1 Hashing (33 points)

Let $H = \{h_1, h_2, \ldots \}$ be a collection of hash functions, where $h_i : U \to \{0, 1, \ldots, M - 1\}$ for every $i$ and we assume that $|U| = 2^u$ and that $M = 2^b$ (the same setup as in class when we designed a universal hash family). Recall that $H$ is a universal hash family if $\Pr_{h \sim H}[h(x) = h(y)] \leq 1/M$ for all $x, y \in U$.

Consider the following, slightly different definition. We say that $H$ is a 2-universal hash family if $\Pr_{h \sim H}[h(x) = a \land h(y) = b] \leq 1/M^2$ for all $x, y \in U$ with $x \neq y$ and $a, b \in \{0, 1, \ldots, M - 1\}$.

(a) (11 points) Prove that any 2-universal hash family is also a universal hash family.

(b) (11 points) Prove that for every $u$ and $b$ with $u > b \geq 1$ there is some universal hash family from $U$ to $\{0, 1, \ldots, M - 1\}$ (with $|U| = 2^u$ and $M = 2^b$) which is not a 2-universal hash family. Hint: think about the constructions from class and the textbook.

(c) (11 points) Give a universal hash family from $U = \{0, 1, 2, 3, 4, 5, 6, 7\}$ to $\{0, 1\}$ that contains at most four functions (and prove it is universal). Is this also a 2-universal hash family? Why or why not?

2 $k$-th largest elements (33 points)

You are given an array $a_1, a_2, \ldots, a_n$ and an integer $k \in [n]$ (recall that $[n] = \{1, 2, \ldots, n\}$). For each $i \in [n + 1 - k]$, let $b_i$ be the $k$-th largest element in $a_1, \ldots, a_{k+i-1}$.

(a) (17 points) Design an algorithm which outputs the array $b_1, \ldots, b_{n+1-k}$. The running time should be $O(n \log k)$ (you will prove this in part (b)).

(b) (16 points) Prove the correctness and the running time for your algorithm.

3 Union-Find (34 points)

Suppose you are given a collection of trees representing a partition of the set $\{1, 2, \ldots, n\}$ into disjoint subsets. As in class, these trees are represented by having every node have a pointer to its parent (and not to its children). However, you have no idea how these trees were constructed: they might not have been constructed using union by rank and path compression. You are also given an array $\text{node}[1 \ldots n]$, where $\text{node}[i]$ is a pointer to the tree node containing element $i$. Your task is to create a new array $\text{label}[1 \ldots n]$ using the following algorithm:
LabelEverything():
    For i = 1 to n:
        label[i] = Find(node[i]);

(a) (17 points) What is the worst-case running time of LabelEverything() if we implement Find() without path compression? Use \( \Theta(\cdot) \) notation, and prove your answer.

(b) (17 points) Prove that if we implement Find() using path compression, then LabelEverything runs in \( O(n) \) time in the worst case.