Homework 6

Antonio Zea Jr

December 7, 2022

Problems

7.1 Answer each part TRUE or FALSE.

1. $2n = \mathcal{O}(n)$ is TRUE
   $2n \leq c(n)$, where $c = 3$ and $n_0 = 1$ then
   $2n \leq 3(n)$, $n \geq 1$

2. $n^2 = \mathcal{O}(n)$ is FALSE
   $n^2 \geq c(n)$, $\forall c \in \mathbb{N}$

3. $n^2 = \mathcal{O}(n \log^2 n)$ is FALSE
   $n^2 \geq c(n \log^2 n)$, $\forall c \in \mathbb{N}$

4. $n \log n = \mathcal{O}(n^2)$ is TRUE
   $n \log n \leq c(n^2)$, where $c = 1$ and $n_0 = 2$ then
   $n \log n \leq 1(n^2)$, $n \geq 2$

5. $3^n = 2^{\mathcal{O}(n)}$ is FALSE
   $3^n \geq 2^{c(n)}$, $\forall c \in \mathbb{N}$

6. $2^{2^n} = \mathcal{O}(2^{2^n})$ is TRUE
   $2^{2^n} \leq c(2^{2^n})$, where $c = 1$ and $n_0 = 1$ then
   $2^{2^n} \leq 1(2^{2^n})$, $n \geq 1$

7.2 Answer each part TRUE or FALSE.

1. $n = o(2n)$ is TRUE
   $n < c(2n)$, where $c = 1$ and $n_0 = 1$ then
   $n < 1(2n)$, $n \geq 1$

2. $2n = o(n^2)$ is TRUE
   $2n < c(n^2)$, where $c = 1$ and $n_0 = 3$ then
   $2n < 1(n^2)$, $n \geq 3$

3. $2^n = o(3^n)$ is TRUE
   $2^n < c(3^n)$, where $c = 1$ and $n_0 = 1$ then
   $2^n < 1(3^n)$, $n \geq 1$

4. $1 = o(n)$ is TRUE
   $1 < c(n)$, where $c = 1$ and $n_0 = 2$ then
   $1 < 1(n)$, $n \geq 2$

5. $n = o(\log n)$ is FALSE
   $n \geq c \log n, \forall c \in \mathbb{N}$

6. $1 = o(\frac{1}{n})$ is FALSE
   $1 \geq c\frac{1}{n}, \forall c \in \mathbb{N}$

7.3 Which of the following pairs of numbers are relatively prime? Show the calculations that led to your conclusions.

1. $$\begin{aligned} 10505 &= 1274 \cdot 8 + 313 \\ 1274 &= 313 \cdot 4 + 22 \\ 313 &= 22 \cdot 14 + 5 \\ 22 &= 5 \cdot 4 + 2 \\ 5 &= 2 \cdot 2 + \mathbf{1} \\ 2 &= 1 \cdot 2 + 0 \end{aligned} \quad \therefore \gcd(1274, 10505) = 1$$

2. $$\begin{aligned} 8029 &= 7289 \cdot 1 + 740 \\ 7289 &= 740 \cdot 9 + 629 \\ 740 &= 629 \cdot 1 + 111 \\ 629 &= 111 \cdot 5 + 74 \\ 111 &= 74 \cdot 1 + \mathbf{37} \\ 74 &= 37 \cdot 2 + 0 \end{aligned} \quad \therefore \gcd(7289, 8029) = 37$$

7.5 Is the following formula satisfiable?
$(x \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y})$

x	y	$(x \vee y)$	$(x \vee \overline{y})$	$(\overline{x} \vee y)$	$(\overline{x} \vee \overline{y})$	$(x \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y})$
0	0	0	1	1	1	0
1	0	1	1	0	1	0
0	1	1	0	1	1	0
1	1	1	1	1	0	0

$\therefore (x \vee y) \wedge (x \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (\overline{x} \vee \overline{y})$ is not satisfiable.

7.6 Show that P is closed under union, concatenation, and complement.
For any two $A, B \in \mathcal{P}$, let M_1 and M_2 be the deterministic single-tape Turing machines that decide them in polynomial time. We construct M' a deterministic single-tape Turing machines that decides the union of $A$and B in polynomial time.
"On input $w \in A \cup B$:

1. Run M_1 on w if it accepts then M' accepts. If M_1 rejects then move onto step 2.
2. Run M_2 on w if it accepts then M' accepts. If M_2 rejects then M' rejects."

