1. In a simplified form of the game mastermind there is a hidden sequence $H = (c_1, \ldots, c_k)$ of k colored pegs. There are C different possible colors. Colors can be repeated in the hidden sequence. The game consistes of repeated rounds. To start a round the guesser gives the hider a sequece $G = (g_1, \ldots, g_k)$ of k colors. The hider then tells the guesser how many of the guesses were correct, that is, the number of indices j such that $h_j = g_j$. This ends a round.

   Compute a lower bound as a function of k and C on the number of the number rounds required by the guesser to guarantee that he/she will solve the puzzle. A typical values of $k = 4$ and $C = 6$; what do you get for your lower bound in this case.

   †

2. In the game mastermind there is a hidden sequence $H = (c_1, \ldots, c_k)$ of k colored pegs. There are C different possible colors. Colors can be repeated in the hidden sequence. The game consistes of repeated rounds. To start a round the guesser gives the hider a sequece $G = (g_1, \ldots, g_k)$ of k colors. The hider then tells the guesser how many of the guesses were correct, that is, the number of indices j such that $h_j = g_j$. In addition, the hider tells the guess how many colors are correct, but are in the wrong position (think of H and G as being multi-sets and the hider tells the guesser the cardinality of the multi-set intersection of H and G). This ends a round.

   Compute a lower bound as a function of k and C on the number of the number rounds required by the guesser to guarantee that he/she will solve the puzzle. A typical values of $k = 4$ and $C = 6$; what do you get for your lower bound in this case.

   ††

3. Prove that computing the OR of n bits requires $\Omega(\log n)$ steps on a EREW PRAM, independent of the number of processors used.

   ††

4. Consider the following situation. You have two workstations A and B connected by a communication line. The workstation A is initially given an n bit integer x. The workstation B is initially given an n bit integer y. The goal of the two workstations is to communicate over the line in such a way that they both know the number of bits that are 1 in x plus the number of bits that are 1 in y. Show that the number of bits sent across the line must be $\Omega(\log n)$.

   ††

5. Consider the following situation. You have two workstations A and B connected by a communication line. The workstation A is initially given an n bit integer x. The workstation B is initially given an n bit integer y. The goal of the two workstations is to communicate over the line in such a way that they both know the bit-wise exclusive-or of x and y. Show that the number of bits sent across the line must be $\Omega(n)$.

   ††

6. You goal is to find a counterfeit coin among a group of 9 coins. Counterfeit coins are lighter than real coins. You know that exactly one of the 9 coins is counterfeit. To help you decide which coin is legitimate you have a pan balance. A pan balance functions in the following manner. You can give the pan balance any two disjoint subcollections, say S_1 and S_2, of the coins. Let $|S_1|$ and $|S_2|$ be the cumulative weight of the coins in S_1 and S_2, respectively. The pan balance then determines whether $|S_1| < |S_2|$, $|S_1| = |S_2|$, or $|S_1| > |S_2|$. Show how to solve this problem in two weighings on the pan balance.

   HINT: This is not a lower bound problem. It's purpose is to build you intution.

   †

7. You goal is to find a counterfeit coin among a group of n coins. Counterfeit coins are lighter than real coins. You know that exactly one of the n coins is counterfeit. To help you decide which coin is legitimate you have a pan balance. A pan balance functions in the following manner. You can give the pan balance any two disjoint subcollections, say S_1 and S_2, of the coins. Let $|S_1|$ and $|S_2|$ be the cumulative weight of the coins in S_1 and S_2, respectively. The pan balance then determines whether $|S_1| < |S_2|$, $|S_1| = |S_2|$, or $|S_1| > |S_2|$. Show that solving this problem requires at least $\lceil \log_3 n \rceil$ weighings on the pan balance.

   †

8. Show that there is no comparison based sorting algorithm who running time is linear for at least half of the $n!$ inputs of length n.

   ††

9. Show that $2n - 1$ comparisons are necessary in the worst case to merge two sorted lists of length n.

   ††

10. In the broadcast gossip problem there are n workstations. There workstations may be initially programmed in any way that you like. After they are programmed, an arbitary k of them are given an integer. Assume time is divided into unit slots and that a workstation can transmit its integer in a unit slot. Workstations can only communicate via broadcast. If more than one workstation tries to broadcast in a time slot then all the workstations detect interference (that is, they don't get the message, but they can tell that more than two workstations attempted to broadcast). If only one workstation broadcasts in a time slot then it succeeds and all workstations hear the broadcast. If only no workstation broadcasts in a time slot then this can be detected by all workstations.

    Here assume that the workstations goal is for all workstations to successfully transmit their messages. Explain how to solve this problem in time n.

    HINT: This is not a lower bound question, it is just a warm-up question to understand the problem.

    †

11. Consider the broadcast problem where the goal is for at least one workstation to successfully transmit its message. Show how to solve this problem in time $O(\log n)$.

