Problem 1

Consider the problem of deciding whether two integer *multisets* S_1 and S_2 are identical (that is, each integer occurs the same number of times in both sets). This problem can be solved by sorting the two sets in $O(n \log n)$ time, where $n = |S_1| = |S_2|$. In this question, you will devise 2 faster randomized algorithms for this problem.

You can assume that the multisets S_i only have integers of bit complexity w, and that integer operations of $O(w)$-bit integers can be executed in $O(1)$ time (RAM model), and that a prime with $O(w)$-bits can be found in $O(n)$ time.

1. Use polynomial identity testing to give a $O(n)$ time algorithm for the problem above.
2. Use hashing to give a $O(n)$ time algorithm for the problem above.

Your algorithm for both parts should succeed with probability $\geq 2/3$.

Problem 2

Compute the mixing time (both upper bounds and lower bounds) of the dumbbell graph: the graph on $2n$ nodes that consists of two complete graphs on n nodes joined by a single edge.

Hint: you can solve this by using elementary probability calculations

Problem 3

Suppose someone searches a keyword (like "car") and we would like to identify the webpages that are the most relevant for this keyword and the webpages that are the most reliable sources for this keyword (a page is a reliable source if it points to many of the most relevant pages).

First we identify the pages with this keyword and ignore all other pages. Then we run the following ranking algorithm on the remaining pages.

- Each vertex corresponds to a remaining page, and there is a directed edge from page i to page j if there is a link from page i to page j. Call this directed graph $G(V, E)$, where $n = |V|$.
- For each vertex i, we have two values $s(i)$ and $r(i)$, where intendedly $r(i)$ represents how relevant is this page and $s(i)$ represents how reliable it is as a source (the larger the values the better).
- We start from some arbitrary initial values, say $s(i) = 1/n$ for all i, as we have no idea of their relevance at the beginning.
- At each step, we update s and r (where s and r are vectors whose i^{th} entries are $s(i)$ and $r(i)$) as follows:
- First we update
$$r(i) = \sum_{j:(j,i)\in E} s(j)$$
for all i, as a page is more relevant if it is linked by many reliable sources.
- Then we update
$$s(i) = \sum_{j:(i,j)\in E} r(j)$$
for all i (using the just updated values $r(j)$), as a page is a more reliable source if it points to many relevant pages.
- To keep the values small, we let $R = \sum_{i=1}^{n} r(i)$ and $S = \sum_{i=1}^{n} s(i)$, and divide each $s(i)$ by S and each $r(i)$ by R.
- We repeat this step for many times to refine the values.

Let $s, r \in \mathbb{R}^n$ be the vectors of the s and r values.

1. Give a matrix formulation for computing s and r.
2. Suppose G is weakly connected (that is, when we ignore the direction of the edges, the underlying undirected graph is connected) and there is a self-loop at each vertex. Prove that there is a unique limiting s and a unique limiting r for any initial s as long as $s \geq 0$ and $s \neq 0$.

Hint: You may use the Perron-Frobenius theorem which states that for any irreducible matrix, there is a unique positive eigenvalue with maximum absolute value and the entries of the corresponding eigenvector are all positive.

Problem 4

Let $S = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ and $T = \{x \in \mathbb{R}^n \mid Bx \leq c\}$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $B \in \mathbb{R}^{t \times n}$ and $c \in \mathbb{R}^t$. Given A, B, b, c as inputs, give a polynomial time algorithm for the problem of checking whether $S \subset T$.

Hint: Linear Programming

Problem 5

Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.

1. Each expert begins with weight 1.
2. We predict the result of a weighted-majority vote of the experts.
3. If an expert makes a mistake, we penalize it by dividing its weight by 2, *but only if its weight was at least* $1/4$ *of the* ***average weight*** *of experts.*

Prove that in any contiguous block of trials (e.g., the 51st day through the 77th day), the number of mistakes made by the algorithm is at most $O(m + \log n)$, where m is the number of mistakes made by the best expert *in that block*, and n is the total number of experts.

Problem 6

Consider the maximum flow problem from s to t on a *directed graph*, where each edge has capacity **one**. A fractional s-t flow solution with value k is an assignment of each edge e to a fractional value $x(e) \in \mathbb{R}$, satisfying

$$\sum_{e \in \delta^{out}(s)} x(e) = \sum_{e \in \delta^{in}(t)} x(e) = k$$

$$\sum_{e \in \delta^{in}(s)} x(e) = \sum_{e \in \delta^{out}(t)} x(e) = 0$$

$$\sum_{e \in \delta^{in}(v)} x(e) = \sum_{e \in \delta^{out}(v)} x(e) \quad \forall v \in V \setminus \{s, t\}$$

$$0 \leq x(e) \leq 1, \quad \forall e \in E$$

Use the multiplicative weights update method to solve this Linear Program by reducing the flow problem to the problem of finding shortest paths between s and t. Analyze the convergence rate and total complexity of your algorithm to compute a flow of value $k \cdot (1 - \epsilon)$ for any $\epsilon > 0$.