CS340: Theory of Computation Sem I 2017-18

Lecture Notes 20: Space Complexity

Raghunath Tewari IIT Kanpur

Recall the following definitions.

$$\begin{aligned} \mathsf{SPACE}(f(n)) &= \{L \mid L = L(M) \text{ for some det. TM } M \text{ that uses } O(f(n)) \text{ amount of space}\} \\ \mathsf{NSPACE}(f(n)) &= \{L \mid L = L(N) \text{ for some NDTM } N \text{ that uses } O(f(n)) \text{ amount of space}\} \end{aligned}$$

Note:

- The space used by a TM is the number of cells used in the work tape of the TM.
- The space used by a NDTM is the maximum space used over all computation paths of the TM.

1 Basic Relations Between the Time and Space Classes

Claim 1. *(a)* $\mathsf{DTIME}(f(n)) \subseteq \mathsf{SPACE}(f(n))$ *and*

(b) $\mathsf{NTIME}(f(n)) \subseteq \mathsf{NSPACE}(f(n))$.

If a TM runs for t steps on a input then it cannot visit more than t cells.

Claim 2. $\mathsf{SPACE}(f(n)) \subseteq \mathsf{NSPACE}(f(n)) \subseteq \mathsf{DTIME}(2^{O(f(n))})$

The first containment follows from the definition of nondeterminism.
Every cell of a TM (deterministic or non deterministic) can contain c different symbols, where c is a constant that depends on the description of the TM. Now if the TM uses $kf(n)$ cells on an input of length n then the total number of possible configurations of the TM is $c^{kf(n)}$. Now design a deterministic TM that does a DFS/BFS on the configuration graph and accepts if and only if there is a path from the start configuration to an accept configuration. Since the size of the graph is at most $c^{kf(n)}$ therefore the running time is $c^{O(f(n))}$ which is also $2^{O(f(n))}$.

Note 1. For every TM (deterministic or non deterministic) there is a constant K, such that if the TM uses s cells to compute its answer then it must produce the answer within K^s number of steps. This is because, if the TM does not produce an answer within K^s number of steps then there must be some configuration which is visited twice during the computation and hence the machine gets into an infinite loop.
One can get rid of such a scenario by maintaining a counter in the TM that counts upto K^s, and if the machine has not accepted or rejected its input by that time, then it rejects the input.

Claim 3. $\mathsf{NTIME}(f(n)) \subseteq \mathsf{SPACE}(f(n))$

Consider the configuration graph of a NDTM N on an input x, that takes t steps to produce an answer. The maximum length of any path in $G_{N,x}$ from the root vertex (start config.) to a leaf (halting config.) is at most t. Note that since the running time of N is t therefore $G_{N,x}$ has $2^{O(t)}$ vertices in it. Hence a standard traversal algorithm will not work.

We perform a "non-marking DFS" on $G_{N,x}$ to check for reachability. Starting from the root, we implement DFS using a stack that stores the path from the root to the current vertex in the graph. Whenever all children of a vertex v gets visited, we go back to the parent of v and move to the lexicographically next sibling of v. Whenever we visit a new vertex we push it onto the stack and whenever we are done visiting all children of a vertex, we pop it out of the stack.
Note that the above algorithm does not mark a vertex once it is visited. Hence a vertex can get visited several times and the running time of the algorithm can be as large as $2^{O(t)}$, although the number of vertices can be much smaller than that. The only catch in the above algorithm is if the configuration graph contains cycles. Since we are not marking the vertices we can run into an infinite loop. However the graph can be made acyclic by a little bit of preprocessing, as mentioned in Note 1.

1.1 A Simple Example

Claim 4. $\mathsf{SAT} \in \mathsf{SPACE}(n)$.

Input: A Boolean formula ϕ having n variables.

