Lecture 11

Gidon Rosalki

2025-06-15

1 Sub linear space complexity

In order to discuss sub linear space classes, we defined TMs with 3 tapes:

1. Input: read only
2. Work tape: read / write, we use only this tape to measure space
3. Output: Write only, at every step may write a letter, or not. If we write, then we move the head one step to the right, and never return left.

We defined the classes

$$L = Space\left(\log\left(n\right)\right)$$
$$NL = NSpace\left(\log\left(n\right)\right)$$

Theorem 1 (Savitch)**.**

$$NSpace\left(f\left(n\right)\right) \subseteq Space\left(f^2\left(n\right)\right)$$

For every $f(n)$ that is computable in $O(f(n))$.

Proof. We showed this for functions $f(n) \geq n$, and the theorem is true also for functions $f(n) < n$ □

Let there be $f(x)$, a computable function in space $O(\log(n))$. That is to say, there is a TM M_f, with 3 tapes (as defined above), such that for every input x on the input tape. M_f finished in state q_{acc}, with $f(x)$ on the output tape, while using space $O(\log(n))$ on the work tape, where $n = |x|$. Let $g(n)$ be a computable function, in space $O(\log(n))$, by the TM M_g.

Theorem 2. *$f(g(x))$ is computable in space $O(\log(n))$.*

Proof . We showed that if f and g are computable, then $f \circ g$ is computable, and we showed that if f and g are computable in polynomial time, then so too $f \circ g$ are computable in polynomial time, by creating a new TM M, which for an input x, runs M_g on x, so $w = M_g(x)$, and saves w on the tape, and then runs M_f on w, so thus we receive $f(g(x))$, which is clearly computable, since each individual function is computable, and clearly occurred in polynomial time, since both individual computations occurred in computational time.

How can we create 2 machines, both with 3 tapes, by using a machine with 3 tapes, that computes $f(g(x))$?
Solution 1: Correct, but wasteful of space: We will change M_g such that instead of printing to the output, it will print to the work tape, and we will change M_f, such that instead of reading from the input tape, it will read from the work tape. The new machine will run as follows: Run the new M_g on x, and then run the new M_f on what is written on the work tape. Correctness is trivial, we simply need to understand the space complexity. Let us call the combined machine M. We showed last week that the output of M is indeed $f(g(x))$, but the problem is that M may use space that is not bounded by $O(\log(n))$. This is because we showed last week that a machine with a limit of linear space, will have the runtime, and the output space bounded by a polynomial. This means that $|w| = |g(x)|$ may be polynomial, and so the machine M may use polynomial space in n, instead of logarithmic in n.

We will show a solution that allows the function assembly, such that in total the used space will be $O(\log(n))$. .The concept is that we will compute the letters of x dynamically, every time from the beginning. That is to say, we will run M_f, and M_g simultaneously. We will hold 2 counters, one that changes the print head of M_g, and one that changes the print head of M_f . The machine M will compute $f(g(x))$ as follows: It will run M_f. With every run step, in order to know what is the current letter of $g(x)$, M will run M_g the required number of prints, which is to say, for every step left of the read head of f, we will simulate this by reducing the relevant counter by 1, and every step right will increase it by one. Every instruction print in M_g will be simulated by increasing the counter of the output of M_g. Every step of M_f, we will run $M_g(x)$ from the beginning (zeroing the print counter). We will run M_g until the print counter is equal to the counter for the input tape of M_f. Then, M will know the input of the letter from what $M)g$ prints, and M_f reads.

The correctness is trivial.
The runtime is very large. We want to convince ourselves that the space complexity is $O(\log(n))$. What does M need to hold on the work tape?

1. Space of the work tape of M_f
2. Space of the work tape of M_g
3. Counter for the head of M_g
4. Counter for the head of M_f

1. M_f is contained in space $O(\log(|g(x)|))$, ie log the length of output of $g(x)$. Reminder, a DTM that is contained in logarithmic space, has output contained by a polynomial, which is to say, length $\leq n^c$, where c is some constant. That is to say, M_f is contained in space $O(\log(n^c))$. That is to say that M_f is contained in space $O(\log(n))$.
2. M_g is contained in $O(\log(|x|))$, in order to compute $g(x)$, which is to say $O(\log(n))$
3. We need counters that can count until $|g(x)|$, the length of the output of M_g, which is to say the length of the input of M_f. We have already shown that $|g(x)|$ is constrained by n^c, and therefore for a counter, we need $O(\log(n^c))$ cells, which is $O(\log(n))$.
4. As above, and so below

In total, we have $O(\log(n))$ cells, which is to say that M is a machine that computes $f(g(x))$ in $O(\log(n))$ space. Therefore, the assembly of two logarithmic space functions, is itself a logarithmic space computation.

Note: We have shown that this is true for logarithmic space. It is also possible to show for other sub linear functions (but not all of them). □

2 Classes L and NL

We will define a linear space reduction (we will stop reminding that this refers to space, and just say reduction for this lecture). For the languages A, B, we will say that there is a linear reduction from A to B, by writing

$$A \leq_L B$$

if and only if there exists a reduction f from A to B such that f is computable in linear space.
Note: If $A \leq_L B$ then $A \leq_p B$. (The machine that computes the reduction in logarithmic space will also compute in linear time).
The theorem of $L = NL$ is an open question, as is the theorem $L = P$. Additionally, so is the theorem $NL = P$.
We have seen that $L \subseteq P$, and we will show in the tutorial that $NL \subseteq P$.

