Second Level Algorithms

Prof. Palash Dey

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

Week – 12

Lecture 58

Welcome to the 58th lecture of the second-level algorithm course. From the last couple of lectures, we have been studying NP-completeness. In this lecture, we will prove some very important problems to be NP-complete. So, let us begin. So, in this lecture, we will prove this theorem that 3SAT is NP-complete. So, let us first recall the definition of 3SAT. SAT stands for satisfiability. The input to 3SAT is a Boolean formula in 3CNF form, in 3CNF representation. So, what is 3CNF representation? That means the formula F is a logical AND of some m number of clauses, where each clause is an OR of three literals, OK. And the question that we are interested in is whether the formula F is satisfiable. In the last class, we concluded that CNF-SAT is NP-complete. There was some bug in the proof.

I take it as a homework to find out that we only prove that formula set is or formula satisfiability is NP-complete, not CNF satisfiability. Today we will show that 3 CNF formula is that means, 3 SAT is NP complete. So, in particular satisfiability is or CNF satisfiability will also be NP complete that will follow from this theorem because CNF SAT is a generalization of 3 SAT and if a special case is NP complete then the general problem is also NP complete. So, let us prove this theorem. 3 set clearly belongs to NP.

Why because we can take a satisfying assignment as a certificate for the YES instances which can be easily verified by a polynomial time algorithm that it is indeed a satisfying assignment for the formula that formula indeed satisfies all the clauses $C_1, \ldots, C_m$. To prove NP hardness we reduce from circuit set. which is known to be NP complete thanks to Cook-Levin theorem. So, let us take an arbitrary instance of circuit set let I_1 be an arbitrary of circuit set. What we first do? We convert I_1 to an equivalent instance of circuit set let us call it I_2, where the fan in of every gate is at most 2. How? So, suppose there is a gate, suppose there is a OR gate with L fan-in.

So, fan-in is the number of input gates or input connections of the gate. So, then this gate we replace it as the first one, or the first one takes input x_1, x_2. OK, the size of I_2, which is the number of gates plus the number of connections, is polynomial. In the size of I_1, and clearly I_1 and I_2, this transformation is equivalent. y_1 here, or y here, is 1 if and only if in the reduced instance I_2, y is 1. OK. So, now, we do the standard reduction from circuit set to formula set. Now, the standard reduction from circuit set to formula set produces the instance I_3 as follows. OK. How does I_3 look like?

If you recall, it has y_1 and y_2 if and only if something, and y_3 if and only if something. y_k if and only if something, right? And because the fan-in is at most 2. So, the number of literals here is the fan-in. of the corresponding gate, which is at most 2 in I_2. OK. So, now, our formula is in the form of a logical AND of some number of clauses.

And each clause has at most three variables. The problem is each clause may not contain exactly three variables. So now, we transform each clause into a logical AND of some number of clauses where each clause is an OR of three literals. So, let $f(x_1, x_2, x_3)$.

So, each clause here involves some one y_i's and two more literals, at most three literals. So, let $f(x_1, x_2, x_3)$ be any clause. So, now for concreteness, let us take a concrete example and see how we transform it into 3-CNF format. So, suppose $f(x_1, x_2, x_3)$ is, say, (x_1 AND x_2) OR (NOT x_3). It does not matter what it is.

What do we do? We write down the truth table of f. So, here is x_1, x_2, x_3, and this is $f(x_1, x_2, x_3)$. So, I have listed all eight possible values of x_1, x_2, x_3 and will evaluate them here. So, if x_1 is false, then f is false, whereas if x_1 is true, then either x_2 needs to be true or x_3 needs to be false. So, if x_2 is true, then the formula is 1, and if x_3 is false. So, this is the truth table of the function f. What do we do? We write down f complement from the truth table in disjunctive normal form or some So, when is $\overline{f}$ 1? $\overline{f}$ is 1 when f is 0, that means when x_1 is 0, x_2 is 0, x_3 is 0.

So, we will use alternatively plus sign with logical OR and product sign with logical AND. So, at least write down the zeros. This is f bar. So, f is the complement of this expression, and using De Morgan's law, this expression, which is in conjunctive normal form (3CNF), and because this truth table has 8 rows, the number of clauses in the 3CNF format for each formula will have at most 8 clauses. Hence, when we replace each clause of this formula I_3 with 3CNF format in 3CNF formula equivalent 3CNF formula, then the blow-up in size is at most 8 times. The size of I_4, which is the sum of the number of variables and clauses in it, is at most 8 times the size of I_3. Now, since size of I_3 was polynomial in size of I_2 and thus size of size of I_3 which was polynomial in the size of I_1, the size of I_4. is bounded by some polynomial size of I_1 ok.

So, the out so, the size of the equivalent instance of 3 set is polynomially bounded of the size of the initial instance I_1. Also all the reductions from I_1 to I_2, from I_2 to I_3 and from I_3 to i4. run in polynomial time ok. So, this is a polynomial time reduction and from the construction it follows that I_1 is satisfiable if and only if I_2 is satisfiable I_2 is satisfiable if and only if I_3 is satisfiable and I_3 is satisfiable if and only if I_4 is satisfiable ok. So, I_1 and I_4 are equivalent of respectively circuit set and 3 set.

So, this shows the reduction from circuit SAT to 3SAT, which proves that 3SAT is NP-hard. We have already seen that 3SAT belongs to NP; hence, 3SAT is NP-complete. Now, we can derive the corollary that CNF-SAT is also NP-complete. Why?

Now, a one-line proof: because CNF-SAT is not a special case but a generalization of 3SAT. It is also NP-hard. Why? Because the same reduction or we can reduce it from 3SAT; we do not need to change the instance. Also, I_1 can be the same as I_2.

If I_1 is an instance of 3SAT, I_2 can be the same as I_1, which is a valid instance of CNF-SAT. Obviously, I_1 is equivalent to I_2. So, NP-hardness is obvious, but the membership in NP is also obvious for CNF-SAT. Since a satisfying assignment is a certificate that can be verified in polynomial time for YES instances. So, let us stop here.

Thank you.