Final Exam
Practice Problems

1. Consider the following two computational problems:
   - Given as input a number A, the output is a number B such that $B^3 = A$. If no such number exists, the output is "False".
   - Given as input numbers A and N, the output is a number B such that $B^3 \equiv_N A$. If no such number exists, the output is "False".

   For each of these problems, say whether it can be computed in polynomial time. If you claim that a problem can be computed in polynomial time, then present the polynomial-time algorithm. If you claim that a problem cannot be or is not known to be computable in polynomial time, then no justification is required.

2. As we saw in lecture, if one has an efficient algorithm to factor a given number, then one can break RSA by efficiently computing $\varphi(N)$ where $N = PQ$ is the product of two distinct primes. Show that if one has an efficient algorithm that given N (which is a product of two distinct prime numbers) computes $\varphi(N)$, then one can factor N efficiently.

3. Let $M, N \geq 2$ be integers satisfying $\gcd(M, N) = 1$. Furthermore, let $0 \leq A < M$ and $0 \leq B < N$ be integers. We are interested in finding an integer X which simultaneously satisfies
$$X \equiv_M A,$$
$$X \equiv_N B.$$
   (a) Solve this problem when $M = 3$, $N = 5$, $A = 1$, $B = 2$.

   (b) Solve this problem when $M = 25$, $N = 249$, $A = 7$, $B = 11$.
   (Hint: Use the fact that $X \equiv_{25} 7$ if and only if $X = 25k + 7$ for some integer k. Then "plug this in" to the second congruence, $X \equiv_{249} 11$ and solve. . . We hope you find a certain relationship between 25 and 249 easy to compute. . .)

   (c) For the **general** version, prove that the desired X always exists. Be sure to clearly state where you use the assumption $\gcd(M, N) = 1$. In addition, explain how X can be found in poly(n) time, where $n = \log_2 M + \log_2 N$.

4. Let G be a group with 251 elements. Show that G must be Abelian.
   Hint: 251 is a prime number.

5. Recall the Diffie-Hellman secret-key exchange protocol: Alice picks a prime P, a generator $G \in \mathbb{Z}_P^*$, and a random $A \in \mathbb{Z}_{P-1}$. She sends P, G, and G^A to Bob. Bob picks a random $B \in \mathbb{Z}_{P-1}$ and sends G^B to Alice. Alice computes $(G^B)^A = G^{AB}$ and Bob computes $(G^A)^B = G^{AB}$, which is their secret key. All computation is done in $\mathbb{Z}_P^*$.

   (a) Generalize the above scheme to 3 players so that Alice, Bob and Charlie can share a secret key.

(b) What are the values seen by the eavesdropper Eve in your 3-party secret-key exchange protocol?

(c) Show that if your 3-party protocol is secure, then so is the original 2-party protocol. (We say that the protocol is *not secure* if Eve is able to compute the secret key given what she sees.)

6. Fix the alphabet $\Sigma = \{0, 1\}$. Let $\text{diff}(w)$ denote the number of 1's in w minus the number of 0's.

   (a) Define $L_n = \{w \in \Sigma^* : \text{diff}(w) \text{ is divisible by } n\}$. Show that L_n is regular.

   (b) Define $L = \bigcup_{p \text{ a prime}} L_p$. Show that $L = \{w \in \Sigma^* : \text{diff}(w) \neq \pm 1\}$.

   (c) Using the second characterization of L, and the 'pigeons'

   $$\epsilon, 111, 111111, 111111111, \ldots, 1^{3i}, \ldots$$

   show that L is not regular.

   Using the fact that regular languages are closed under the union operation, conclude: There are infinitely many primes!

7. (a) Let $\mathcal{C}$ be the set of all circuits (using NOT gates, and OR and AND gates of arbitrary fan-in). Prove or disprove: $\mathcal{C}$ is countable.

   (b) Let $\mathcal{F}$ be the set of all families of circuits. Prove or disprove: $\mathcal{F}$ is countable.

8. In class we defined a Boolean circuit so that it has a single output gate. However, we can generalize the definition so that multiple gates are designated as output gates. A circuit with n input gates and m output gates would compute a function $f : \{0,1\}^n \to \{0,1\}^m$.

   Consider $\text{ADD}_n : \{0,1\}^{2n} \to \{0,1\}^{n+1}$ which takes as input two n-bit numbers, and outputs their sum as an $(n+1)$-bit number. Show that ADD can be computed by a circuit family of size $O(n)$.

9. In this question we fix our alphabet to Σ and define the notion of NP-hardness using Karp reductions.

   (a) Prove that $\emptyset$ and Σ^* are not NP-complete. (Note that we are not making any assumptions about whether $\mathsf{P} = \mathsf{NP}$ or not.)

   (b) Prove that if $\mathsf{P} = \mathsf{NP}$, then every language in NP, except for $\emptyset$ and Σ^*, is NP-complete.

10. Let $L = \{\langle M \rangle : M \text{ is a TM and } L(M) \text{ is } \mathsf{NP}\text{-complete}\}$. Prove or disprove: L is decidable.

