Problem 1

In this problem you will develop an analysis for QuickSelect similar to that for QuickSort covered in lecture. Suppose we call QuickSelect(\(A, k\)), to find the \(k\)th smallest element in an array \(A[1..n]\) of size \(n\). Assume the elements of \(A\) are distinct. Note the running time of QuickSelect is proportional to the number of comparisons. Thus if we let \(X_{i,j}\) be a random variable which is 1 if \(i\)th smallest item and \(j\)th smallest item are ever compared throughout the execution of QuickSelect(\(A, k\)), then the running time is proportional to \(\sum_{i<j} X_{i,j}\).

(a) (6 points) For \(i < j\), give an exact expression for \(E X_{i,j}\) in terms of \(i, j, k, n\). You may need to employ case analysis.

(b) (4 points) Using (a), show that \(E \sum_{i<j} X_{i,j} = O(n)\).

Problem 2

An \(m \times n\) matrix is said to be a Diagonally Down-and-right Repeating (DDR) matrix if all its diagonals that go down and to the right are constant. For example, the \(4 \times 5\) matrix

\[
\begin{bmatrix}
1 & 4 & 2 & 9 & 8 \\
-3 & 1 & 4 & 2 & 9 \\
0 & -3 & 1 & 4 & 2 \\
1 & 0 & -3 & 1 & 4
\end{bmatrix}
\]

is DDR. In general, knowing just the first column and first row in a DDR matrix \(A\) fully specifies all other entries in \(A\), since \(A_{i,j} = A_{i-1,j-1}\) for \(i, j > 1\).

(a) (3 points) For \(x \in \{0, 1\}^n\) and \(A \in \{0, 1\}^{m \times n}\) a DDR matrix for \(m \leq n\), show that \(Ax\) can be computed quickly, where all addition and multiplication is performed modulo 2. Full credit is given for \(O(n \log m)\) running time. You should assume that arithmetic operations on complex numbers can be done in constant time.
(b) (7 points) For $A$ as in the previous problem part, define $h_A(x) = Ax$, again where addition and multiplication in this matrix-vector product are done mod 2. Let $\mathcal{H}$ then denote the family of hash functions $\{h_A : A \in \{0,1\}^{m \times n}, A \text{ DDR}\}$ mapping $\{0,1\}^n$ into $\{0,1\}^m$. Show that $\mathcal{H}$ is a universal hash family.

**Problem 3**

It is a fact, which you may take for granted without proof, that any positive integer has a unique factorization into the product of primes. For example, $8 = 2^3$, and $550 = 2 \times 5^2 \times 11$. In general, we can write for any positive integer $n$, $n = \prod_{i=1}^{r(n)} p_i^{d_i}$ for some primes $p_1 < p_2 < \ldots < p_{r(n)}$, and each $d_i \geq 1$ an integer. Here $r(n)$ denotes the number of distinct prime divisors of $n$.

(a) (5 points) Show $r(n) = O(\log n)$.

(b) (5 points) Show $r(n) = O(\log n / \log \log n)$. **Hint:** $(k/2)^{k/2} \leq k! \leq k^k$ for integer $k \geq 1$.

**Problem 4: Programming Problem**

Solve COUNTPRODUCTS on the programming server (https://cs124.seas.harvard.edu).

**Hint:** You may find the results of Problem 3 useful. You may also use lists of large primes found online; for example, we found https://primes.utm.edu/lists/small/millions/ useful in writing the staff solutions.

It may also help to know the following two facts, which you are not required to prove (but feel free to think about why these facts hold): for any positive integers $a, b, p$,

- $(a + b) \% p = ((a \% p) + (b \% p)) \% p$
- $(a \times b) \% p = ((a \% p) \times (b \% p)) \% p$