1 Time lower bound for palindromes

Here is a quick writeup of the quadratic lower bound for deciding palindromes on a single-tape Turing machine that I gave in class.

Intuition: the pieces of information that the TM must compare are spread far apart. When the head travels from one character to its match, it can only carry a small amount of information in the local state, so it will have to make many such trips.

Theorem 1 *Any (single-tape) TM that decides PAL uses $\Omega(n^2)$ steps in the worst case.*

Proof: Consider any TM M that decides PAL.

Let $T(n)$ be the maximum number of steps that M takes on an input of length n. We shall show that $T(n)$ is $\Omega(n^2)$.

Define the *crossing sequence at location i on input x*, denoted $C_i(x)$, to be the sequence of states M is in when its head crosses the boundary between tape square i and tape square $i+1$, if the TM is run on input x.

Lemma 2 *Suppose $C_i(x) = C_j(y)$ for two strings $x, y \in PAL$. Let x' be the first i characters of x and let y' be the last $|y| - j$ characters of y. (If $j > |y|$, then $y = \varepsilon$.) Then M must accept the string $x'y'$.*

Proof (sketch): Intuitively, from "the point of view of x'" it doesn't matter what comes later on the tape: all that matters is that it generates the same crossing sequence as the rest of x does. Similarly for y'.

The execution of the Turing machine on input $z = x'y'$ can be chopped up into chunks: we start a new chunk whenever the TM's head crosses the boundary between square i and $i+1$. Call these chunks $E_z^0, E_z^1, \ldots, E_z^m$, where m is the length of the crossing sequence $C_i(x) = C_j(y)$. Similarly, M's execution on input x can be chopped into chunks $E_x^1, E_x^2, \ldots, E_x^m$ according to when the head crosses the boundary between square i and $i+1$ and M's execution on input y can be chopped into chunks $E_y^1, E_y^2, \ldots, E_y^m$ according to when the head crosses the boundary between square j and $j+1$. Then, the sequence of steps M performs during E_z^k will be identical to the steps it performs during E_x^k for all even k and the sequence of steps M performs during E_z^k will be identical to the steps it performs during E_y^k for all odd k. (To make this formal, we would have to do an induction proof on k, and for each k a nested induction proof to show that the sequence of steps within that chunk would be the same in the two executions.)

M accepts both x and y since they are both palindromes. Since $C_i(x) = C_j(y)$, it must halt when the head is in the same "half" of both strings. (*I.e.*, it halts on input x when the head is in the first i squares iff it halts on input y when the head is in the first j squares.) It follows that M will accept $x'y'$: The accepting step of either M's computation on input x or M's computation on input y will have a corresponding step in M's computation on input $x'y'$. So M outputs yes for input $x'y'$. This completes the proof of the lemma. ■

Now, I would like to argue that no two strings can have a crossing sequence in common, to use a counting argument: If $T(n)$ is too small, crossing sequences must be short, and there won't be enough different crossing sequences to assign to all the different strings.

The crossing sequences would have to all be different if $x'y'$ was guaranteed not to be a palindrome. So we will consider a bunch of strings designed so that if you glue together the prefix of one with the suffix of another, the result will not be a palindrome.

Let $L_n = \{wc^{2n}w^R : w \in \{a,b\}^n\}$.

Lemma 3 *If $n+1 \leq i,j \leq 3n$ and $x,y \in L_n$ with $x \neq y$ then $C_i(x) \neq C_j(y)$.*

Proof: Assume $C_i(x) = C_j(y)$. Then $x'y'$ (defined above) consists of the first n characters of x followed by at least 1 c, followed by the last n characters of y. Since $x \neq y$, the first n characters of $x'y'$ are not the same as the last n characters of $x'y'$ reversed. So $x'y'$ is not a palindrome, contradicting the previous lemma, which says M accepts $x'y'$. ■

Now, for each input string in L_n, some crossing sequence (for a position between $n+1$ and $3n$) must have length at most $\frac{T(4n)}{2n}$. (If not, the running time would be more than $T(4n)$ since each element of every crossing sequence corresponds to a step of M.)

Now we can do our counting argument.

Total number of crossing sequences of length at most $\ell = \frac{T(4n)}{2n}$ is $\sum_{i=0}^{\ell} |Q|^i \leq |Q|^{\ell+1}$, where Q is the state set of M.

Each of the 2^n strings in L_n must be assigned at least one of those short crossing sequences and no two can be assigned the same one (by claim above).

So,

$$\begin{aligned} |Q|^{\ell+1} &\geq 2^n \\ \ell + 1 &\geq n \log_{|Q|} 2 \\ \frac{T(4n)}{2n} &\geq n \log_{|Q|} 2 - 1 \\ T(4n) &\geq (2 \log_{|Q|} 2) n^2 - 1 \end{aligned}$$

It follows that $T(n)$ is $\Omega(n^2)$. ■