CS4445/9554 Analysis of Algorithms II

Fourth Assignment. Optional
Due date: December 8 in class.

1. Let A be an array containing n integer values. Consider the problem of deciding whether there is a value that appears in at least one third of the entries of array A. For example, for the following array

2	1	5	2	8	2	2	7	5
1	2	3	4	5	6	7	8	9

   The value 2 appears in at least one third of the entries.

   - (10 marks) Write a **randomized algorithm** that receives as input an array A of size n and decides whether there is a value x appearing at least $\frac{n}{3}$ times in A. If such an element exists, then the algorithm must return the value `true`, and if such an element does not exist, the algorithm must return the value `false`. You might assume that n is a multiple of 3. If more than one value appears in at least one third of the entries of A your algorithm must return the value `true`.

     Your algorithm must give the wrong answer with probability at most $\left(\frac{1}{3}\right)^{100}$, and it must run in $O(n)$ time.
   - (20 marks) Prove that your algorithm gives the wrong answer with at most the above probability.
   - (5 marks) Compute the time complexity of your algorithm. Assume that selecting a value from a set of m elements requires $O(\log m)$ time.

   (Hint. Design first an algorithm that: (a) when there is a value x appearing at least $n/3$ times in A, the algorithm finds it with probability at least $1/3$, and (b) if such a value x does not exist, the algorithm always returns the value `false`.)

2. Consider a communications network with total bandwidth B. The administrator of the network receives requests from clients asking for the use of some fraction of the bandwidth. Each request consists of a pair of numbers (b, p), where b is the bandwidth required, and p is the price that the client is willing to pay. When a request is received the administrator must decide right away (without knowing which requests will arrive in the future) whether it should be accepted or rejected. If the request is accepted, then the bandwidth is allocated to the client for the rest of the day. At the end of the day the bandwidth is freed, so that at the beginning of the following day the whole bandwidth B is again available. The administrator's goal is to maximize the daily profit without exceeding the bandwidth of the network.

   - (10 marks) Write an **online algorithm** with constant competitive ratio for the above problem.
   - (20 marks) Compute the competitive ratio of your algorithm.

     Assume that for each request (b, p), the ratio p/b is at least 1 and at most α, where α is a constant value (i.e. a client is willing to pay between one and α dollars per unit of bandwidth used). Moreover, a client never requests to use more than half of the total bandwidth, or in other words, for each request (b, p), $b \leq B/2$.

   (Hint. A very simple strategy ensures that either (i) all requests are accepted, or (ii) at least half of the total network's bandwidth is used. What is the maximum profit that can be achieved?)

3. (25 marks) In class we will study the *paging problem*: Given a cache of size k and a request sequence $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$, where each σ_i is a request for some page p_i, the problem is to decide which pages to maintain in the cache so that the number of page faults is minimized. A page fault happens when the requested page σ_i is not in the cache. We assume that all the pages have size 1, so exactly k pages can be stored in the cache. We will learn in class that the 1-bit Least Recently Used (1-bit LRU) strategy has competitive ratio k.

   Consider now the (h, k)-*paging problem*, which is identical to the above one, except that now we assume that the optimum algorithm has a cache of size $h < k$. Compute the competitive ratio c of the 1-bit

LRU algorithm for this problem. Note that $c = SOL(k)/OPT(h)$, where $SOL(k)$ is the value of the solution produced by the 1-bit LRU algorithm using a cache of size k, and $OPT(h)$ is the value of the solution produced by an optimum algorithm that uses a cache of size h.

The competitive ratio depends on k and h. (Hint. Use a similar analysis as the one we will use in class, but modify the part of the analysis where we count the number of page faults for the optimum solution.)

4. The exact subset sum problem is: given a set $S = \{s_1, s_2, \ldots, s_n\}$ of n integer, positive values, and a target value T, find a subset $\mathcal{R} \subseteq S$ such that $\displaystyle\sum_{s_i \in \mathcal{R}} s_i = T$. This problem is NP-hard.

   Peggy claims to know a solution $\mathcal{R}$ for a large instance S, T of the exact subset sum problem, and she wants to convince Victor that she knows $\mathcal{R}$, but she does not want to reveal what $\mathcal{R}$ is. She proposes the following "zero knowledge" protocol.

   - Peggy chooses a random permutation π for the values s_i. This means that she creates a new set $\bar{S}$ that contains the same values as S, but in a different order, $\bar{S} = \{\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_n\}$, where $\bar{s}_i = s_{\pi(i)}$. For example, let $S = \{4, 6, 12, 8\}$ and π be $\pi(1) = 3$, $\pi(2) = 2$, $\pi(3) = 4$ and $\pi(4) = 1$, then set $\bar{S}$ is $\bar{S} = \{12, 6, 8, 4\}$.
   - Then she picks n random numbers $r_1, \ldots r_n$, and she creates a new instance of the exact subset sum problem S', T', with $S' = \{s'_1, s'_2, \ldots s'_n\}$, where $s'_i = s_{\pi(i)} \times r_i$, and $T' = \displaystyle\sum_{s_{\pi(i)} \in \mathcal{R}} s'_i$.
   - She sends S, T and S', T' to Victor.
   - Victor flips a coin, and asks Peggy to either
     - show a solution for problem S', T', or
     - show that S' was obtained from S by revealing the random multipliers r_i and the permutation π.

   They repeat the protocol 100 times. If Peggy correctly answers all the questions then Victor is convinced that Peggy knows the solution $\mathcal{R}$, otherwise he knows that she is lying.

   - (5 marks) Argue that Victor does not learn anything about the solution $\mathcal{R}$ from Peggy.
   - (5 marks) Show that Peggy can fool Victor. (Hint. Peggy can cheat by not selecting the numbers r_i randomly. If these numbers are chosen properly then she can answer all of Victor's questions even if she does not know the solution $\mathcal{R}$. Show how to choose these numbers.)