For any two $A, B \in \mathcal{P}$, let M_1 and M_2 be the deterministic single-tape Turing machines that decide them in polynomial time. We construct M' a deterministic single-tape Turing machines that decides the intersection of $A$and B in polynomial time.
"On input $w \in A \cap B$:

1. RunM_1 and M_2 on w.
2. If either rejects, then M' rejects.
3. If both accept, then M' accepts."

For any two $A, B \in \mathcal{P}$, let M_1 and M_2 be the deterministic single-tape Turing machines that decide them in polynomial time. We construct M' a deterministic single-tape Turing machines that decides the concatenation of $A$and B in polynomial time.
"On input $w \in A \circ B$:

1. Divide w into two substrings where $w = w_1 \circ w_2$, where $w_1 \in A$ and $w_2 \in B$.
2. Run M_1 on w_1. Run M_2 on w_2. Because M_1 and M_2 are deciders the TMs do not need to be run in lockstep unison.
3. If M_1 and M_2 accept then M' accepts.
4. This process needs to be repeated for every possible subdivision of w. All of those subdivisions need to be run against M_1 and M_2 in parallel.
5. M' rejects if all subdivisions reject."

For any $A \in \mathcal{P}$, let M_1 be the deterministic single-tape Turing machine that decides A in polynomial time. We construct M' a deterministic single-tape Turing machines that decides the complement of A in polynomial time.
"On input $w \in \overline{A}$:

1. Run M_1on w.
2. If M_1 accepts, then M' will reject.
3. If M_1 rejects, then M' will accept."

7.7 Show that NP is closed under union and concatenation.
For any $A, B \in \mathcal{NP}$, let M_1 and M_2be the nondeterministic single-tape Turing machines that decide them in polynomial time. We construct M' a nondeterministic single-tape Turing machines that decide the union of A and B in polynomial time.
"On input $w \in A \cup B$:

1. Run M_1 on w if it accepts then M' accepts. If M_1 rejects then move onto step 2.
2. Run M_2 on w if it accepts then M' accepts. If M_2 rejects then M' rejects."

For any two $A, B \in \mathcal{NP}$, let M_1 and M_2 be the nondeterministic single-tape Turing machines that decide them in polynomial time. We construct M' a nondeterministic single-tape Turing machines that decides the concatenation of $A$and B in polynomial time.
"On input $w \in A \circ B$:

1. Divide w into two substrings where $w = w_1 \circ w_2$, where $w_1 \in A$ and $w_2 \in B$.
2. Run M_1 on w_1. Run M_2 on w_2. Because M_1 and M_2 are deciders the TMs do not need to be run in lockstep unison.
3. If M_1 and M_2 accept then M' accepts.
4. This process needs to be repeated for every possible subdivision of w. All of those subdivisions need to be run against M_1 and M_2 in parallel.
5. M' rejects if all subdivisions reject."

7.8 Let CONNECTED = {⟨G⟩| G is a connected undirected graph}. Analyze the algorithm given on page 185 to show that this language is in P.
M = "On input $\langle G \rangle$, the encoding of a graph G:

1. Select the first node of G and mark it.
2. Repeat the following stage until no new nodes are marked:
    (a) For each node in G, mark it if it is attached by an edge to a node that is already marked.
3. Scan all the nodes of G to determine whether they all are marked. If they are, accept ; otherwise, reject ."

Let m be the number of nodes in G. Stages 1 and 3 are executed only once. Stage 2.a runs at most m times because each time except the last it marks an additional node in G. Thus, the total numbers of stages used is at most $1 + 1 + m$, giving a polynomial in the size of G. Stages 1 and 3 of M are implemented in polynomial time easily. Stage 2.a involves looking at each node in G and testing whether it is attached to node that already marked, which also is easily implemented in polynomial time. Therefore M is a polynomial time algorithim for CONNECTED

7.10 Show that $\mathrm{ALL}_{\mathrm{DFA}}$ is in P.
Let $\mathrm{ALL}_{\mathrm{DFA}} = \{\langle A \rangle | A \text{ is DFA and} L(A) = \Sigma^*\}$
A DFA accepts some string iff reaching an accept state from the start state by traveling along the arrows of the DFA is possible. To test this condition, we can design a TM T that uses a marking algorithm.
T = "On input $\langle A \rangle$, where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states are marked:
    (a) Mark any state that has a transition coming into it from any state that is alread marked.
3. If no accept state is marked, reject; otherwise accept."

Stage 1 executes once. Stage 3 has to check each accept state to see if any have been marked, this will take $|F|$ steps (the number of accept states in the DFA). Stage 2.a runs at most $|Q|$ times (the number of states in the DFA), because each time xcept the last it mark an additional state in A. Stage 2.a involves looking at each state n the DFA and testing whether its has a transition coming into it from a state that is already marked, this can be implemented in polynomial time. Therefore $T$is a polynomial time algorithim for $\mathrm{ALL}_{\mathrm{DFA}}$.