Theorem 3 (Closure of reduction assembly)**.**

$$A \leq_L B \wedge B \leq_L C \implies A \leq_L C$$

Proof . The proof is similar to the proof in the polynomial case. We use the closure of computable functions in logarithmic space. That is to say, if f is computable in logarithmic space, and is the reduction from A to B, and g is computable in logarithmic space, and is the reduction from B to C, then $g \circ f$ is computable in logarithmic space, and us the reduction from A to C. □

Definition 2.1. *We will say that the language B is NL-hard, if for every $A \in NL$, it is true that $A \leq_L B$.*

Theorem 4. *If B is a language that is NL-hard, and also $B \leq_L C$, then $C \in$NL-hard*

Proof . Since $B \in$ NL-hard, then it is true that for every $A \in NL$ that $A \leq_L B$. Additionally, $B \leq_L C$ (given). Closure of logarithmic space reductions is given, so the assembly gives $A \leq_L C$. That is to say, $C \in$NL-hard □

Definition 2.2. *We will say that $C \in$ NL-Complete **if and only if** $C \in$NL-hard, and also $C \in NL$.*

In order to show that a language is complete in NL, will will do it similarly to Cook-Levin theorem for showing completeness in NP.

Let us define

$$\text{PATH} = S - T - CONN = \{(G, s, t) : G\text{Is a directed graph } s, t \in V(G) \text{ and there is a directed path from } s \text{ to } t \text{ in } G\}$$

Theorem 5.

PATH ∈ NL-Complete

Proof . We will show that

1. PATH ∈ NL
2. PATH ∈ NL-hard

1. We will create an NTM, that is contained in space $O(\log(n))$, for the language PATH. The machine M will run as follows: Hold a variable for the current node, initialised to s. For every step that M will take, in a non deterministic manner, it will go over the current nodes neighbours, and check if it has reached t. If so, then M will stop, and return q_{acc}. Otherwise, it will continue to the next neighbour.
Correctness: If $(G, s, t) \in$PATH, then there is a directed path from s to t, in G, and therefore there is a computation where M finds the path from s to t, and so returns q_{acc}.
If $(G, s, t) \notin$ PATH, then in every run of M the path will never reach t, and therefore M will not return q_{acc}.
Space: M uses the space on the work tape for the current node $O(\log(n))$, and space for the next node. Another counter is used to find the edges in the representation of G on the input tape in space $O(\log(n))$. In total, we have used $O(\log(n))$ space.

   Note: We can change M such that it will always stop, for example, add a counter for the length of the path, which we increase for every step we take along the path, and if we reach the number of nodes in G, we stop and return q_{rej}.

   Note: The move from a node, to one of its neighbours, is through the input G. The way that this is done is dependent on the representation of G (edges list, neighbours matrix, etc.). I any case, we may do this with one or two counters, which will take the additional space $O(\log(n))$.

2. We will now show that PATH ∈NL-hard. That is to say, for every $A \in NL$, it is true that $A \leq_L$PATH. Or in other words, there is a function f, computable in logarithmic space, such that for every w, it is true that

$$w \in A \Leftrightarrow f(w) = (G, s, t) \in \text{PATH}$$

   The only additional given is that there exists a TM M, non deterministic, that is computable in logarithmic space for A.

   Reminder: The configurations graph $G_{M,w}$ (discussed last week) holds: There is a path from an initial configuration c_0, to the final configuration c_{acc} in the graph $G_{M,w}$ **if and only if** there is an accepting run of $M(w)$, **if and only if** $w \in A$. (W.l.o.g there is a single accepting configuration).
   Therefore, For the input w, the reduction will return $(G_{M,w}, c_0, c_{\text{acc}}) = (G, s, t)$.
   Correctness: Trivial.
   Space complexity: We need to show that we may compute this reduction in logarithmic space. We want to show that given w, to create $(G_{M,w}, c_0, c_{\text{acc}})$ by a DTM, in logarithmic space.
   c_0 is the starting configuration, which is to say the work tape is empty, the heads are at the beginning, and the internal state is q_0.
   c_{acc} is hte accepting configuration, meaning that the for both variables. working tape is empty, the head is at the start, and the internal state is q_{acc}.
   It is easy to create c_0, c_{acc}. It simply remains to clarify how we print $G_{M,w}$ in logarithmic space. We will choose the edges representation, which is to say, pairs of nodes. The reduction machine M_f will create $G_{M,w}$ as follows:
   M_f will hold a pair of variables on the working tape. Each one will representation a configuration. The required space to represent a configuration is

$$|Q| \cdot n \cdot O(\log(n)) \cdot |\Gamma|^{O(\log(n))}$$

   Which is the state, times the location of the input head, times the location of the working head, times the contents of the working tape.
   We may store one configuration in $O(\log(n))$. For every pair of representations of the two variables, the machine will check if they are twinned to the pair of the following configuration. If so, it will copy / paste to the output tape. In every case, and if not, it will increment the variables, and the counters. That is to say, M will pass over all the possible values, for every pair of variables. To check if a pair of values is suitable to a pair of following configurations, which is to say if 2 configurations are almost equal, for example, changes that twin to the δ function of M, are able to be done in the space limitations.
   Note: In order to save a configuration, we need the space for the size of the work tape, $O(\log(n))$, and the location of the head of the work tape $O(\log(\log(n)))$, the head for the input tape $O(\log(n))$, and internal state $O(\log(Q))$. In total, $O(\log(n))$.

□

We have / will show

$$L \subseteq NL \subseteq P \subseteq NP$$

We will show the middle containment in the tutorial. We are left with the following questions:

- $L = NL$
- $L = P$
- $NL = P$
- $P = NP$

These are still open questions, to which the world knows not the answers.