Exercise 1: Let $A$ be an algorithm of space complexity $s(m)$. Show that there is an algorithm $A'$ such that

- $L(A) = L(A')$
- $A'$ has space complexity $s'(m) = O(s(m) + \log n)$
- $A'$ 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 $A_x$ such that

- $L(A_x) = L(A)$
- $A_x$ does not scan the input tape beyond location $2O(s(m) + \log_2 n)$ 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 $t(m) \leq C q(m)$ with $q(m) = O(s(m))$ (where $s(m)$ is a polynomial space bound).

We showed $q(m) = 2^m s(m) + d$, where $C = |\Gamma| + |\{A\}|$

In our case:

- $q(m) = 2^m s(m) + \log_2 n = O(s(m))$

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

$\Rightarrow m - 2O(s(m)) = 2 \log_2 n = 2O(s(m) + \log_2 n)$

different configurations at most.

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

The resulting algorithm $A_i'$ works as follows:
- whenever $A_i$ would move right past the end of the input, $A_i'$ instead:
  - does not move past the end of the input, but maintains a counter on the work tape
  - whenever $A_i$ moves right, the counter is incremented left, and decremented

In this way, $A_i'$ can keep track of the position of the input head of $A_i$.

Whenver $A_i$ moves back again over the input symbol, $A_i'$ does not update the counter (leaving it to 0)

$A_i'$ operates similarly whenever $A_i$ moves left past the beginning of the input.

How much space does the counter use?
Since $A_i$ does not scan the input tape beyond $R_1(n) = 2^{O(n + \log n)}$, the counter takes
\[ \log_2 R_i(n) = O(n + \log n) \]

Hence, the total space used by $A_i'$ is
\[ O(n) + O(n + \log 2) = O(n + \log n) \]
Exercise: Let \( A \) be an algorithm of space complexity \( \mathcal{O}(m) \).
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 \( \mathcal{O}(m) = \mathcal{O}(\log l(m)) \)
  where \( l(m) = \max \{ A(w) \mid w \in \{0,1\}^m \} \) is the size of the maximum output for input \( x \) of length \( m \)
- \( A' \) never overwrites 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 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 \( k \)
- \( i \), \( j \), and \( k \) here to count positions on the output tape, and hence will use \( \log_2 l(n) \) bits each.

We get that \( S'(n) = S(n) + O(\log_2 l(n)) \).
Exercise 11.11 b)

Consider the problem **FALSE-SAT**:

Given a boolean expression $E$ that is false when all its variables are made false, is there some other truth assignment that makes $E$ false besides all-false?

Decide whether the problem is in NP or coNP.

Describe its complement.

If the problem or its complement is NP-complete, prove it.

**Proof:**

The problem is NP-complete.

- In NP: given a boolean expression $E$, we need to check:
  1) that $E$ is false when all variables are assigned false
  2) that there is some other truth assignment making $E$ false

  (1) can be done in poly-time by a DTM
  (2) can be done in poly-time by a NTM
    - guess a truth assignment $T$ different from all false, and
    - answer yes if under $T$, $E$ evaluates to false

- NP-hard: by a reduction from SAT

  Let $E$ be a boolean expression with variables $x_0, \ldots, x_n$.
  We construct an expression $E'$ s.t. $E \in \text{SAT}$ iff $E' \in \text{FALSE-SAT}$

  1) Rest if $E$ is true when all variables are false (polynomial)
  2) if so, $E \in \text{SAT}$, and we convert it to a fixed expression
     that is in FALSE-SAT, e.g., $\land y$. 

2) Otherwise, let \( E' = \neg E \land (x_1 \lor x_2 \lor ... \lor x_n) \).

Clearly, the reduction is poly-time.

We have that \( E' \) is false when all of \( x_1, ..., x_n \) are false.

Notice that in case (2), \( E' \) is false when all variables are false.

Hence, if \( E \in SAT \), then it is satisfied by a truth-assignment \( T \) different from all-false.

Thus, \( \neg E \) is made false by \( T \), and \( E' \in FALSE-SAT \).

Conversely, if \( E' \in FALSE-SAT \), then since \( x_1, ..., x_n \) is false only for the all-false truth assignment, there must be some other truth-assignment \( T \) that makes \( \neg E \) false. Then \( T \) makes \( E \) true, and \( E \in SAT \).