Exercise 1: Let A be an algorithm of space complexity \( s(m) \). Show that there is an algorithm \( A' \) such that:
- \( s'(m) = \mathcal{O}(s(m) + \log m) \)
- \( A' \) does not see 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 \( A' \) such that:
- \( s'(m) = \mathcal{O}(s(m) + \log m) \)
- \( A' \) does not see the input tape beyond location \( 2^{\mathcal{O}(s(m) + \log m)} \) from the input.

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

We showed \( q(m) = 2^{\log C \cdot q(m)} \). Where \( C = \left\lceil \log (1 + |U|) \right\rceil \).

In our case:
- \( q(m) = 2^{\log C \cdot q(m)} = 2^{s(m)} \)
- We have also the position on the input tape that contributes to the configuration:

  \[ m \cdot 2^{\mathcal{O}(s(m))} = 2^{\log 2^{\mathcal{O}(s(m))}} = 2^{\mathcal{O}(s(m) + \log m)} \]

  Different configurations at most

Note: A TM with running time \( t(m) \leq 2^{\mathcal{O}(s(m) + \log m)} \) can see at most \( 2^{\mathcal{O}(s(m) + \log m)} \) 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
  - whenever $A_1$ moves left, the counter is 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 at 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 $d_1(n) = 2^0(n) = n$,

the counter takes $\log_2 d'_1(n) = O(n + \log m)$

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

$O(n) + O(n \cdot \log 2) = O(n + \log m)$
Exercise: 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 $w \in \{0,1\}^*$
- $A'$ has space complexity $S'(n) = S(n) + O(\log l(n))$
  where $l(n) = \max_{w \in \{0,1\}^n} |A(w)|$ is the size of the maximum output for input $x$ 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 $A(x)$.

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
  - 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 worktape
- at the end of its simulation, $A'$ outputs the stored bit to the $i$-th position of its 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 $i^*$ and $c$
- $i^*$ and $c$ need to count positions on the output tape, and hence will use $\log_2 l(n)$ bits each.

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