CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2025**

"Homework 0": Review of Background

No due date: ungraded assignment with no submission

Homework Policy

You do not need to turn in anything for this assignment but you are strongly encouraged to attempt it on your own or with collaboration with other students in the course.

You may not have the required background knowledge for all the questions and that is certainly fine (and some of the questions are more challenging than the rest). But you should generally be able to get > 50 points in this assignment if you have the right background for the course.

Problem 1 (Induction and Probability)**.** We start with one black ball and one white ball in a bin. We repeatedly do the following: choose one ball from the bin uniformly at random, and then put the ball back in the bin with another ball of the same color. We repeat until there are n balls in the bin. Show that the number of white balls is uniformly distributed on numbers 1 and $n-1$. **(10 points)**

Problem 2 (Probability)**.** Suppose we throw a fair six-sided dice until we see the number two. Compute the expected number of throws. Then, compute the expected number of throws conditioned on the event that we only see *even* numbers in all the throws. **(10 points)**

Problem 3 (Probability)**.** Let A and B be two random variables. Prove that if A is independent of B, then

$$\mathrm{Var}\left[A+B\right] = \mathrm{Var}\left[A\right] + \mathrm{Var}\left[B\right].$$

Then, give an example of two correlated random variables A, B such that

$$\mathrm{Var}\left[A+B\right] \neq \mathrm{Var}\left[A\right] + \mathrm{Var}\left[B\right].$$

Try to see if you can come up with differents examples where either side of the above equation is strictly larger than the other. **(10 points)**

Problem 4 (Greedy Algorithms)**.** Suppose we have n persons with their heights given in an array $P[1:n]$ and n skis with heights given in an array $S[1:n]$. Design and analyze a greedy algorithm that in $O(n \log n)$ time assign a ski to each person so that the average difference between the height of a person and their assigned ski is minimized. The algorithm should output a permutation σ of $\{1, \ldots, n\}$ with the meaning that person i is assigned the ski $\sigma(i)$ such that

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \left|P[i] - S[\sigma(i)]\right|$$

is minimized. **(10 points)**

Problem 5 (Dynamic programming). You are given three integer arrays A, B, C of size n each. Design an $O(n^2)$ time algorithm that determines if there are indices $i, j, k \in [n]$ such that $A[i] + B[j] = C[k]$. You will receive half the points for designing an algorithm with $O(n^2 \cdot \log n)$ time. **(10 points)**

Problem 6 (Matrices and Graphs). Let $G = (V, E)$ be an undirected graph on n vertices and A be the adjacency matrix of A. Prove that the number of triangles in A, i.e., cycles of length 3, is equal to $trace(A^3)/6$. Recall that for a square matrix B, $trace(B)$ is the sum of diagonal entries of B. **(10 points)**

Problem 7 (Graph Theory). A tree is any connected graph without a cycle. Use only this definition to prove that every tree has exactly $n - 1$ edges. **(10 points)**

Problem 8 (Graph Theory). Let $G = (V, E)$ be an undirected graph with *distinct* edge-weights $w : E \to \mathbb{N}$. Under this condition, prove that the minimum spanning tree (MST) of G is unique. **(10 points)**

Problem 9 (Graph Theory). Let $G = (V, E)$ be a directed graph such that for every vertex $v \in V$, the in-degree and out-degree of v are equal. Suppose G contains k edge-disjoint paths from some vertex s to another vertex t. Under these conditions, must G also contain k edge-disjoint paths from t to s?

Give a proof or a counterexample with explanation. **(10 points)**

Problem 10 (Complexity Theory). In the *3-SAT* problem, we are given a *3-CNF* formula with m clauses the form $(a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots$ where each a_i, b_i, c_i is a literal chosen from n variables or their negations; the goal is to determine if there exists any assignment to these n variables that satisfies the formula. *3-SAT* is a well-known NP-complete problem.

In the *NAE 3-SAT* problem, we are again given a *3-CNF* formula with m clauses and n variables. But now the goal is to decide if there exists an assignment to these n variables such that in each clause, there is at least one true literal and at least one false literal.

Use a reduction from the *3-SAT* problem to prove that *NAE 3-SAT* is NP-hard. **(10 points)**