11. Let T be a tree on n vertices that has no vertex of degree 2. Show that T has more than $n/2$ leaves.

12. A *subgraph* H of an (undirected) graph G is any graph whose set of vertices and edges are a subset of the set of vertices and edges of G respectively. Prove that a graph $G = (V, E)$ is bipartite if and only if every subgraph $H = (V_H, E_H)$ of G has an independent set consisting of at least half of V_H. (An independent set is a subset of the vertices in which no two vertices are connected by an edge.)

13. (a) We are given an undirected graph G in which each edge has a positive integer cost associated to it. We are also given two vertices s and t, and we want to find the cheapest path from s to t. One idea for this starts as follows: "First, replace each edge of cost c with a path of length c (by inserting $c-1$ dummy vertices into it). Now, use BFS…". Explain the rest of this idea, and why it is correct.

    (b) Prove that algorithm just described, while correct, is not a polynomial-time algorithm.

14. (a) Show that there is a polynomial time algorithm that given as input a Boolean formula, outputs True if and only if there is a satisfying assignment for the formula in which exactly two variables are set to True.

    (b) Show that for any $k \geq 0$, there is a polynomial time algorithm A_k that given as input a Boolean formula, outputs True if and only if there is a satisfying assignment for the formula in which exactly k variables are set to True.

    (c) Consider the following algorithm for SAT: given as input a Boolean formula φ with n variables, run $A_1(\langle\varphi\rangle), A_2(\langle\varphi\rangle), \ldots, A_n(\langle\varphi\rangle)$, and if any of them returns True, then return True. Otherwise return False. Does this algorithm show that SAT is in P?

15. In this problem, 0 and 1 represent False and True, respectively.

    Let $M : \{0,1\}^5 \to \{0,1\}$ be the Boolean predicate that is True if at least 3 of its inputs are True; in other words, $M(x_1, x_2, x_3, x_4, x_5) = 1$ if and only if $x_1 + x_2 + x_3 + x_4 + x_5 \geq 3$.

    ("M" stands for Majority.) An "MNF" is something that looks like this:

    $$M(x_2, x_2, x_8, \neg x_6, x_{10}) \wedge M(x_1, x_n, x_{32}, \neg x_5, x_{11}) \wedge \cdots \wedge M(\neg x_3, x_{33}, x_{12}, x_1, x_9).$$

    More precisely, an MNF is the logical AND of a bunch of M-predicates applied to some Boolean variables $x_1, \ldots, x_n$ and their negations.

    The MNF-SAT decision problem is: Given as input an MNF, accept if it is "satisfiable" and reject if it is "unsatisfiable". As usual, we say that an MNF is satisfiable is there is a truth assignment to its variables that overall makes it evaluate to True.

    Prove that the MNF-SAT problem is NP-hard. Your reduction must be a Karp reduction.

16. Show that PARTITION is NP-complete.

17. Let $n \geq 2$ be an integer. Suppose we run the following randomized code, which fills up a length-n array called A with a sequence 0's and 1's, and then computes the sum:

