Problem Set 4

All parts are due Thursday, October 30 at 11:59PM. Please download the .zip archive for this problem set, and refer to the `README.txt` file for instructions on preparing your solutions. Remember, your goal is to communicate. Full credit will be given only to a correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Name: Rishad Rahman

Collaborators: Cheng Wang

Part A

Problem 4-1.

Solution: Hash the n integers into a table. Set a counter to 0 and iterate through the list of integers doing the following for each x:

$\triangleright$ Compute $7x^{-1} \mod p$ and search for it in the hash table.

$\triangleright$ If we hit then we increase the counter by 1 otherwise do nothing.

Return the count at the end.

Correctness: We need to make sure we increment the counter exactly once for every pair (a, b) which satisfies $ab \equiv 7 \mod p$. Since we let a take all the values in the list and check if the *unique* value, by CRT, $b = 7a^{-1} \mod p$ was hashed this is precisely what happens.

Runtime: The runtime is $O(n)$ since we are hashing n integers and doing n searches and computations each taking $O(1)$. The space is $O(n)$ since the hash table is of size $m = O(n)$.

Problem 4-2.

(a) Solution: For an insertion to take more than m probes, each of the first m probes must have failed at finding an empty slot in the hash table. The probability of finding an empty slot is obviously greater than $\frac{1}{2}$ since we never have more than n of the $2n$ slots filled so we have the probability of colliding on a probe is less than or equal to $\frac{1}{2}$ and the probes are independent so after m probes we have a probability less than or equal to 2^{-m} which is an upper bound for any further calculation.

(b) Solution: Substituting $m = 2\log_2 n$ into (a) gives $2^{-2\log_2 n} = \frac{1}{n^2}$.

(c) Solution: We can subtract the complement so the probability is, where p_i is the probability that more than $2\log_2 n$ probes is taken on insert i, $1 - \prod(1 - p_i) \leq 1 - \left(1 - \frac{1}{n^2}\right)^n \leq 1 - (1 - \frac{1}{n}) = \frac{1}{n}$ where we have used Bernoulli's Inequality to show $(1 - \frac{1}{n^2})^n \geq 1 - \frac{1}{n}$.

(d) Solution: We have

$$\mathbb{E}(X) = \sum_{i=0}^{n} i \cdot p(i) = \sum_{i=0}^{2\log_2 n} i \cdot p(i) + \sum_{i=2\log_2 n+1}^{n} i \cdot p(i) \leq$$

$$2\log_2 n \sum_{i=0}^{2\log n} p(i) + n \sum_{i=2\log_2 n+1}^{n} p(i) \leq 2\log_2 n + n \cdot \frac{1}{n} = O(\log_2 n)$$

We have used the fact that the sum of probabilities is less than or equal to 1 for the first part of the summation and that $\sum_{i=2\log_2 n+1}^{n} p(i) = P(X > 2\log_2 n) \leq \frac{1}{n}$ which we showed in (c) for the second part.

Problem 4-3.

Solution: We create an undirected graph of $m + n$ nodes consisting of the m tracks and n participants where an edge connects a node representing a track and a node representing a participant iff the participant has chosen that track. We run a modified version of BFS which runs as follows:

▷Arbitrarily choose one of the track nodes as the start node and mark it with SAT then set a variable d to the value SUN.

▷Repeat the following until BFS terminates

→Every 2 stages/nodes of the BFS we look at the tracks 1 participant away from the previously searched track (by the construction of the graph all sets of three adjacent nodes are of the form track-participant-track or participant-track-participant but our start node was a track so the former applies) so they cannot be on the same day as the previously searched track. We mark them with the value in d then change d to the opposite day. If a node was already marked we see if it checks with the value in d and if it does not we terminate and return FALSE

→If the graph is a union of disjoint graphs then BFS continues starting at a track node on one of the other unexplored graphs (if any)

▷Return TRUE

Correctness: Note that the property we need to keep is that the tracks adjacent to a participant are on different days (if there are two). The algorithm preserves this property as any track following a SAT track and a participant will get a SUN and vice versa since d was changed. Suppose we discovered a node that was already marked SAT and that the value of d is SUN. We must have arrived at that node from a participant that followed a track which was marked SAT since that would cause d to be SUN but then that participant

would be adjacent to two SAT's which we do not allow so the algorithm returns FALSE. Obviously if this never happens we are good so TRUE is returned rightfully if this always passes. Disjoint graphs don't affect this since they translate to different participants with different desired tracks so they are independent.

Runtime: The number of nodes is $m + n \leq 3n = O(n)$ while the number of edges is less than or equal to $2n = O(n)$ and we are doing $O(1)$ computations at each step of the BFS so the total runtime is $O(n)$.

Part B

Submit your implemented python script on alg.csail.mit.edu.

Part C

Indicate whether or not you complete the Piazza poll.

Completed.