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 $s'(n) = O(s(n) + \log n)$
- $A'$ 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 $A_1$ such that:
- $L(A_1) = L(A)$
- $A_1$ does not scan the input tape beyond location $2^{O(s(n) + \log_2 n)}$ from the input

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

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

In our case:

$$q(n) = 2\log_2 C - q(n) = 2$$

we have also the position on the input tape that contributes to the configuration:

$$\Rightarrow n \cdot 2^{O(s(n))} = 2 \log_2 2^{O(s(n))} = 2^{O(s(n) + \log_2 n)}$$

different configurations at most.

Note: A TM with running time $t(n) \leq 2^{O(s(n) + \log n)}$ can scan at most $2^{O(s(n) + \log n)}$ 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 $D_1(n) = 2^{O(n \log n)}$ the counter reaches $\log_2 D_1(n) = O(n \log n + \log n)$.

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

$D(n) + O(n \log n + \log n) = O(n \log n + \log n)$.
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., $A'(w) = A(w) \forall w \in \{0,1\}^*$
- $A'$ has space complexity $O'(m) = O(m) = O(\log \omega(m))$
- $\omega(m) = \max_{w \in \{0,1\}^m} |A(w)|$ is the size of the maximum output for input $x$ of length $m$
- $A'$ 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 $A(x)$.

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:
  - The counter $i'$ of the next output bit that will be written.
  - The counter $\ell$ 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 $\ell$:
  - If $i' \neq \ell$, then $A'$ does not output anything.
  - If $i' = \ell$, then $A'$ stores the written bit on its worktape.
- 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 \( n \)?

- \( s(n) \) cells, since it performs the computation of \( A \)
- The space for the counters \( i, j \), and \( c \)
- \( c \) and so here to count positions on the output tape, and hence will use \( \log_2 k(n) \) bits each.

We get that \( s'(n) = s(n) + O(\log_2 k(n)) \)