Practice problems: Time and Space complexity

1. Show that there exists a function that is not time-constructible.
2. Show that $NTIME(f(n)) \subseteq DSPACE(f(n))$.
3. Show that $P \neq DSPACE(n)$
4. PSPACE is closed under union, intersection and complement.
5. NL is closed under union, intersection and complement.
6. Show that if a language is NL-complete then it is also coNL-complete.
7. Show that the language consisting of strings with properly nested parentheses is in L.
8. A ladder is a sequence of strings $s_1, s_2, \ldots, s_k$ wherein every string differs from the preceding one in exactly one character. Fr example, the following is a ladder of English words, starting with "head" and ending with "free": head, hear, near, fear, bear, beer, deer, deed, feed, feet, fret, free. Let

   $LADDER_{DFA}$ Takes as input $< M, s, t >$ where M is a DFA and determines if $L(M)$ consists of a ladder of strings starting with s and ending with t, where $s, t \in \Sigma^*$ and M is over input alphabet Σ.
   Show that the language is in PSPACE. (Hint: PSPACE = NPSPACE).

Solutions in the next page. Please try the problems first.

1. We use diagonalisation to construct such a function. Let $M_0, M_1, M_2, \ldots$ be an enumeration of TMs. We construct a function $f : N \to N$ such that there is no machine that can compute the function in $f(n)$ time. Let f be computed as follows. On input n, simulate $M_n(n)$. If $M_n(n) \neq n$, set $f(n) = n$; otherwise set $f(n) = n^2$. Output $f(n)$.
   Let n_0 be the encoding of a TM computing f . If $f(n_0) = n_0$, then $M_{n_0}(n_0) \neq n_0$ i.e., $M_{n_0}(n_0) \neq f(n_0)$. If $f(n_0) = n_0^2$, then $M_{n_0}(n_0) = n_0$ i.e., $M_{n_0}(n_0) \neq f(n_0)$. This contradicts the choice of M_{n_0} as the TM computing f. Thus f is not constructible.

2. Let $A \in NTIME(f(n))$. Then there is an NDTM M running in time $f(n)$ deciding A. We describe a space $f(n)$ DTM N deciding A. That is, given a string x, N decides whether or not $x \in A$. Consider the configuration graph $G_{M,x}$ on input x. Each computation path is of length at most $f(n)$. Also M makes at most $f(n)$ choices, which can be encoded using an $f(n)$-bit string (we can assume there are two transition choices at each configuration - think about how to design an equivalent TM where this property holds. Since the transition function has finitely many transitions there are finitely many transitions that can be applied at any step). N checks for each $f(n)$-bit string whether following these choices M enters an accepting configuration or not. If for some sequence of choices, M accepts then N accepts and halts. If after trying all $f(n)$-bit strings representing choices, M rejects, then N rejects. At any point of time, N stores an $f(n)$-bit string representing choices. And to check whether a path ends in accepting configuration, N writes down the configurations resulting from the choices represented in the $f(n)$-bit string, one at a time, erasing the previous one in each iteration. Each configuration can be encoded using $O(f(n)$-bits. So the total space requirements of N is $O(f(n))$ implying that $A \in DSPACE(f(n))$.

3. (Proof by contradiction) Assume that $P = DSPACE(n)$. Let $L \in DSPACE(n^2)$ and let M be a space n^2 deterministic TM deciding L. Construct a language $L_0 = \{x01^{|x|-|x|-1} | x \in L\}$ (padding trick). The map from strings in L to strings in L_0 can be computed in n^2 time i.e., $L \leq_p L_0$. By our assumption, there exists a space n deterministic TM that decides L_0 which ignores (does not have enough time to read) the last $n^2 - n$ bits of the input (n^2 here is the length of the input) and then simulates M. Since M runs in space n^2 which is linear in length n^2 of the input, $L_0 \in DSPACE(n)$. By our assumption that $DSPACE(n) = P$, we have $L_0 \in P$. But since $L \leq_p L_0$, we have $L \in P$. This holds for any $L \in DSPACE(n^2)$, thus implying that $DSPACE(n^2) \subseteq P = DSPACE(n)$, contradicting the space hierarchy theorem.
   Note: Similarly, one can show that $DSPACE(n) \neq NP$.

4. Let $L_1, L_2 \in PSPACE$ and let M_1, M_2 be deterministic TMs deciding L_1, L_2 respectively in polynomial space. We describe constructions of de-

terministic TMs running in polynomial space for the following languages.
$L_1 \cup L_2$: On input x, run M_1 on x. If it accepts, then accept. Otherwise, run M_2 on x. If it accepts, then accept; else reject.
$L_1 \cap L_2$: On input x, run M_1 on x and then run M_2 on x. If both accept, then accept; otherwise reject.
L_1^c : On input x, run M_1 on x and negate its output. (This can be done since M is deterministic).

5. Same as previous problem. The complement result comes from the Immerman-Szelepcsenyi Theorem.

6. Consider language Π that is NL-complete. This means that $\Pi \in NL$ for any $\Pi' \in NL$, $\Pi' \leq_l \Pi$. By Immerman-Szelepcsenyi Theorem, $\Pi \in coNL$. Similarly, every language in NL is also in $coNL$. Thus every language in $coNL$ logspace reduces to Π. This implies that Π is coNL-complete.

7. Initialise a counter to zero. While reading the input string from left to right, increment the counter whenever a left paranthesis is encountered. Decrement the counter whenever a right paranthesis is read. Reject the string if the counter becomes negative or does not become zero after reading the end of the input string. Otherwise accept. It requires log-space to store the counter.

8. Since $NPSPACE = PSPACE$, it suffices to show that $LADDER_{DFA} \in NPSPACE$. We describe a non-deterministic algorithm that decides $LADDER_{DFA}$. Given an instance $< M, s, t >$, reject if $|s| \neq |t|$. Otherwise, consider a graph G whose vertices are labeled with strings in $\Sigma^{|s|}$. There is a directed edge from vertex u to vertex v if u and v differ in exactly one character and $u, v \in L(M)$. Then, $< M, s, t > \in LADDER_{DFA}$ iff there is a path from s to t in G. This can be checked non-deterministically in polynomial space (although G has exponential number of vertices) – guess the path, storing at each step only the label of the current vertex (which is of size $|s|$). At a step, suppose the current vertex is u. Non-deterministically choose a new out-neighbour v of u and check if M accepts v. To ensure that the machine always halts, maintain a counter which is incremented after each guess. Reject and halt when the counter crosses $|\Sigma|^{|s|}$. This is because any path from s to t (without loops) can be of length at most $|\Sigma|^{|s|}$.