# Tutorial 8

## Time Complexity Classes

1. Prove that the following languages (defined over graphs) are in **P**.
   (a) BIPARTITE – the set of all bipartite graphs. That is, $G = (V, E) \in$ BIPARTITE if $V$ can be partitioned into two sets $V_1, V_2$ such that every edge in $E$ is adjacent to a vertex in $V_1$ and a vertex in $V_2$.

   **Solution:** Choose a vertex $v \in V$ and perform a breadth-first search (BFS) starting at $v$ as the root, colouring vertices visited in odd interations red and those visited in even iterations black. At the end, if there exists an edge whose end points are both coloured with the same colour, then $G \notin$ BIPARTITE. Else, $G \in$ BIPARTITE. In the worst case, this algorithm would take $O(n^2)$ time where $n = |V|$.

   (b) TRIANGLE-FREE – the set of all graphs that do not contain a triangle (where triangle is a set of three distinct vertices that are mutually connected).

   **Solution:** Let $n = |V|$. Check all $\binom{n}{3}$ sets of vertices and check if they are mutually connected. If none of them are, then $G \in \mathsf{TRIANGLE-FREE}$. Since $\binom{n}{3} = O(n^3)$, the algorithm runs in polynomial time.

2. Normally, we assume that numbers are represented as strings using the binary basis. That is, a number $n$ is represented by the sequence $x_0, x_1, \ldots, x_{\log n}$ such that $n = \sum_{i=0}^{\log n} x_i 2^i$. However, we could have used other encoding schemes. If $n \in \mathbb{N}$ and $b \geq 2$, then the representation of $n$ in base $b$, denoted by $\llcorner n \lrcorner_b$ is obtained as follows: first represent $n$ as a sequence of digits in $\{0, \ldots, b-1\}$, and then replace each digit by a sequence of zeroes and ones. The unary representation of $n$, denoted by $\llcorner n \lrcorner_1$ is the string $1^n$ (i.e., a sequence of $n$ ones).
   (a) Show that choosing a different base of representation (other than unary) will make no difference to the class **P**. That is, show that for every subset $S$ of the natural numbers, if we define $L_S^b = \{\llcorner n \lrcorner_b : n \in S\}$ then for every $b \geq 2$, $L_S^b \in \mathbf{P}$ iff $L_S^2 \in \mathbf{P}$.

   **Solution:** Let $D(k)$ (resp. $M(k)$) be the time complexity of division (resp. multiplication) of two $k$-bit numbers. Suppose that $L_S^b \in \mathbf{P}$. Then there exists a Turing machine $\mathcal{M}$ that runs in polynomial time recognising $L_S^b$. Any $k$-bit input $\llcorner x \lrcorner_2$ ($x \in \mathbb{N}$) can be converted to base-$b$ representation in time $D(k) \cdot \log_b x$, which is polynomial in $k$. The resulting string $\llcorner x \lrcorner_b$ can then be provided as input to $\mathcal{M}$, thereby recognising $L_S^2$. In other words, $L_S^b \in \mathbf{P}$ implies $L_S^2 \in \mathbf{P}$. Similarly, an input string of length $k$ for $L_S^b$ (i.e., a base-$b$ number) can be converted to its binary representation in $M(k) \cdot \log_b x$ time. This shows that $L_S^2 \in \mathbf{P}$ implies $L_S^b \in \mathbf{P}$.

   (b) Show that choosing the unary representation makes a difference by showing that the following language is in **P**.

$$\mathsf{UNARYFACTORING} = \{\langle \llcorner n \lrcorner_1, \llcorner k \lrcorner_1 \rangle : \text{there is a } j \leq k \text{ dividing } n\}.$$

**Solution:** Check for divisibility of $n$ by all numbers $\leq k$. This takes $k$ steps which is polynomial in the size $(n+k)$ of the input.

3. Prove that $\mathbf{P} = \mathbf{coP}$ and $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{coNP}$.

**Solution:** Let $L \in \mathbf{P}$. Then $\neg L$ can be decided in polynomial time – run the TM deciding $L$ and flip its decision. So, $\mathbf{P} = \mathbf{coP}$.

