#1 Line 'Em Up

Arrange the following complexity classes in an ascending chain (ordered by containment).

$$\mathcal{O}(\log n), \mathcal{O}(n), \mathcal{O}(n^2), \mathcal{O}(n^n), \mathcal{O}(n^3), \mathcal{O}(n \log n), \mathcal{O}(n!)$$

I'll start for you...

$$\mathcal{O}(\log n) \subseteq \ldots$$

#2 Representation (again)

For each of the following objects, describe a plausible representation for those objects as strings.

1. An undirected graph G.
2. An assignment of truth values to a finite collection of boolean variables $x_1, \ldots, x_k$.
3. A deterministic finite automata D.

#3 Analyze This

Consider the following language of descriptions of DFAs that accept all strings.

$\text{ALL}_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a DFA and } L(D) = \Sigma^* \}$.

I will first give an algorithm that decides this language.

#3.1 The Algorithm

For our purposes, it is enough to presume the input has Q as some list of states and that it's easy (i.e., constant-time) to check if δ connects state q to state q'. Our strategy is to start at state q_0 and "mark each state it can reach" with a stone. Then repeat until we have marked no new states. More explicitly:

1. Write the state q_0 to some intermediate work tape (Call it WT1).
2. Do one sweep of the δ portion of the input-tape and, for each state q' such that δ maps q_0 to q', add q' to a new section of WT1.
3. Repeat the above process for each state just added, making sure to not add duplicate states. For example, if q_1 and q_2 were reachable from q_0, Then do a linear sweep of the δ portion of the input tape checking for all (new) states reachable from q_1 and q_2. Add these states to a new section of WT1, making sure they do not already occur on WT1.
4. Repeat until no new states are added.
5. Check if all states on WT1 are accepting states—do a linear pass of WT1 and check if each state q on WT1 is in the F portion of the input tape.

#3.2 The Analysis

Perform a worst-case complexity analysis of the above algorithm. Is $\textsc{All}_{\text{DFA}}$ in P? Justify your analysis by analyzing each "part" of computation done in the algorithm.

#4 Prove a language is in P

Show that each of the following languages are in P. If possible, separate your description of the algorithm from its complexity analysis.

1. $\{ \langle G, V' \rangle \mid G = (V, E)$ is a graph, $V' \subseteq V$ and no two vertices of V' are incident with each other $\}$.
2. $\{ \langle G \rangle \mid G$ is a graph and G contains a triangle (a 3-clique) $\}$.