1 Hashing (50 points)

Let $H = \{h_1, h_2, \ldots\}$ be a collection of hash functions, where $h_i : U \rightarrow \{0, 1, \ldots, M - 1\}$ 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) (15 points) Prove that any 2-universal hash family is also a universal hash family.

(b) (15 points) Prove that there is some universal hash family which is not a 2-universal hash family. Hint: consider the matrix method from class.

(c) (20 points) We talked in class (very briefly) about uses of hashing in cryptography. Let’s do an example. Suppose that Alice and Bob secretly choose hash function $h$ uniformly at random from a 2-universal hash family $H$. Later, Alice sends Bob a message $m$ and an authentication tag $t = h(m)$ (so the total message sent is $(m, t)$). An adversary, Eve, intercepts $(m, t)$, and tries to fool Bob by replacing it with a different pair $(m', t')$. Eve’s only goal is to find such a pair where $m' \neq m$ and $t' = h(m')$ (she’s not trying to replace the true message with any particular message, just with something different that still passes the hashing test).

Prove that the probability that Eve produces an $(m', t')$ with $t' = h(m')$ and $m' \neq m$ is at most $1/M$, even if Eve knows $H$ and has unlimited computational power (note that Eve does not know $h$, but does know that $h(m) = t$).
2 Union-Find (50 points)

In this problem we’ll consider what happens if we change our Union-Find data structure to not use path compression. We will still use union-by-rank, but Find operations will no longer compress the tree. More formally, consider the following tree-based data structure. Every element has a parent pointer and a rank value.

- **Make-Set**($x$): Set $x \rightarrow parent := x$ and set $x \rightarrow rank := 0$.
- **Find**($x$): If $x \rightarrow parent == x$ then return $x$. Else return $Find(x \rightarrow parent)$.
- **Union**($x, y$):
  - Let $w := Find(x)$ and let $z := Find(y)$.
  - If ($w \rightarrow rank$) $\geq$ ($z \rightarrow rank$) then set $z \rightarrow parent := w$, else set $w \rightarrow parent := z$.
  - If ($w \rightarrow rank$) == ($z \rightarrow rank$), set ($w \rightarrow rank$) := ($w \rightarrow rank$) + 1

In this problem we will analyze the running time of this variation.

(a) (20 points) Recall that the height of any node $x$ is the maximum over all of the descendants of $x$ of the length of the path from $x$ to that descendant. Prove that for every node $x$, the rank of $x$ is always equal to the height of $x$.

(b) (10 points) Prove that if $x$ has rank $r$, then there are at least $2^r$ elements in the subtree rooted at $x$ (we did this in class for the more complicated data structure which uses path compression, but now you should do it for this version without path compression).

(c) (20 points) Using the previous two parts, prove that every operation (Make-Set, Union, and Find) takes only $O(\log n)$ time (where $n$ is the number of elements, i.e., the number of Make-Set operations).