Lecture Notes ——- Week 11

There is no Lecture 21 (due to exam 2).

Lecture 22: Concept of NP

We now switch to Chapter 8: NP-completeness. We define \( \mathcal{P} \) as the set of problems that have polynomial-time algorithms. We define \( \mathcal{NP} \) as the set of problems that we can verify proposed solutions in polynomial-time. The key concept discussed in class is the concept of verifying a proposed solution. In graph 3-coloring problem, for example, a proposed solution is the coloring of nodes of the graph. Remember in this problem we need to assign one of the three colors to the nodes s.t. no edge has the same color at its two end nodes. We can easily check whether the coloring satisfies the coloring property. So this problem is in \( \mathcal{NP} \). On the other hand, no one knows polynomial-time algorithm to find a legal coloring for a graph. So it is not known whether this problem is in \( \mathcal{P} \) or not. Another example is the Hamiltonian cycle problem. We are given a graph, and we want to find a path that visits each exactly once and return to the starting node. Again, no one knows efficient algorithms for this problem, while it is easy to see this problem is also in \( \mathcal{NP} \).

The concept of NP is important: a problem is in NP if this problem can be verifiable in polynomial-time. Note: we are not asking for efficient finding of the solution, but rather just verifying whether a proposed solution is legal or not.

As an example, consider the problem of graph 3-coloring, where we are given a graph \( G \) and we want to find coloring each node with three colors s.t. no edge has same color on its two ends. No efficient algorithm is known for this problem. So instead of looking for a valid coloring, you are given a coloring (each node is colored), and this coloring is a proposed solution. Is this a legal solution (i.e. the coloring)? It is easy to see we can verify this coloring property in polynomial time: for each of the \( m \) edges, we check to see if its two nodes are colored the same or not; if any edge has same color on its two ends, the solution is not legal. If no such violation is found, the proposed solution is good. For a graph with \( m \) edges, this takes \( O(\text{m}) \) time and so verification can be done in polynomial time. That is, graph 3-coloring is in NP.

Our next problem is the satisfiability problem. We are given a boolean formula in so-called Conjunctive normal form (CNF). A CNF formula contains several clauses (connected by logical and), where in each clause we apply logical or to connect several literals. Literal means a boolean variable \( x \) or its negation \( \overline{x} \). Example:

\[
\Phi = (x \lor y \lor z) \land (x \lor \overline{y}) \land (y \lor \overline{z}) \land (z \lor \overline{x}) \land (\overline{x} \lor \overline{y} \lor \overline{z})
\]

Satisfying truth assignment: a setting of boolean variables s.t. each clause is evaluated to be true (i.e. for each clause, at least one literal is true). In class, I explained that this formula does not have satisfying truth assignment. Study this if you forget about the argument. The satisfiability problem asks whether there exists satisfying truth assignment for a given formula. Again, no efficient algorithm is known for satisfiability (SAT) problem. But verifying a proposed truth assignment can be easily done in polynomial time.

Another example is Hamiltonian cycle: given a directed graph \( G(V,E) \), find a path that visit each node exactly once and return to the origin. Again, no efficient algorithm is known for this problem. Now what if I proposed a tour of \( v_1, v_2, v_3 \ldots v_n \) as the cycle? It is not hard to see we can easily verify whether this proposed solution is legal or not in polynomial time. Thus, HAM-CYCLE is also in NP.