CS 4510: Automata and Complexity **Fall 2019**

Lecture 15: Time and Space

October 28, 2019

Lecturer: Santosh Vempala *Scribe: Aditi Laddha*

15.1 Introduction

Church-Turing thesis: Any computable function is computable by a Turing machine.
Note that it is a hypothesis.
Question: What are the resources available for computation?

- Space
- Time
- Random Bits
- Non-determinism

15.1.1 Space and Time

Definition 15.1 (NTIME) *A language L is said to belong to class* NTIME$(t(n))$ *if there exists a NTM M that decides L and for input strings of size n, L runs for $O(t(n))$ steps.*

Definition 15.2 (DTIME) *A language L is said to belong to class* DTIME$(t(n))$ *if there exists a TM M that decides L and for input strings of size n, L runs for $O(t(n))$ steps.*

Definition 15.3 (NSPACE) *A language L is said to belong to class* NSPACE$(s(n))$ *if there exists a NTM M that decides L and for input strings of size n, L uses $O(s(n))$ additional space.*

Definition 15.4 (DSPACE) *A language L is said to belong to class* DSPACE$(s(n))$ *if there exists a TM M that decides L and for input strings of size n, L uses $O(s(n))$ additional space.*

Note that we do not count input cells in the space used by a TM.

Definition 15.5 (Space-constructible) *A function $s : \mathbb{N} \to \mathbb{N}$ is called space-constructible if $s(n) \geq \log_2(n)$ and there exists a Turing machine that on input 1^n outputs $s(n)$ in binary while using $O(s(n))$ space.*

Definition 15.6 (Configuration of a TM) *The configuration of a Turing machine M can be completely specified by the 3-tuple*

$$\langle \text{Tape Content, State, Head Position} \rangle$$

If we know that a TM M never uses more than $s(n)$ on input of size n, then the number of possible configuration of M while computing on x is at most $|\Gamma|^{s(n))} \cdot |Q| \cdot (s(n) + n)$ as every cell on the tape can have one of Γ tape symbols and the head can be positioned on the input or any of the $s(n)$ cells used by M.

15.2 Space complexity and Savitch's Theorem

Theorem 15.7 NTIME$(t(n)) \subseteq$ DTIME$(2^{O(t(n))})$.

Proof: Consider a language $L \in$ NTIME$(t(n))$ and let N be a NTM that decides L in $O(s(n))$ space. On input x with $|x| = n$, let T_x be the computation tree produced when N computes on x. If $x \in L$, then at depth $t(n)$ at least one of the nodes is a leaf with state q_{accept} and if $x \notin L$, then at depth $t(n)$, all the nodes must be leaves with states q_{reject}. The branching factor of this tree is $b = |Q| \cdot |\Sigma| \cdot 2$, a constant. Hence, T has at most $b^{t(n)}$ nodes. Consider a TM M which performs BFS on this tree starting from q_0 and accepts if it reaches a configuration with state q_{accept} and rejects otherwise. M decides L is $b^{t(n)} = 2^{O(t(n))}$ time. ■
Note that performing DFS would not work because even if N accepts x, there might be a non-deterministic computation path of unbounded depth.

Lemma 15.8 DTIME$(t(n)) \subseteq$ DSPACE$(t(n))$.

Because a TM cannot use more space than its run-time as at each step the head moves only one step to the left or the right.

Theorem 15.9 NSPACE$(s(n)) \subseteq$ DTIME$(2^{O(s(n))}) \subseteq$ DSPACE$(2^{O(s(n))})$.

Proof: Consider $L \in$ NSPACE$s(n)$ and let M be a NTM that decides L in $O(s(n))$ space. Total number of configurations of with $s(n)$ space is

$$\begin{aligned}
\#\text{ configurations} &= |\Gamma|^{s(n))} \cdot |Q| \cdot (s(n) + n) \\
&= 2^{s(n)\log_2(|\Gamma|)+\log_2(|Q|)+\log_2(s(n)+n)} \\
&= 2^{O(s(n)+\log_2(s(n)+n))} \\
&= 2^{O(s(n))}
\end{aligned}$$

We will construct a deterministic TM D that decides L in $O(2^{O(s(n))})$ space. For an input x of length n, consider a directed graph $G_x = (V, E)$ where V = set of configurations of M on input x and there is an edge between configurations $C_i, C_j \in V$ if M can go from configuration C_i to configuration C_j in one step. Then x is accepted by M if and only if there exists a path in G_x from C_0 to C_{appect}. We can check the existence of such a path in $O(|V|)$ time by running BFS. ■

Theorem 15.10 (Savitch's theorem) *For a function $f : \mathbb{N} \to \mathbb{R}^+$, where $f(n) \geq n$,*

$$\text{NSPACE}(s(n)) \subseteq \text{DSPACE}((s(n))^2)$$

Consider $L \in$ NSPACE$s(n)$ and let M be a NTM that decides L in $O(s(n))$ space. We will construct a deterministic TM D that decides L in $O((s(n))^2)$ space. For an input x of length n, consider a directed graph $G_x = (V, E)$ where V = set of configurations of M on input x and there is an edge between configurations $C_i, C_j \in V$ if M can go from configuration C_i to configuration C_j in one step. Without loss of generality, we can assume that the TM M on reaching accept state write 0 on each cell of the tape and moves to head to the leftmost cell. Let's call this configuration C_{accept}. So every time M is in state q_{accept}, it goes to configuration C_{accept}. Then x is accepted by M if and only if there exists a path in G_x from C_0 to C_{appect}.

Given a directed graph $G = (V, E)$ with n vertices and $s, t \in V$, algorithm 1 can decide whether there exists a path from s to t of length k in $O(\log_2(n)\log_2(k))$ space.
So, in G_x, $|V| = 2^{O(s(n))}$. Let $N = |V|$. The maximum length of path between C_0 and C_{accpet} is N. So, the space needed to decide whether there exists a path from C_0 to C_{accpet} is

$$\begin{aligned}
&= \log_2(N) \cdot \log_2(k) \\
&= (\log_2(N))^2 \\
&= (O(s(n)))^2
\end{aligned}$$

15.2.1 Reachability

Given a directed graph $G = (V, E)$ with n vertices and $s, t \in V$, the following algorithm decides whether there exists a path from s to t of length at most k in G.

Algorithm 1: PATH(u, v, k)

if $u = v$ OR $\{(u, v) \in E \land k \geq 1\}$ **then**
    return YES
else
    for $w \in V \backslash \{u, v\}$ **do**
        if PATH$(u, w, \lfloor \frac{k}{2} \rfloor)$ AND PATH$(u, w, \lceil \frac{k}{2} \rceil)$ **then**
            return YES
return NO

Let $|V| = n$, then we can store the current intermediate vertex w with $\log_2(n)$ bit. So, the space needed by the algorithm is

$$\begin{aligned} &= \log_2(n) \cdot \text{depth of recursion} \\ &= \log_2(n) \cdot \log_2(k) \end{aligned}$$

15.3 References

- Ch 8.1 Savitch's Theorem, "Introduction to the Theory of Computation"