Assignment 3, COMP 252, January 30, 2024. Due Feb 9, 2024, 10 pm.

Exercise 1. Decision tree lower bound. Our expert biologist (our oracle) can determine whether two monkeys are from the same subspecies or not. The input consists of n monkeys that are either proboscis monkeys, popa langurs or ollala brother's titis. Your mission is to group them. The output of your algorithm should be in the form of up to three sets of identical monkeys. Complexity is measured in terms of the number of times the biologist is consulted.

(i) Give a simple algorithm for this problem and an upper bound on the complexity. For $n = 5$, your answer should be 7.

(ii) Show that all leaves in the decision tree must be different (i.e., correspond to different outputs).

(iii) Using a decision tree-based method, show that the lower bound for this problem is 6 when $n = 5$.

(iv) For general n, show that the decision tree lower bound is at least $an - b$ and at most $an + b$ for $n \geq n_0$, for some positive numbers a, b, n_0. Determine suitable values for these numbers. Compare this with your answer in (i) and conclude that the decision tree bound for this problem is rather weak.

Exercise 2. Adversarial lower bound. Continuation of the previous exercise. Using the method of adversaries, show that the lower bound for this problem is 7 when $n = 5$. Not part of this exercise, but a challenge for the adventurous: determine a lower bound for general n that matches the upper bound obtained in part (i) of exercise 1 above.

Exercise 3. Lower bounds by the method of adversaries, and an algorithm. Let the data consist of an $n \times n \times n$ matrix of different integers that are sorted along all three axes from small to large. We have a ternary comparison oracle at our disposal. For a given integer x, we are asked to verify whether x is an element of the matrix. The only comparisons that are allowed are between x and matrix elements. Hint: try this first for an $n \times n$ matrix.

(i) Using the method of adversaries, derive a lower bound (called $L(n)$) for the worst-case number of times the oracle has to be consulted to perform a successful search for x. (I am not telling you what $L(n)$ is, and don't ask me during my office hours.)

(ii) Give an algorithm that takes worst case time $O(L(n))$.