key. They pick their favorite prime $p = 19$ and a generator $g = 4$ (wait a second, 4 is not a generator for $\mathbb{Z}_{19}$! 4 does not generate 18 mod 19, for example) Sarah then picks the private key $a = 5$ and Matt picks the private key $b = 16$.

   (a) Compute Sarah's and Matt's public keys A and B as well as their shared secret key, k.

   Solution: Sarah's public key is $A \equiv 4^5 \equiv 17 \mod 19$ and Matt's public key is $B \equiv 4^{16} \equiv 6 \mod 19$. This is computed via fast modular exponentiation as follows:

$$\begin{aligned} 4^2 &\equiv 16 \mod 19 \\ 4^4 &\equiv 16^2 \equiv 256 \equiv 9 \mod 19 \\ 4^8 &\equiv 9^2 \equiv 81 \equiv 5 \mod 19 \\ 4^{16} &\equiv 5^2 \equiv 25 \equiv 6 \mod 19 \end{aligned}$$

   We note that $k \equiv A^b \equiv B^a \mod 19$. Computing it either way is acceptable. We proceed by performing fast modular exponentiation to calculate $k \equiv 6^5 \mod 19$:

$$\begin{aligned} 6^2 &\equiv 36 \equiv 17 \mod 19 \\ 6^4 &\equiv 17^2 \equiv 289 \equiv 4 \mod 19 \\ 6^5 &\equiv 4 \cdot 6 \equiv 24 \equiv 5 \mod 19 \end{aligned}$$

   So, $k \equiv 6^5 \equiv 5 \mod 19$.

   (b) Their arch nemesis, Eve, has been eavesdropping the whole time! What are **all** the parameters that Eve is able to extract *without performing any computation*?

   Solution: $p = 19, g = 4, A = 17, B = 6$. Note, Eve can only find out their shared secret key through computation.

   (c) Notice that a generator was not used here at all, but this procedure worked fine. Using a generator is not necessary for the correctness of Diffie-Hellman. Why is it that we *prefer* to use generators (or elements that can generate close to the entirety of $\mathbb{Z}_p^*$ - *"almost generators"*)? Your justification does not need to be lengthy, 2-3 sentences suffices.

   Solution: We prefer to use generators since generators will generate a larger *range* of integers in $\mathbb{Z}_p^*$ through exponentiation. In the context of Diffie-Hellman, since everything is computed through exponentiation, this would mean that an attacker would have a wider *range* of integers to search/brute force through in order to successfully figure out the shared secret key.

2. (10 points) Suppose you have $n \geq 2$ bins. You independently toss n^2 balls at the bins, uniformly at random. You may find formulas on the first page to be helpful.

   (a) For each $i \in \{1, \ldots, n\}$, what is the expected number of balls in bin i?

   (b) Use Markov's inequality to bound the probability that bin i has at least $2n$ balls.

   (c) Use Chernoff-Hoeffding to bound the probability that bin i has at least $2n$ balls.

Solution: (a) Let X be the number of balls in bin i and let X_j be the indicator random variable for whether ball $j \in \{1, \ldots, n^2\}$ lands in bin i. Since each $E[X_j] = 1/n$, by linearity of expectation, $E[X] = \sum_j E[X_j] = n$.

(b) Markov's inequality says that $\Pr(X \geq 2n) \leq E[X]/2n = 1/2$.

(c) Chernoff-Hoeffding says that $\Pr(X \geq 2n) \leq e^{-2n^2/n^2} = e^{-2}$.

3. (4 points)

   Prove or disprove the following statement *assuming* $\mathsf{P} \neq \mathsf{NP}$

   Suppose $L \in \mathsf{NP}$, and $L \leq_p \text{SAT}$, then $L \in \mathsf{NP}$-complete.

   Solution: This is not always true, and we will construct a counterexample by picking any language in P. Let $L = \Sigma^*$, then we know $L \in \mathsf{P}$, so $L \in \mathsf{NP}$. We also know that $L \leq_p \text{SAT}$, because a valid mapping function could just ignore input from L and output a satisfiable boolean formula (notice that L is on the left side of polynomial time reduction but not on the right side). However, $L = \Sigma^*$ is not in NP-hard, so $L \notin \mathsf{NP}$-complete.

4. (10) Exactly two of the following statements are known to be true. Fill in the circle next to the true statements and then briefly show why each of the those two statements are true. For the statements you think are false, you do not need to prove anything. **You are to assume** $\mathsf{P} \neq \mathsf{NP}$**.**

   ◯ $A \leq_p A$ for any language $A \in$ NP.

