This is a “work-through” problem. It is based on the answer to “presentation problem (6)” from last year at https://cse.buffalo.edu/~regan/cse491596/CSE491596ps6preskey.pdf but tries to put the point more cleanly. It is worth 36 extra-credit points.

Problem (1) on Prelim II constructed a deterministic TM that can decide whether a given binary string of any length \( n \) is a palindrome in \( O(\log n) \) space but takes order-\( n^2 \) time to do so. It can be proved that no deterministic log-space TM can do better than \( \Omega(\frac{n^2 \log n}{\log n}) \) time. However, we can build a randomized TM \( M \) that can work in \( O(\log n) \) space and \( O(n) \) time and be correct with small error.

To help smooth the operation of \( M \), we structure inputs \( x \) as follows:

- The possible breakpoint for a palindrome is marked by a special \# character. The input is given in the form \( x = y\#z \) where \( y, z \in \{0, 1\}^* \).
- We only consider inputs where \(|y| = |z| = m\) and where \( m \) has a factor \( k \) that is roughly equal to \( \log m \).
- \( M \) is also given \( k \) ahead of time, so it can keep its workspace usage down to \( k \) and keep track of \( k \) chars at a time while doing a single left-to-right scan of its read-only input tape.

(a) First we keep \( M \) itself deterministic but use distributional analysis. Show how to design \( M \) using the idea of a checksum (can be numerical sum or bitwise XOR) so that

- If \( y = z^R \), then \( M(k, y\#z) \) accepts.
- For all \( y \in \{0, 1\}^m \), the probability over \( z \in \{0, 1\}^m \) that \( y \neq z^R \) but \( M(k, y\#z) \) erroneously accepts is at most \( \frac{1}{2^k} \). (Which, given \( k \approx \log_2 m \), is \( \approx \frac{1}{m} \).)

(b) The issue with (a) is that if \( z \) (or rather, \( z^R \)) happens to give the same “checksum” as \( y \), then we’re cooked. Put another way, simple checksums are reproducible but inflexible: there are false inputs \( x \) that pass the check. So now let us make \( M \) randomized. We will take for granted the existence of a function \( H(r, y) = u \) where \(|r| = |u| = k\) and \(|y| = m\) with the following properties:

- \( H(r, y) \) is computable in \( O(k) \) space and \( O(m) \) time.
- Whenever \( y \neq z \), the probability over \( r \) that \( H(r, y) = H(r, z) \) equals at most some fixed constant \( c \) times \( \frac{1}{2^k} \).

Explain how (a machine computing) \( H(r, y) \) can be used to build a randomized TM \( M'(k, x) \), where \( x \) has the form \( y\#z \) as above, that flips coins to generate a random \( r \in \{0, 1\}^k \) so that for all \( x \):

- If \( x \) is a palindrome then \( M'(k, x) \) accepts regardless of \( r \);
- If not, then the probability over \( r \) that \( M'(k, x) \) accepts is at most \( c \) times \( \frac{1}{2^k} \).

And show that \( M' \) runs in linear time and log space. This is like “having the exam problem’s cake and eating it too”—except for the small chance of error.

---

1One way to do this is to treat \( r \) and size-\( k \) blocks of \( y \) and \( z \) as elements of the finite field \( \mathbb{F}(2^k) \) and make different powers of \( r \) multiply different blocks. The details are messy: it works only for those \( r \) that have long enough periods in the field and makes \( c \) depend on the constant in taking \( k = O(\log m) \). There are “space-efficient universal hash functions” giving \( c = 1 \) but they are harder to describe concretely.