We define our own Turing Machine to have everything the RAM has (all of its states and tapes and the same alphabet, but with an extra character we call the separator and is denoted by ,) as well as an extra work tape. The idea of this extra tape is to imitate the array A in the RAM and is in the format of $\lfloor i \rfloor, A[i], \lfloor i+1 \rfloor, A[i+1], \lfloor i+2 \rfloor, A[i+2]$ …. The RAM goes through at most $T(n)$ q_{access} states, and the length of any $\lfloor i \rfloor$ is at most $T(n)$ because the RAM makes at most $T(n)$ steps. The length of each $A[i]$ is exactly 1 because it's a character of the machine alphabet. Thus, the length of our extra tape is at most $T(n)(T(n)+1) = O(T(n)^2)$. For each step our new TM makes in the q_{access} state, it accesses our extra tape once to either set or get an $A[i]$, which is $O(T(n)^2)$ in runtime because we're doing string matching aided by the separator character. Thus, our new TM make at most $O(T(n) \cdot T(n)^2) = O(T(n)^3)$ steps.