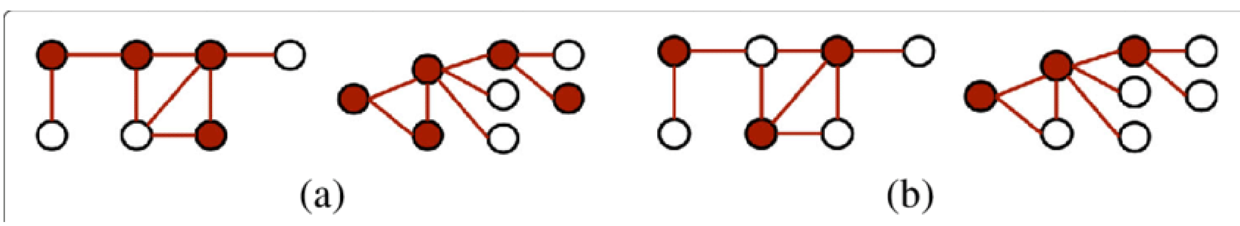
Theory of Computation: Assignment 11

Arjun Chandrasekhar

Due Thursday, 04/21/2022 at 11:59 pm (50 points)

1. Let L_1 and L_2 be formal languages, with $L_1, L_2 \in \text{NTIME}(n^c)$
   (a) (5 points) Prove that $L_1 \cup L_2 = \{w | w \in L_1 \text{ or } w \in L_2\} \in \text{NTIME}(n^c)$
   (b) (5 points) Prove that $L_1 \circ L_2 = \{w | w = xy \text{ for some } x \in L_1, y \in L_2\} \in \text{NTIME}(n^c)$
   (c) (0 points) Convince yourself that parts (a) and (b) establish that NP is closed under union and concatenation. You don't need to write anything down, but make sure you understand why this is true.
2. For this problem we will study the relationship between P, NP, and EXP.
   (a) (5 points) Prove that $\text{P} \subseteq \text{NP}$
   (b) (5 points) Prove that $\text{NP} \subseteq \text{EXP}$ (**Hint:** Construct a machine that tries out all possible certificates for a string w.)
3. Let. $G = (V, E)$ be a graph. A **vertex cover** is a set of vertices $C \subseteq V$ such that every edge in the graph touches one of the vertices in C.

   Here are some examples of vertex covers. Notice how every edge is adjacent to at least one red vertex.

   Formally, we will work with the following language

   $$\text{VERTEX-COVER} = \{\langle G, k\rangle | G \text{ is a graph that has a vertex cover of size k}\}$$

   In the textbook, Sipser proves that this language is NP-complete by giving a reduction from 3-SAT. We will give another proof that it is NP-complete by reducing from IND-SET.

   (a) (5 points) Prove that $\text{VERTEX-COVER} \in \text{NP}$ (**Hint:** create machine that either guess a vertex cover, or verifies that a proposed cover is valid and big enough)
   (b) (5 points) Prove that for any graph G, a set of vertices C is a vertex cover if and only if $V \backslash C$ is an independent set.
   (c) (5 points) Prove that IND-SET $\leq_{\text{poly}}$ VERTEX-COVER. This will prove that VERTEX-COVER is NP-Hard. Since we already proved that $\text{VERTEX-COVER} \in \text{NP}$, this will establish that it is NP-complete.

4. Consider the following language

$$\text{PARTITION} = \left\{ S = (s_1, s_2, \dots s_n) | \text{There is a subset } R \subseteq S \text{ where } \sum R = \sum S \backslash R \right\}$$

We are given a set of numbers. We want to determine if we can partition the set into two sets such that the two partitions add up to the same amount. For example

- $(1, 1, 2, 4, 6) \in$ PARTITION because we can partition it into $(1, 2, 4)$ and $(1, 6)$, both of which add up to the same number
- $(3, 3, 3, 3, 3) \notin$ PARTITION because no matter how you split it up there will be more 3s in one set than the other.

(a) (5 points) Show that PARTITION $\in$ NP. (**Hint:** construct a machine that either guesses a partition or verifies that a given partition is valid)

(b) (5 points) Let $T = 2B - (\sum s_i)$. Prove that S has a subset that adds up to B if and only if $S \cup \{T\}$ has a partition.

(c) (5 points) Show that PARTITION is NP-Hard. This, along with the part (a), will establish that PARTITION is NP-complete (**Hint:** reduce from SUBSET-SUM. Make use of the result from part (b)).