We know that $P \subseteq \mathbf{NP}$. Now, a language $L \in \mathbf{coNP}$ iff $\neg L \in \mathbf{NP}$. Let $L \in P$. Then $\neg L \in \mathbf{coP} = \mathbf{P} \subseteq \mathbf{NP}$. So, by the definition of $\mathbf{coNP}$, $\neg\neg L = L \in \mathbf{coNP}$. Therefore, $\mathbf{P} \subseteq \mathbf{coNP}$. □

4. Assuming $\mathbf{NP} \neq \mathbf{coNP}$, show that no $\mathbf{NP}$-complete problem can be in $\mathbf{coNP}$.

**Solution:** Suppose that $L$ is an $\mathbf{NP}$-complete language and $L \in \mathbf{coNP}$. Then for every language $L' \in \mathbf{NP}$, we have $L' \leq_p L$ and since $L \in \mathbf{coNP}$, $L' \in \mathbf{coNP}$. So, we have $\mathbf{NP} \subseteq \mathbf{coNP}$. Now consider any $L' \in \mathbf{coNP}$. Then $\neg L' \in NP$ and $\neg L' \leq_p L$. Using the same reduction, we can say $L' \leq_p \neg L$ and $\neg L \in \mathbf{NP}$, since $L \in \mathbf{coNP}$. So, we have $L' \in \mathbf{NP}$ and hence $\mathbf{coNP} \subseteq \mathbf{NP}$, thus implying that $\mathbf{NP} = \mathbf{coNP}$ – contradiction! □

5. Show that the halting problem is $\mathbf{NP}$–hard.

**Solution:** HP is $\mathbf{NP}$–hard if $L \leq_p$ HP for all $L \in \mathbf{NP}$. Equivalently HP is $\mathbf{NP}$–hard if SAT $\leq_p$ HP. We show a polynomial time computable function mapping a formula $\phi$ to a TM $\mathcal{N}$ and a string $x$ such that $\phi \in SAT$ iff $(\mathcal{N}, x) \in$ HP. Let $\mathcal{M}$ be a polynomial time deterministic machine that takes as input a formula $\phi$, an assignment $z$ and accepts iff $z$ satisfies $\phi$. Such a TM exists since SAT $\in \mathbf{NP}$. Construct $\mathcal{N}$ so that on input $x$ it does the following:

- Parse $x$ as a Boolean formula over $n$ vairables.
- For all assignments $z \in \{0,1\}^n$, run $\mathcal{M}(x, z)$ to check if $z$ satisfies $x$; accept and halt if $\mathcal{M}$ accepts.
- If $x$ is not satisfied by any $z$, enter a trivial loop.

Map $\phi$ to $\mathcal{N}, \phi$. This map can be computed in time polynomial in $|\phi|$ (the size of $\phi$) irrespective of fact that $\mathcal{N}$'s running time would be exponential in the $|\phi|$. Now, $\phi \in$ SAT iff $\exists z$ such that $\mathcal{M}(\phi, z)$ accepts iff $\mathcal{N}$ halts on input $\phi$ iff $(\mathcal{N}, \phi) \in$ HP. This completes the reduction SAT $\leq_p$ HP.

6. Let

$$\text{DOUBLESAT} = \{\langle\phi\rangle : \phi \text{ is a CNF formula having at least two satisfying assignments}\}.$$

Show that DOUBLESAT is $\mathbf{NP}$–complete.

**Solution:** DOUBLESAT $\in \mathbf{NP}$ – the certificate is a pair of assignments of $0, 1$ values to the variables of $\phi$, each of which can be checked in polynomial time (in fact, in linear time) for whether they satisfy $\phi$ or not.

To show that DOUBLESAT is $\mathbf{NP}$–hard, we show that SAT $\leq_p$ DOUBLESAT. Let $\phi$ be an instance of SAT defined over $n$ variables $u_1, \ldots, u_n$. The reduction picks a new variable $u_{n+1}$

and generates $\phi' = \phi \wedge (x \vee \bar{x})$ as the instance of DOUBLESAT. If there's a satisfying assignment $z \in \{0,1\}^n$ for $\phi$, then there are two assignments $z_1 = z||0$ (indicating $u_{n+1} = 0$) and $z_2 = z||1$ (indicating $u_{n+1} = 1$) for $\phi'$. So $\phi \in$ SAT $\implies$ $\phi' \in$ DOUBLESAT. Now, if $\phi'$ has 2 satisfying assignments, all clauses except the last one, which is nothing but $\phi$, would be satisfiable by at least one assignment. Therefore, $\phi' \in$ DOUBLESAT $\implies$ $\phi \in$ SAT.

7. (a) A *vertex cover* in a graph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that every edge of $G$ is incident on at least one vertex in $S$. Show that the language

$$\text{VERTEXCOVER} = \{(G, k) \mid \text{graph } G \text{ has a vertex cover of size } \leq k\}$$

is **NP**-complete.

**Solution:** VERTEXCOVER is in **NP** – given a subset $S \subseteq V$ of size $\leq k$, it can be verified in $O(|E|)$ time whether $S$ is a vertex cover.