    HINT: This is not a lower bound question, it is just a warm-up question to understand the problem.

    ††

12. Consider the broadcast problem where the goal is for all workstations to successfully transmit their messages. Show how to solve this problem in time $O(k \log n)$.

    HINT: This is not a lower bound question, it is just a warm-up question to understand the problem.

    ††

13. Consider the broadcast problem where the goal is for all workstations to successfully transmit their messages and all workstations know a priori that $k = 2$. That is, the workstations can be programmed under the assumption that $k = 2$. Show to solve this problem requires time $\Omega(\log n)$.

    ††

14. Consider the broadcast problem where the goal is for all workstations to successfully transmit their messages. Show to solve thie problem requires time $\Omega(k \log \frac{n}{k})$.

    HINT: The math gets a bit involved here. The first fact you need is that $x!$ can be approximated well by $\sqrt{2\pi x}(\frac{x}{e})^x$, where e is the base of the natural logarithm. The second fact you need is that $(1 + x/y)^y$ can be approximated well by e^x when y is large.

    † † †

15. Consider the following problem. You start with one zyglit in a petri dish. Zyglit's reproduce asexually by splitting into two Zyglits (the original Zyglit disappears in this process). The genetic material of the two resulting Zyglits is subject to mutation, and are not identical to either siblings or parents. You are given a petri dish with n Zyglits (that are the leaves of the family) tree. Your goal is to reproduce the family tree. To help you you can perform a DNA check on any three Zyglits, say z_i, z_j and z_k that will determine which pair of Zyglits (among the three possible pairs z_i and z_j, z_i and z_k, and z_j and z_k) have the lowest common ancestor in the tree.

    Show that to recompute the tree requires $\Omega(n \log n)$ DNA checks.

    ††

16. You are given a collection of n VLSI chips. Up to $2n/3$ of the chips may not be reliable. There is jig that allows you to test pairs of chips. Say you are testing a pair X and Y of chips. If X is good then X will accurately report on whether Y is good or not. If X is bad, X may say that Y is good, or X may say that Y is bad (independent of whether Y is good or bad). Show that it is not possible to identify a good chip with surity under this setting.

    ††

17. Consider the element uniqueness problem. The input is n reals $x_1, \ldots, x_n$. The question is does there exist an i and j, with $i \neq j$, with that $x_i = x_j$, that is, are there two numbers in the input that are identical. Show that the information theoretic lower bound on the worst case time complexity for this problem is $\Omega(n \log n)$.

    HINT:

    (a) Think of the input as a point $(x_1, \ldots, x_n)$ as a point in n-dimensional space R^n. Let U be the portion of R^n where no pair of coordinates are the same.

    (b) Show that U consists of $n!$ disjoint regions. Perhaps its easiest to first consider $n = 2$, then $n = 3$, and then you should see it.

    (c) Assume that the algorithm A has made k comparisons. Let A_k be the region of R_n consistent with the answers that A has received to date. For example if $n = 4$, assume that in its first $k = 3$ comparisons A learned that $x_1 < x_4$, $x_2 > x_3$, and $x_1 > x_4$, then A_k would be exactly thost points (x_1, x_2, x_3, x_4) where $x_1 < x_4$ and $x_2 > x_3$ and $x_1 > x_4$. Show that it must always be the case the A_k is convex, that is, if s and t are points in A_k, then the line segment from s to t is in A_k.

    (d) Conclude that at least $\Omega(n \log n)$ comparisons are needed by any algorithm for element uniqueness.

    † † †

18. Let $x_1, \ldots, x_n$ be a collection of reals. Let σ be the permutation that sorts this collection, that is, $x_{\sigma(1)} \leq x_{\sigma(2)} \leq \ldots x_{\sigma(n)}$. Then the *maximum gap* is defined as the i, $1 \leq i \leq n-1$ that maximizes $x_{\sigma(i+1)} - x_{\sigma(i)}$, that is, this is the maximum gap between any two consecutive numbers in the sorted list. Show that the information theoretic lower bound for computing the maximum gap is $\Omega(n \log n)$.

    † † †

19. Show how to compute maximum gap in time $O(n)$ given that you can in constant time take the floor of a real number.

    † † †

20. You have n workstations on the internet. Each workstation knows the indentify of the other workstations and knows how to send messages to each of the other workstations. You know a priori that 40been infiltrated by some hacker and may function in arbitrarily malicious ways. The goal of the remaining 60value of some bit (the value of the bit doesn't matter, all we need is that all of the good workstations agree on the value). Note that a workstation knows whether or not it is good, but it has no idea which other workstations are good. Assume that all sent message are eventually delivered,

although they may be delayed arbitrarily long. Show that there is no algorithm that will allow the good workstations to agree on a bit.

HINT: As the rating suggests, this should not be attempted by the faint of heart. This is a **very** hard problem.

† † ††