CSE 347 Analysis of Algorithms February 27, 2019

Lecture 12 Summary

Instructor: Brendan Juba

1. NP is the set of verifiable decision problems
   (a) Precisely, a decision problem X is *"in* NP*"* if there exists a *certifier* algorithm B that runs in polynomial time, with the following behavior. X outputs "yes" on x if and only if there exists a c of size polynomial in the size of x such that B outputs "yes" on the pair (x, c).
   (b) Independent Set is in NP since for $x = (G, k)$ B can interpret c as describing $S \subseteq V(G)$ and can check if for every pair $u, v \in S$ $(u, v) \notin E(G)$ (and that $|S| \geq k$). This is indeed polynomial time, and since it returns "yes" if and only if S is an independent set of size at least k, such a c exists if and only if (G, k) is a "yes" instance for Independent Set.
   (c) Vertex Cover is in NP since for $x = (G, k)$, B can interpret c as describing $S \subseteq V(G)$ and can check if for every $(u, v) \in E(G)$, either $u \in S$ or $v \in S$, and that $|S| \leq k$.
   (d) 3-SAT is in NP since for a 3CNF formula x, B can interpret c as an assignment to the variables of x, and can scan the clauses of x and check that at least one of the literals is set to true by c.
   (e) NP contains many, many other problems like integer factoring and encodings of most engineering problems.
2. The "P vs NP" question is equivalent to whether or not "NP-complete" problems have polynomial-time algorithms
   (a) The "P vs NP" question is: do all problems in NP have polynomial-time algorithms?
   (b) A problem X is NP-*complete* if it is in NP and every other Y in NP reduces to X.
   (c) So, either X does not have a polynomial-time algorithm (and so the answer to the "P vs NP" question is "no") or else every problem in NP has a polynomial-time algorithm, and so the answer is "yes"
3. There exist many natural NP-complete problems.
   (a) *Circuit-SAT* is the problem: given a Boolean (AND/OR/NOT) circuit C with a single output wire, is there an input to C that makes it output 1?
   (b) *Cook-Levin Theorem*: Circuit-SAT is NP-complete.
      i. Circuit-SAT is in NP since we can interpret c as a setting of the input wires and evaluate C on c, and return "yes" if it outputs 1.
      ii. For any other Y in NP, we have the following reduction. On input y, we can take the polynomial-time certifier algorithm B for Y (running in $p(n)$ steps) and in polynomial time output a circuit that runs B for $p(|y|)$ steps on $(y, \cdot)$, i.e., with y fixed and the second input an input to the circuit. We run Circuit-SAT on this circuit and return its answer.

iii. Circuit-SAT returns "yes" if and only if there is a setting of the second input c that makes $B(y, c) = 1$. By definition of a certifier, this is true if and only if y is a "yes" for Y, so the reduction is correct.

(c) Lemma: if Y is NP-complete, X is in NP, and $Y \leq_P X$, then X is also NP-complete.

(d) We will show (next time) that 3-SAT is NP-complete.

(e) Since we previously showed 3-SAT reduces to Independent Set (which is in NP), Independent Set is also NP-complete. Furthermore, since we showed Independent Set reduces to Vertex Cover (which is also in NP), Vertex Cover is also NP-complete.