To show that VERTEXCOVER is **NP**-hard, we provide a reduction from 3-SAT. Let $\phi$ be an instance of 3-SAT, consisting of $m$ clauses over $n$ variables $u_1, u_2, \ldots, u_n$. The reduction will map $\phi$ to a graph $G = (V, E)$ and integer $k$ with $|V| = 3m+2n$, $|E| = 6m+n$ and $k = 2m+n$. $G$ is constructed as follows. $2n$ vertices correspond to all literals $u_1, \neg u_1, u_2, \neg u_2, \ldots, u_n, \neg u_n$ and $3m$ vertices correspond to the literals present in each clause i.e., for each clause $(x_1 \vee x_2 \vee x_3)$, $V$ would contain 3 separate vertices corresponding to the literals $x_1, x_2, x_3$. Include in $E$ the following edges: $\{u_i, \neg u_i\}$ in $E$ for all $i \in [1, n]$; $\{x_1, x_2\}, \{x_2, x_3\}, \{x_1, x_3\}$ for each clause $(x_1 \vee x_2 \vee x_3)$; and finally for each clause $(x_1 \vee x_2 \vee x_3)$, include edges $\{x_i, u_k\}$ (or $\{x_i, \neg u_k\}$) if $x_i = u_k$ (or $x_i = \neg u_k$).

Suppose that $\phi \in$ 3-SAT. Define the set $S$ as follows: if $u_i = 1$, then include vertex $u_i$ in $S$; otherwise include vertex $\neg u_i$ in $S$. This ensures that all edges of the form $\{u_i, \neg u_i\}$ are covered. Each clause $(x_1 \vee x_2 \vee x_3)$ contains at least one true literal, say $x_1$. The edge connecting $x_1$ to its equivalent literal is already covered by the first set of edges added to $S$. Now include the vertices corresponding to the other two literals $x_2, x_3$ in $S$. This would ensure that the edges connecting $x_1, x_2, x_3$ and those connecting $x_2, x_3$ to the equivalent literals are covered. $S$ therefore is a vertex cover of $G$ and contains $n + 2m = k$ vertices as required.

Suppose that $G \in$ VERTEXCOVER i.e., $G$ has a vertex cover $S$ of size $k = n + 2m$. $S$ musst contain either $u_i$ or $\neg u_i$ for each $i$ in order to cover edges $\{u_i, \neg u_i\}$. Also $S$ must contain 2 vertices from each triangle corresponding to clauses. Now, assign $u_i = 1$ if $u_i \in S$ and $u_i = 0$ if $\neg u_i \in S$. We now argue that this is a satisfying assignment for $\phi$. Consider a clause $x_1 \vee x_2 \vee x_3$ of $\phi$. Two of the vertices in the corresponding triangle would be in $S$. Without loss of generalisty, suppose that $x_1, x_2 \in S$ and $x_3 = u_k$. The edge $\{x_3, u_k\}$ must be covered by $u_k$ and not $x_3$. That is $u_k \in S$ and is hence set to 1 as a result of which the clause evaluates to 1. Since this holds for all clauses, $\phi \in$ 3-SAT.

(b) Let $S$ be a set and let $C = \{X_1, \ldots, X_n\}$ be a collection of $n$ subsets of $S$ (for each $i \in [1, n]$, $X_i \subseteq S$). A set $S'$, with $S' \subseteq S$, is called a hitting set for $C$ if every subset in $C$ contains at least one element in $S'$, i.e., $|X_i \cap S'| \geq 1$ for each $i \in [1, n]$. Let HITSET be the language $\{\langle C, k\rangle : C \text{ has a hitting set of size } k\}$. Prove that HITSET is **NP**–complete.

**Example** $S = \{a, b, c, d, e, f, g\}$, $C = \{\{a, b, c\}, \{d, a\}, \{d, e, f\}, \{g\}\}$

- $k = 2$, no hitting sets exist.
- $k = 3$, $S' = \{a, d, g\}$ (other choices exist).

**Hint:** Try reducing from VERTEXCOVER.

**Solution:** Clearly, HITSET $\in$ **NP** – guess a set $S'$ of size $k$ and for each set $X_i \in C$, check for inclusion of elements of $S'$ in $X_i$. This can be done in $nk$ time.

We now show that VERTEXCOVER $\leq_p$ HITSET. We need to transform an instance $\langle G = (V, E), k\rangle$ into an instance $\langle S, C, k\rangle$ such that $\langle G = (V, E), k\rangle \in$ VERTEXCOVER iff $\langle S, C, k\rangle \in$ HITSET. Set $S = V$ and for each edge $\{u, v\} \in E$, create a set $X_{u,v} = \{u, v\}$ and add it to $C$.