   > **Solution:** True. There exists an efficiently-computable function which takes an element in A and maps it to an element in A; this is the identity function $f(x) = x$. The fact the language is in NP doesn't change this since it's true of all languages.

   ◯ (L is undecidable) $\implies$ ($L \in$ NP)

   > **Solution:** This statement is false. In fact no undecidable language is in the class NP.

   ◯ $(L \in P) \implies (\overline{L} \in P)$.

   > **Solution:** True. If $L \in P$, there exists an efficient decider M_L for L. We can construct an efficient decider $M_{\overline{L}}$ that, on input x, simply runs $M(x)$ and outputs the opposite answer. It is obvious that $M_{\overline{L}}$ is an efficient decider, and $M_{\overline{L}}(x)$ accepts if and only if $M(x)$ rejects, which happens if and only if $x \notin L$. Therefore, $M_{\overline{L}}$ decides $\overline{L}$.

   ◯ (($A \leq_p B$) and ($B \in$ NP-hard)) $\implies$ ($A \in$ NP-hard)

   > **Solution:** This is false. Let $A \in \mathsf{P}$ and $B = \mathsf{SAT}$. By our assumption $\mathsf{P} \neq \mathsf{NP}$, then $A \leq_p B$, B is NP-hard, and yet A is not NP-hard.

Proofs and longer questions – 30 points

Answer two of the following three questions. **Clearly cross out the question you do not want graded.** If it isn't clear what problem you don't want graded, we will grade the first two. Each of the two questions are worth 15 points.

1. Let A be a array of n items each with a weight of A[i]. Given a integer size B, and a capacity, K, we say that the array is "packable" if we can split it up into K partitions such that the sum of each partition is less than or equal to B. Formally we define:

$$\textsc{Packable} = \{\langle A, B, K\rangle \mid \text{An array } A \text{ with } n \text{ elements, with each element a rational number, can be partitioned into } K \text{ subarrays such that the sum of each subarray is less than or equal to an integer B.}\}$$

   (a) Show that PACKABLE $\in$ NP

Solution:
A certificate c is a set of K non-intersecting subbarrays of A. We can construct a verifier for PACKABLE as follows:
V on input $((A, B, K), c)$

i. If c is not made up of K subarrays, reject
ii. Concatenate all subarrays of c then sort. If this doesn't equal A after it is sorted, reject
iii. For each subarray in c, if the sum is $>$ B in any, reject
iv. accept

The verifier is correct because if there is a valid partitioning of A, there will be a certificate c which will be made up of K subarrays that when put together and sorted will be equivalent to A sorted and for all of the subarrays the sum will be less than or equal to B. If there is no valid partitioning of A, there will be no such certificate c.
The verifier is efficient because checking the structure of c, sorting and summing are all efficient operations with max of $O(n^2)$ complexity.

   (b) Show that PACKABLE $\in$ NP-HARD. HINT: You may use the fact that that

$$\text{PARTITION} = \{\langle A\rangle : \text{A is a set of } 2m \text{ numbers which can be partitioned into 2 non-intersecting subsets with equal sums}\}$$

is NP-Complete.

The next page has been left blank, please keep your answer on this page and the next.

Solution:
We will show that PARTITION $\leq_p$ PACKABLE
Consider the mapping f: $\Sigma^* \to \Sigma^*$ defined by f($\langle A \rangle$) = $\langle A', 1, 2 \rangle$
Where A'[i] = $\frac{2*A[i]}{\Sigma_{i=1}^{i=2*m} A[i]}$
This is clearly an efficient mapping as we can calculate the sum of A in linear time and then figure out each element of A' in linear time.
If some $w = A \in$ PARTITION then A' can be parititoned into two subsets that each sum to 1
If $\langle A', 1, 2 \rangle \in$ PACKABLE then that means that there must exist a partitioning of A' into two subarrays that each have a sum less than or equal to 1. Notice that the total sum of all the elements in A' is 2 so each of these 2 partitions must sum to exactly 1 in order to meet the requirement of having sums less than or equal to 1. This means that the sum of the two subarrays is equal which means that there must exist a partitioning of A into two subarrays of equal size in the original set A. thus $f(w) \in$ PACKABLE $\implies w \in$ PARTITION
Thus PARTITION $\in$ NP-HARD

UNIQNAME (print): ______________________________

This page left blank, please use only for proof problem 1

2. Bob is sending a message to Alice using the RSA algorithm as taught in class. Say that Bob wishes to communicate message m and thus sends $m' = m^e \bmod n$ as normal, where e is Alice's public key. Eve knows e, the RSA modulus n, and the encoded message m'. In this problem, we consider the classic model of computation. (In particular, we exclude quantum algorithms.) Hence, you may assume that there is no (publicly) known efficient way to find m given only e, n, and m'.

