Exercises on space complexity

Exercise 1: Let $A$ be an algorithm of space complexity $s(n)$. Show that there is an algorithm $A'$ such that
- $L(A) = L(A')$
- $A'$ has space complexity $s'(n) = O(s(n) + \log m)$
- $A'$ does not scan the input tape beyond the boundaries of the input

Proof: We proceed in two steps

1) We prove that on input $x$, there is an algorithm $A_x$ such that
   - $L(A_x) = L(A)$
   - $A_x$ does not scan the input tape beyond location $2^{O(s(n) + \log m)}$ from the input
   - $A_x$ has space complexity $s_x(n) = O(s(n) + \log m)$

This proof is analogous to the one that we did in class to show that a poly-space bounded (N)$\text{TM}$ is equivalent to one that has running time $t(n) \leq C \cdot q(n)$ with $q(n) = O(s(n))$ (where $s(n)$ is a polynomial space bound).

We showed $q(n) = 2 \cdot s(n) + 1$, where $C = |\Gamma| + 1$ (Q1)

In our case:
- $q(n) = \log C \cdot q(n) = O(s(n))$
- we have also the position on the input tape that contributes to the configuration:
  $n - 2^{O(s(n))} = 2 \log \cdot 2^{O(s(n))} = 2 \cdot \log m$
  different configurations at most

Note: $Q\text{TM}$ with running time $t(n) \leq 2^{O(s(n) + \log m)}$ can scan at most $2^{O(s(n) + \log m)}$ cells of the input tape
2) We modify the algorithm $A_1$ in such a way that it does not move beyond the input.

The resulting algorithm $A'_1$ works as follows:

- whenever $A_1$ would move right past the end of the input, $A'_1$ instead:
  - does not move past the end of the input, but maintains a counter on the work tape
  - whenever $A_1$ moves right, the counter is incremented left, decremented

In this way, $A'_1$ can keep track of the position of the input head of $A_1$

Whenever $A_1$ moves back again over the input symbol, $A'_1$ does not update the counter (leaving it to 0)

$A'_1$ operates similarly whenever $A_1$ moves left past the beginning of the input

How much space does the counter use:

Since $A_1$ does not scan the input tape beyond $S_i(n) = 2^{O(n)}$, the counter takes $\log_2 S_i(n) = O(n + \log n)$

Hence, the total space used by $A'_1$ is

$S(n) + O(S(n) + \log 2) = O(S(n) + \log n)$
Exercise 2: Let $A$ be an algorithm of space complexity $S$. Show that there is an algorithm $A'$ such that

- $A'$ computes the same function as $A$, i.e. $A'(w) = A(w)$ for all $w$.
- $A'$ has space complexity $S'(m) = S(m) + O(\log \ell(m))$ where $\ell(m) = \max_{w \in \{0, 1\}^n} A(w)$ is the size of the maximum output for input $w$ of length $n$.
- $A'$ never rewrites on the same location of its output tape.

Proof:

$A'$ proceeds in successive iterations, each time simulating the whole computation of $A$:

- In the $i$-th iteration, $A'$ outputs the $i$-th bit of $A(w)$.

When simulating $A$, in its $i$-th iteration, $A'$ proceeds as follows:

- It does not directly (re)write on the output tape.
- Instead, it maintains on the work tape:
  - A counter $i$ of the next output bit that will be written.
  - A counter $c$ of the bit that $A$ is currently writing.
  - The value of the bit written by $A$ in position $i$.
- When $A$ would write an output bit, $A'$ operates depending on the values of $i$ and $c$:
  - If $i \neq c$, then $A'$ does not output anything.
  - If $i = c$, then $A'$ stores the written bit on its work tape.
- At the end of its simulation, $A'$ outputs the stored bit to the $i$-th position of the output tape.
How much space $S'(n)$ does $A'$ use on the working tape for inputs of length $n$?
- $O(n)$ cells, since it performs the computation of $A$.
- The space for the counters $i$, $i_0$, and $c$.
- $c_0$ and $c_0$ have to count positions on the output tape, and hence will use $\log_2 l(n)$ bits each.

We get that $S'(n) = S(n) + O(\log_2 l(n))$. 