

CS 561: Homework 7

Prof. Jared Saia, University of New Mexico

Remember: You are encouraged to work on the homework in groups, but please observe the “Star Trek” rule from the syllabus.

1. **Count-Min Sketch.** Fix a stream of $T \geq 1$ unit updates. For every item i , let f_i be its true frequency after the stream has been processed and let $\hat{f}_i$ be its Count-Min estimate. Assume that the stream and the item being queried are fixed independently of the random hash functions.

For parameters $0 < \varepsilon, \delta < 1$, use a Count-Min sketch with

$$w = \left\lceil \frac{e}{\varepsilon} \right\rceil \quad \text{and} \quad d = \left\lceil \ln \frac{1}{\delta} \right\rceil,$$

where the d row hash functions are chosen independently and each satisfies $\Pr(h_a(i) = h_a(j)) \leq 1/w$ for distinct items i and j .

- (a) Prove that $\hat{f}_i \geq f_i$ for every item i , regardless of the random choices of the hash functions.
- (b) Fix an item i and one row a . Let Z_a be the number of updates to items other than i that collide with i in row a . Prove that

$$\mathbb{E}[Z_a] \leq \frac{T}{w} \quad \text{and} \quad \Pr(Z_a > \varepsilon T) \leq \frac{1}{e}.$$

Your proof should state where linearity of expectation and Markov’s inequality are used.

- (c) Use independence across rows to prove that, for any fixed item i ,

$$\Pr(\hat{f}_i > f_i + \varepsilon T) \leq \delta.$$

Conclude that, with probability at least $1 - \delta$, this fixed item satisfies

$$f_i \leq \hat{f}_i \leq f_i + \varepsilon T.$$

- (d) State the number of counters used, the time for one update, and the time required to query one item. Give your bounds using ε and δ .

2. **Alternative Activity-Selection Rules.** An activity a is a half-open interval $[s(a), f(a))$, where $s(a) < f(a)$. Thus, two activities that only touch at an endpoint do not conflict. The goal is to select a maximum-cardinality set of pairwise nonconflicting activities.

Consider each of the following recursive algorithms. At each step, the algorithm selects an activity according to the stated rule, discards every activity that conflicts with it, and recurses on the remaining activities. When a rule has a tie, choose the tied activity with smaller start time; then choose smaller finish time, and break any remaining tie arbitrarily.

For each rule, either prove that the algorithm always returns an optimal schedule or give a counterexample and trace the algorithm on your instance.

- (a) Select an activity having shortest duration $f(a) - s(a)$.
- (b) Select the activity having earliest start time.
- (c) Select the activity having latest finish time.
- (d) Select an activity that conflicts with the fewest other remaining activities.

3. **Binary Matrix with Prescribed Sums.** Let $r_1, \dots, r_n$ and $c_1, \dots, c_n$ be nonnegative integers. Assume that there exists an $n \times n$ matrix with entries in $\{0, 1\}$ whose row sums are $r_1, \dots, r_n$ and whose column sums are $c_1, \dots, c_n$.

Consider the following greedy algorithm. It constructs the rows in the order $1, 2, \dots, n$. Just before row i is constructed, let d_j be the number of additional 1's still required in column j . The algorithm places 1's in row i in the r_i columns with largest values of d_j , breaking ties arbitrarily, and places 0's in the other columns. It then decreases the chosen residual demands by 1.

- (a) Suppose that, before constructing row i , the residual instance has a feasible completion. Let G be the set of columns chosen by the greedy rule for row i , and let S be the set of columns receiving a 1 in row i in some feasible completion. If $G \neq S$, choose $p \in G \setminus S$ and $q \in S \setminus G$ such that $d_p \geq d_q$. Prove that some later row has a 1 in column p and a 0 in column q . Show how to exchange two entries in row i and two entries in that later row so that all row and column sums are preserved and row i agrees with the greedy choice in one additional column.
- (b) Use the exchange step repeatedly, followed by induction on the row number, to prove that the greedy algorithm constructs a valid matrix whenever a valid matrix exists.