CSCI 270 Lecture 24: NP-Complete Problems

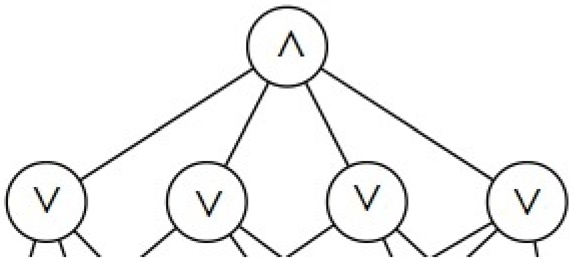
3-SAT

We have a set of n variables $x_1, x_2, ..., x_n$, and a set of m clauses $C_1, C_2, ..., C_m$. Each clause is a disjunction of exactly 3 variables or their negation, such as $C_1 = (x_1 \vee \neg x_2 \vee x_4)$.

A 3-SAT formula is a conjunction of all clauses $C_1 \wedge C_2 \wedge ... \wedge C_m$, such as:
$(x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_4)$.

We want to know if there is an assignment of boolean values to variables such that the formula evaluates to true.

- Is there a satisfying assignment for this example?
- Is 3-SAT $\in NP$?
- What's wrong with the following reduction?



Reduction from Circuit-SAT to 3-SAT:

1. Add variable names to every wire in the Circuit-SAT problem.
2. Hard-code output: $C_1 = (x_6)$
3. Transform not-gates. $x_5 = \neg x_3$ becomes $C_2 = (x_3 \vee x_5)$ and $C_3 = (\neg x_3 \vee \neg x_5)$
4. Transform or-gates. $x_6 = x_4 \vee x_5$ becomes $C_4 = (x_6 \vee \neg x_4)$ and $C_5 = (x_6 \vee \neg x_5)$ and $C_6 = (\neg x_6 \vee x_4 \vee x_5)$
5. Transform and-gates. $x_4 = x_1 \wedge x_2$ becomes $C_7 = (\neg x_4 \vee x_1)$ and $C_8 = (\neg x_4 \vee x_2)$ and $C_9 = (x_4 \vee \neg x_1 \vee \neg x_2)$
6. What's missing in our reduction?

Independent Set

We know IS $\in$ NP. Now we want to show that 3-SAT $\leq_p$ IS. Turn an arbitrary 3-SAT instance into an IS problem.

$(\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$

1. Add 3 nodes in a triangle for each clause.
2. What's missing in our reduction?
3. How many problems have we shown are NP-Complete?

Remember, this example is only an illustration to clarify my proof. My proof must work on all 3-SAT instances, not just this one!

There are a lot of NP-complete problems. This is why we highly suspect that P $\neq$ NP. All we would have to do to prove P=NP is come up with a polynomial time algorithm for one of these problems. With how much effort has been expended, we probably would have done so by now if it were possible. Proving that no polynomial algorithm exists for a problem is much much harder.

Set Packing

Given n elements $U = \{u_1, u_2, ..., u_n\}$, m subsets $S_1, S_2, ..., S_m \subseteq U$, and an integer k. Are there k sets which don't intersect?

Think about the similarities between the problems. In Independent Set we are packing as many nodes as we can such that no edge is represented twice. In Set Packing we are packing as many subsets as we can such that no element is represented twice.

k-Clique

Given a graph and an integer k, is there a set S of $\geq k$ nodes such that every pair of nodes in S have an edge between them?

There are a lot of NP-complete problems. This is why we highly suspect that P $\neq$ NP. All we would have to do to prove P=NP is come up with a polynomial time algorithm for one of these problems. With how much effort has been expended, we probably would have done so by now if it were possible. Proving that no polynomial algorithm exists for a problem is much much harder.

Extra Practice:

- Chapter 8: exercises 5, 6, 7, 8, 14, 17, 26, 29
- Challenge problems: Chapter 8, exercises 12, 28, 36