Suppose that $V' \subseteq V$ is a vertex cover for $G$ fo size $k$, then for every edge $\{u, v\} \in E$ either $u \in V'$ or $v \in V'$. That is, $|X_{u,v} \cap V'| \geq 1$ thus giving us a hitting set $S' = V'$ of size $k$. Suppose that $S' \subseteq S$ is a hitting set of size $k$ for $C$. Then $|S' \cup X_{u,v}| \geq 1$ for every $X_{u,v}$ with $\{u, v\} \in E$, that is, choosing $V' = S'$ ensures that $V'$ contains at least one end-point of every edge.

8. (Scaling recource bounds.) Let $\mathbf{CL}_1, \mathbf{CL}_2$ denote some time/space complexity classes. Show that if $\mathbf{CL}_1(f(n)) \subseteq \mathbf{CL}_2(g(n))$, then $\mathbf{CL}_1(f(n^c)) \subseteq \mathbf{CL}_2(g(n^c))$.

   **Solution:** Let $L \in \mathbf{CL}_1(f(n^c))$ and let $\mathcal{M}$ be a TM deciding $L$ with resource bound $f(n^c)$ i.e., $\mathcal{M}$ decides $L$ in $f(n^c)$ time/space. The statement trivially holds for $c = 1$. For $c \geq 2$, define

   $$L' = \{\langle x \| 0 \| 1^{|x|^c - |x| - 1}\rangle : x \in L\}.$$

   Define $\mathcal{M}'$ that accepts input $y$ iff $y$ is of the form $x\|0\|1^{|x|^c-|x|-1}$ and $\mathcal{M}$ accepts $x$. $\mathcal{M}'$ runs in time/space $O(f(|x|^c))$. But $O(f(|x|^c)) = O(f(|y|))$ implying that $L' \in \mathbf{CL}_1(f(n)) \subseteq \mathbf{CL}_2(g(n))$. Hence there is a TM $\mathcal{N}'$ deciding $L'$ using $g(n)$ time/sapce. Define a machine $\mathcal{N}$ deciding $L$ as follows: $\mathcal{N}(x) = \mathcal{N}'(x\|0\|1^{|x|^c-|x|-1})$. Clearly $\mathcal{N}$ runs in time/space $g(|x01^{|x|^c-|x|-1}|) = g(|x|^c)$. Therefore $L \in \mathbf{CL}_2(g(n^c))$. □

9. The following two classes are exponential time analogues of **P** and **NP**.

   $$\mathbf{EXP} = \cup_{c\geq 1}\mathbf{DTIME}(2^{n^c})$$
   $$\mathbf{NEXP} = \cup_{c\geq 1}\mathbf{NTIME}(2^{n^c})$$

   Clearly, $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP} \subseteq \mathbf{NEXP}$. Show that if $\mathbf{EXP} \neq \mathbf{NEXP}$, then $\mathbf{P} \neq \mathbf{NP}$.

   **Hint:** Consider padding strings in **EXP**/**NEXP** languages with exponentially sized strings in order to "scale down" to **P**/**NP**.

   **Solution:** We show that if $\mathbf{P} = \mathbf{NP}$, then $\mathbf{EXP} = \mathbf{NEXP}$. Let $L \in \mathbf{NTIME}(2^{n^c})$ and let $\mathcal{M}$ be a Turing machine deciding $L$. Define language $L'$ as follows.

   $$L' = \{\langle x \| 1^{2^{|x|^c}}\rangle : x \in L\},$$

   where $\|$ denotes concatenation. We show that $L' \in \mathbf{NP}$ by constructing an NDTM $\mathcal{M}'$ deciding $L'$. Given input $y$, $\mathcal{M}'$ checks whether $y$ is of the form $z||1^{2^{|z|^c}}$ for some string $z$. If not, reject. Otherwise, run $\mathcal{M}$ on $z$ for $2^{|z|^c}$ steps and output its decision. The running time of $\mathcal{M}'$ is $O(|z| + 2^{|z|^c})$ which is polynomial in $|y|$. Hence, $L' \in \mathbf{NP}$. If $\mathbf{P} = \mathbf{NP}$, then $L' \in \mathbf{P}$ i.e., there is a deterministic TM $\mathcal{M}''$ deciding $L$ in polynomial time. If $L' \in \mathbf{P}$, then $L \in \mathbf{EXP}$ – given any $x \in L$, pad $x$ with $1^{2^{|x|^c}}$ and provide the resulting string as input to $\mathcal{M}''$. If $\mathcal{M}''$ accepts, then accept; else, reject. Thus we can decide membership in $L$ deterministically in exponential time. Therefore, $\mathbf{EXP} \subseteq \mathbf{NEXP}$ and as a result $\mathbf{EXP} = \mathbf{NEXP}$. □