   (a) If Eve knows the factors of n, is there a (publicly) known efficient algorithm for Eve to find m? If so, provide that algorithm, an argument as to why that algorithm works, and an explanation as to why that algorithm is efficient. If not, explain how you know that.

   (b) Suppose Bob's friend Carl spills the beans to Eve and tells her the correct value of m which encrypts to m'. Now Eve has access to m in addition to all the public information she already had. Is there any (publicly) known efficient algorithm that she could use to factor n given this information, i.e., message m, its ciphertext m', RSA modulus n, and public key e? If so, provide that algorithm, an argument as to why that algorithm works, and an explanation as to why that algorithm is efficient. If not, explain how you know that.

Solution:

(a) Yes. She knows the factors p and q of n, so she can compute $\phi(n) = (p-1)(q-1)$. Then she can compute $d = e^{-1} \bmod n$, and thus she can compute $(m')^d = m \bmod n$. This works because $ed = 1 \bmod \phi(n)$, so $m^{ed} = (m')^d = m \bmod n$ by extended Fermat's Theorem. She can compute modular inverses efficiently by Extended Euclid, and she can compute $(m')^d$ efficiently by fast modular exponentiation.

(b) No. If such an alg existed, then Eve could efficiently recover Alice's private key given only n and e:

- Pick a random message m and compute $c = m^e \bmod n$.
- Use the alg from the problem statement to get the factors p, q of n given message m, ciphertext c, modulus n, and public key e.
- Compute $\varphi(n) = (p-1)(q-1)$, and then compute Alice's private key $d = e^{-1} \bmod \varphi(n)$.

There is no known way that Eve could efficiently recover Alice's private key given only n and e, so there is not a known algorithm which could accomplish the task from the problem statement.

UNIQNAME (print): ____________________

3. In the original 3SAT problem, a clause is satisfied if at least one of its literals is true. Suppose we modify the 3SAT problem such that a clause is satisfied only if at least one of its literals is true and at least one of its literals is false. We name this problem 3SATFT. Given an arbitrary **exact** 3CNF formula ϕ (i.e. within any given clause, there are 3 distinct variables), prove that there exists an assignment that satisfies 3/4 of the clauses for 3SATFT.

Solution: First, we will show that a random assignment of truth values to the variables will satisfy three quarters of the clauses in expectation. Our plan is to use linearity of expectation.

For 3SATTF: $\ldots \wedge (x_{i1}, x_{i2}, x_{i3}) \wedge \ldots,$

x_{i1}	x_{i2}	x_{i3}	clause satisfied
F	F	F	F
F	F	T	T
F	T	F	T
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	T
T	T	T	F

Suppose we have n clauses that are labeled c_i for $1 \leq i \leq n$. Each c_i consists of the three literals x_{i1}, x_{i2}, and x_{i3}. The i^{th} clause is satisfied when precisely two of these literals are true. Note that six of the eight possible assignments of truth values to these literals will satisfy the clause (the only exceptions being when all of the literals are true or when all of the literals are false).

If we define Y_i to be the indicator variable that is 1 when the i^{th} clause is satisfied by a random assignment of truth values, then the above shows that

$$\text{Ex}[Y_i] = \frac{6}{8} = \frac{3}{4}.$$

Now, $Y := \sum_{i=1}^{n} Y_i$ is the number of satisfied clauses. By linearity of expectation,

$$\text{Ex}[Y] = \text{Ex}\left[\sum_{i=1}^{n} Y_i\right] = \sum_{i=1}^{n} \text{Ex}[Y_i] = \sum_{i=1}^{n} \frac{3}{4} = \frac{3}{4}n.$$

So, three quarters of the clauses are satisfied in expectation.

Our next step is to prove the statement using this fact, along with results from class. Namely, the random variable Y must take on a value that is at least as large as its expected value. Rephrasing in the context of this problem, there is at least one assignment of truth values that satisfies three quarters of the clauses. $\square$

UNIQNAME (print): ______________________________

This page intentionally left blank.

If you have answers on this page be certain you write “answer continues on page 13” under the actual question.

UNIQNAME (print): ______________________________

This page intentionally left blank.

If you have answers on this page be certain you write "answer continues on page 14" under the actual question.