    $A[1] \leftarrow 0$

    $A[2] \leftarrow 1$

    for $t = 3 \ldots n$

        $prev \leftarrow \texttt{RandInt}(t-1)$

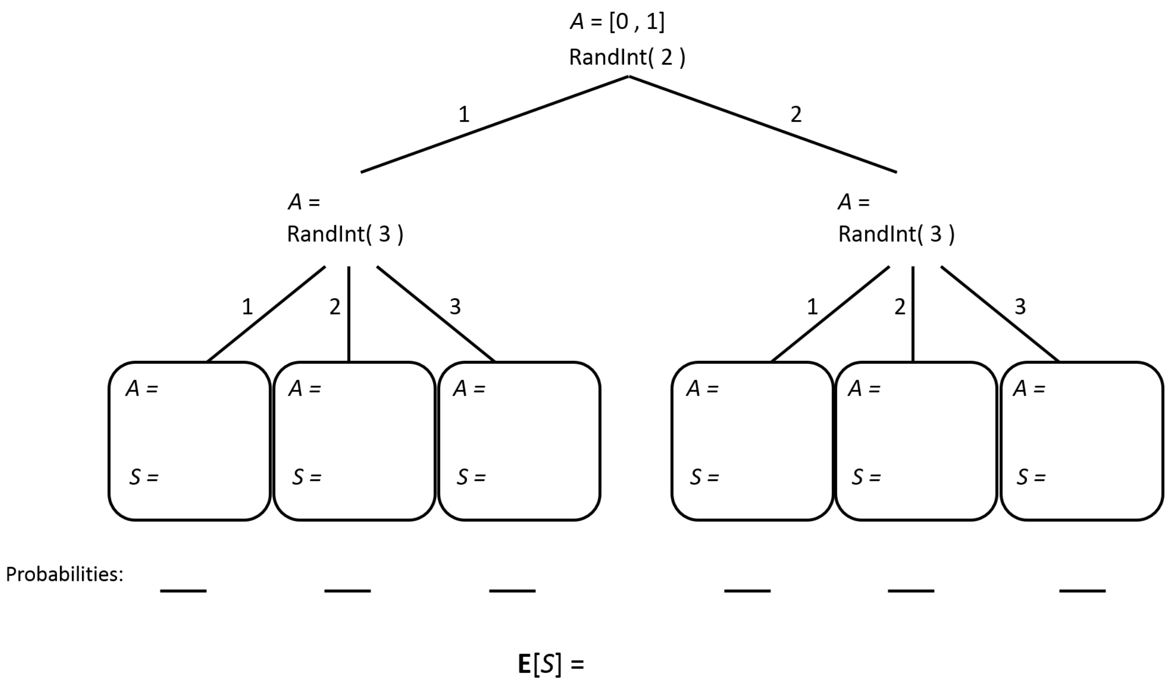
        $A[t] \leftarrow A[prev]$

    end for

    $S \leftarrow A[1] + A[2] + \cdots + A[n]$

    The below probability tree is for the special case of $\boxed{\boldsymbol{n=4}}$. Do the following:

- List the contents of array A at the **eight** missing spots (two internal nodes, six leaves).
- Write the final value of random variable S at each of the six leaves.
- Write the probabilities associated to each outcome under the leaves.
- Fill in $\mathbf{E}[S]$.



Consider the randomized code now for **general $\boldsymbol{n}$**. Prove that $\mathbf{Pr}[S = k] = \frac{1}{n-1}$ for each integer $1 \le k \le n-1$. (Hint: easiest method involves induction on n.) Also, compute $\mathbf{E}[S]$.

18. Consider the following code:

    $X \leftarrow (\texttt{RandInt}(11) \ \texttt{mod} \ 3) + 1$

    $Y \leftarrow \texttt{Bernoulli}(\frac{5-X}{4}) + X$

    (a) Draw the associated probability tree, and indicate the value of X and Y below each outcome.

    (b) Compute $\mathbf{E}[X]$, $\mathbf{E}[Y]$, $\mathbf{E}[2X - 3Y]$, and $\mathbf{E}[X^2Y]$.

    (c) Compute $\mathbf{E}[X \mid Y = 2]$, $\mathbf{E}[Y \mid X \text{ is odd}]$, and $\mathbf{E}[X^2 - 1 \mid Y = 3]$.

    (d) Prove that X and Y are not independent.

19. Let $L[1 \dots n]$ be an array containing 0's and 1's. A "doubleton" is a pair of consecutive indices $(i, i+1)$ such that $L[i] = 1$ and $L[i+1] = 1$. The "doubleton-count" for L is the total number of doubletons. For example, if $L = [0, 1, 1, 1, 0, 1, 1, 1, 1]$, the doubleton-count is 5, because of the doubletons $(2, 3)$, $(3, 4)$, $(6, 7)$, $(7, 8)$, $(8, 9)$.

    Now suppose we form L by executing the following code:

for i from 1 to n

  $L[i] \leftarrow \text{Bernoulli}(1/2)$

Let D be the random variable giving the doubleton-count of L. Determine $\mathbf{E}[D]$, with proof.

20. (a) Suppose G is a graph with n vertices. Show that the number of cycles in G of length k is at most n^k.

    (b) Suppose we form a graph G with n vertices by choosing its adjacency matrix A as follows:

    for i from 1 to n

      for j from $i+1$ to n

        $A[i,j] \leftarrow \text{Bernoulli}(p)$

        $A[j,i] \leftarrow A[i,j]$

    Here $0 < p < 1$ is a parameter.

    Prove that the *expected* number of cycles in G is at most $\sum_{k=3}^{n}(np)^k$.

    (c) Suppose $p \leq \frac{1}{5n}$. Prove that G is *acyclic* (has no cycles) with probability at least 99%.

21. In this question, we will say that an algorithm is randomized if it has access to Bernoulli(1/2) and does not have access to any other kind of randomness. Let L be a language with the property that there is a polynomial-time randomized algorithm A such that if $x \in L$, then $\mathbf{Pr}[A(x) \text{ accepts}] > 0$ and if $x \notin L$, then $\mathbf{Pr}[A(x) \text{ rejects}] = 1$. Show that $L \in \mathsf{NP}$ using the formal definition of NP.

22. Alan is taking 15-451 this semester. For Homework 17, he is given a problem set with n problems. Unfortunately he has run out of paper, so he decides to write his solutions on toilet paper. He would like to write the solutions in a way that uses the least number of toilet paper squares. The solution to the i^{th} problem takes a_i units of length to write up where a_i is a real number in the range $(0, 1]$, for all $i \in \{1, 2, \ldots, n\}$. Each toilet paper square is exactly 1 unit length. Unfortunately, the TAs are rather eccentric; they demand that each solution must be entirely contained in one square. A square *can* contain more than one answer. As an example, suppose $n = 3$ and $a_1 = 1/3$, $a_2 = 5/8$, and $a_3 = 1/2$. Then we could write the solutions on two squares: one square contains a_2 and the other square contains a_1 and a_3.

    Help Alan out by giving a polynomial-time algorithm that uses at most 2 times the minimum number of toilet paper squares necessary to solve this problem (i.e., give a polynomial-time 2-approximation algorithm).