1. Iterate over all truth assignments $\tau \in \{0,1\}^n$.
2. Compute $\phi(\tau)$, by substituting the values in each variable of ϕ.
3. If $\phi(\tau) = 1$ for some τ then *accept*.
4. If for all τ, $\phi(\tau) = 0$ then *reject*.

The algorithm needs space to store a truth assignment and to compute $\phi(\tau)$. Both can be implemented in linear space in the size of the formula.

1.2 Some Standard Classes and their Relations

- $\mathsf{L} = \mathsf{SPACE}(\log n)$
- $\mathsf{NL} = \mathsf{NSPACE}(\log n)$
- $\mathsf{PSPACE} = \bigcup_{k\geq 0} \mathsf{SPACE}(n^k)$
- $\mathsf{NPSPACE} = \bigcup_{k\geq 0} \mathsf{NSPACE}(n^k)$
- $\mathsf{EXP} = \bigcup_{k\geq 0} \mathsf{TIME}(2^{n^k})$
- $\mathsf{NEXP} = \bigcup_{k\geq 0} \mathsf{NTIME}(2^{n^k})$

Known Relations

$$\mathsf{L} \xrightarrow{\text{defn}} \mathsf{NL} \xrightarrow{\text{Claim 2}} \mathsf{P} \xrightarrow{\text{defn}} \mathsf{NP} \xrightarrow{\text{Claim 3}} \mathsf{PSPACE} \xrightarrow{\text{defn}} \mathsf{NPSPACE} \xrightarrow{\text{Claim 2}} \mathsf{EXP} \xrightarrow{\text{defn}} \mathsf{NEXP}$$

2 Savitch's Theorem

Theorem 5 (Savitch's Theorem)**.** *For $f(n) \geq \log n$,* $\mathsf{NSPACE}(f(n)) \subseteq \mathsf{SPACE}(f(n)^2)$.

Proof. Let $L \in \mathsf{NSPACE}(f(n))$ accepted by a NDTM say N. Given an input x, consider the configuration graph of N on x, $G_{N,x}$. The number of vertices in the graph is $2^{O(f(n))}$. To show that L is in $\mathsf{SPACE}(f(n)^2)$, it is enough to give a deterministic algorithm that solves the reachability problem in a graph with n vertices, using $O(\log^2 n)$ space.
We give a recursive algorithm to solve reachability, that is based on the following idea.
There is a path from s to t in G of length at most n if and only if there exists a vertex $u \in G$ such that there is a path from s to u in G of length at most $n/2$ and there is a path from v to t in G of length at most $n/2$.
Note that in a graph with n vertices the maximum length of a path between any two vertices (if it exists) is n. We define a recursive algorithm $\mathsf{RecReach}$ as follows:

$$\begin{aligned} \mathsf{RecReach}(G,s,t,i) &= 1 \quad \text{if there is a path from } s \text{ to } t \text{ in } G \text{ of length at most } 2^i \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Description of $\mathsf{RecReach}(G,s,t,i)$

(i) If $i = 0$ then output 1 if and only if $s = t$ or if there is an edge from s to t.

(ii) Output 1 if there exists a vertex $u \in G$ such that

- $\mathsf{RecReach}(G,s,u,i-1) = 1$ and $\mathsf{RecReach}(G,u,t,i-1) = 1$.

(iii) If there is no vertex that satisfies the above condition then output 0.

Let $S(n,i)$ be the space used by $\mathsf{RecReach}(G,s,t,i)$ where G has n vertices and the distance between s and t is at most 2^i. Then,

$$\begin{aligned} S(n,\log n) &= S(n,\log n - 1) + c_1 \log n \\ S(n,0) &= c_2. \end{aligned}$$

Solving the recurrence we have $S(n,\log n) = O(\log^2 n)$. □

Corollary 6. *1.* $\mathsf{NL} \subseteq \mathsf{SPACE}(\log^2 n)$.

2. $\mathsf{PSPACE} = \mathsf{NPSPACE}$.

Note 2. Space can be reused but time cannot be.