HW 5

Due: October 16th, 2025

Problem 1:

For this assignment we will look at two algorithms from class in a different model that allows for deletions. We are going to use a model similar to the turnstile model defined in class, but that allows for negative values for v.

General Turnstile Model: The input is a sequence of elements $e = (i, v)$ where i is an item, and v is the change to the number of that item. In contrast to the model in class, we will allow v to be negative (representing for example, deleting or removing the item), and the total count for an item at the end of a stream can also be negative.

Part A.

Algorithm 1 Estimate Distinct Elements via Minimum Hash

Input: A data stream S.
Input: A uniform hash function $h : \mathcal{U} \to [0, 1]$, where $\mathcal{U}$ is the universe of elements.
Output: An estimate $\hat{d}$ of the number of distinct elements in S.

procedure MinHashEstimator(S, h)
  Let $m \leftarrow 1.0$ ▷ Initialize minimum hash value to the maximum possible value
  for all element e in stream S **do**
    $v \leftarrow h(e)$ ▷ Compute the hash of the current element
    if $v < m$ **then**
      $m \leftarrow v$ ▷ Update the minimum hash value if the new one is smaller
    end if
  end for
  $\hat{d} \leftarrow \frac{1}{m} - 1$ ▷ Calculate the final estimate from the minimum value
  return $\hat{d}$
end procedure

Why would the algorithm given for distinct elements not work in a model with deletions?

Part B.

Algorithm 2 Count-Min Sketch

Setup:
Let k be the number of hash functions.
Let T be the size of the counter arrays.
Choose k independent, uniform hash functions $h_1, h_2, \ldots, h_k : \mathcal{U} \to \{1, \ldots, T\}$.
Initialize a 2D array (or k 1D arrays) of counters $C[1..k][1..T]$ to all zeros.

```
1: procedure UPDATE(element e)
2:     for i ← 1 to k do
3:         j ← h_i(e)            ▷ Get the bucket index from the i-th hash function
4:         C[i][j] ← C[i][j] + 1          ▷ Increment the corresponding counter
5:     end for
6: end procedure

7: procedure ESTIMATE(element q)
8:     Let f̂_q ← ∞                    ▷ Initialize the estimate to a large value
9:     for i ← 1 to k do
10:        j ← h_i(q)                ▷ Get the bucket index for the query element
11:        f̂_q ← min(f̂_q, C[i][j])   ▷ Update estimate with the minimum counter value
12:    end for
13:    return f̂_q
14: end procedure
```

Describe how you would alter the algorithm for point query estimation (ESTIMATE above) to allow for deletions. Prove that the modified algorithm again provides an unbiased estimate of the total value for the query element.

Problem 2:

Let $G = (V, E)$ be an undirected graph. An **independent set** is a subset of vertices $S \subseteq V$ such that no two vertices in S are adjacent. The maximum independent set problem, which is NP-hard, seeks to find an independent set of maximum cardinality.

This problem can be formulated as an integer program. For each vertex $v \in V$, let x_v be a binary variable such that $x_v = 1$ if v is in the independent set and $x_v = 0$ otherwise. A basic linear program relaxation for the maximum

independent set problem is:

$$\begin{aligned} \text{Maximize} \quad & \sum_{v \in V} x_v & \\ \text{Subject to:} \quad & x_u + x_v \leq 1 & \forall \{u, v\} \in E \\ & 0 \leq x_v \leq 1 & \forall v \in V \end{aligned}$$

Observe that indeed, if we additionally constrained x_v to only allow integer values, the corresponding *integer* linear program has precisely the vectors $\vec{x}$ corresponding to maximum size independent sets as solutions. Because we omit this constraint, the above linear program has additional fractional solutions.

This relaxation is often weak. To strengthen it, we can add more valid inequalities. A well-known class of such inequalities are the **odd cycle inequalities**. For any simple cycle C in G with an odd number of vertices, the following inequality is valid for the vectors $\vec{x}$ corresponding to independent sets:

$$\sum_{v \in C} x_v \leq \frac{|C| - 1}{2}$$

The **separation problem** for a class of inequalities is as follows: Given a vector $\vec{x}^* \in \mathbb{R}^V$, either prove that $\vec{x}^*$ satisfies all inequalities in the class, or find one that it violates.

In this problem, you will design and analyze a polynomial-time algorithm for the separation problem for the odd cycle inequalities.

Part A.

First, prove that the odd cycle inequalities are satisfied for the **independent set polytope**, denoted $P(G)$, the convex hull of the vectors $\vec{x}$ corresponding to independent sets in G. That is, supposing that for any independent set S in G we have a weight $\lambda_S \in [0, 1]$ (such that $\sum_S \lambda_S = 1$) and a corresponding vector $\vec{x}(S) \in \{0, 1\}^V$, the convex combination $\vec{y} = \sum_S \lambda_S \vec{x}(S)$ satisfies $\sum_{v \in C} y_v \leq \frac{|C|-1}{2}$ for any odd cycle C. Why must a linear program that has all of the vectors $\vec{x}(S)$ as feasible solutions also include all of $P(G)$ as feasible solutions?

Part B.

Give a dynamic programming algorithm that solves the separation problem for odd cycle inequalities. Prove that the algorithm is correct.

Part C.

Analyze the running time of your algorithm from Part B.