Time Complexity and P

a Consider this TM for identifying palindromes: M = "On input w: Repeatedly mark the first and the final unmarked characters, *rejecting* if they differ. When fewer than two remain, *accept*." What is the running time $t(n)$ of M? *(Unless specified otherwise, n is the length of the input string, and we do worst-case asymptotic analysis, i.e. use big-O notation.)*

b Design a 2-tape TM which solves the same palindrome problem faster. Prove that it is impossible to do (asymptotically) better than your design.

c As suggested in the previous parts, there are languages which can be decided more quickly by multi-tape TMs than by normal ones. We will now derive a bound on the size of this speedup effect.

Recall that one way to simulate a k-tape TM R using a normal TM M is to split M's tape into segments where each segment represents one of R's tapes, marking one cell in each segment (i.e. virtual tape) as the position of its tape head. Then at each step, read all the virtual tapes to determine the marked symbol in each, and then scan over all the virtual tapes again to make the appropriate update in each. If in any virtual tape we try to write beyond its currently allotted segment, shift everything to its right one space over to make room.

Let R run in time $t(n)$.

i Find an upper bound on how long M's (written) tape contents can ever get while simulating R. *(As in part a, you can give a big-O answer - always assume we're just looking for asymptotic bounds unless specified otherwise.)*

ii Find an upper bound on how long it takes M to simulate one step of R, assuming we do not have to do any of those shift-everything-right operations.

iii Find an upper bound on how long one shift-right operation can take.

iv Find an upper bound on the running time of M.

d (Sipser Definition 7.7) Let $t : \mathbb{N} \to \mathbb{R}^+$ be a function. Define the *time complexity class* $TIME(t(n))$ to be the collection of all languages that are decidable by an $O(t(n))$ time Turing machine.

(Sipser Definition 7.12) P is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine. In other words, $P = \bigcup_k TIME(n^k)$.

Argue that P is 'robust to adding tapes', i.e. if we change the definition to say "multi-tape" instead of "single-tape", we'd still get the same class of languages. Prove that P is closed under complement, union, and intersection. (You may assume that there is at most a polynomial time overhead involved when turning reasonable high-level descriptions of TMs into formal TMs.)

e We will work with the class P for a while, with the rough idea being that problems in P are tractable for computers while problems not in P are not. Discuss the implications of this choice (as opposed to working with finer-grained time classes like $TIME(n^2)$) - in what ways will it make our analysis easier, and how well do you think it reflects reality?

f *(I don't think we'll be using the formal definition of big-O much, but if you finish the rest of the sheet early, here's a practice problem for you to brush up on it.)*

(Sipser Definition 7.2) Let f and g be functions $f, g : \mathbb{N} \to \mathbb{R}^+$. Say that $f(n) = O(g(n))$ if positive integers c and n_0 exist such that for every integer $n \geq n_0$, $f(n) \leq cg(n)$.

We routinely simplify our work in problems involving big-O by making big-O approximations at multiple steps along the way and then combining them. And following Sipser, we'll allow big-O to be used as part of larger expressions, e.g. we might write $f(n) = 2^{O(n)}$, or $f(n) = O(g(n)) + O(h(n))$. We will explore a related question to check that this seems sufficiently well-defined and well-behaved. Prove that if $f(n) = O(t(n))$ and $g(n) = O(r(n))$ then $f(n) + g(n) = O(t(n) + r(n))$.