NP-Hardness and Reductions

Monday, October 21, 2013
8:46 PM

A (decision) problem is in NP if it has a polynomially checkable proof (also called a certificate).

E.g (1) HAM (Hamiltonian circuit): Does a graph have a cycle containing every vertex exactly once? Certificate: The Hamiltonian cycle

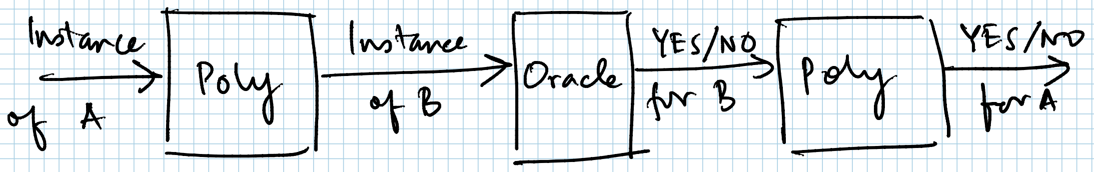
(2) Coloring: Min # of colors where each color class is an independent set

Certificate: The coloring.

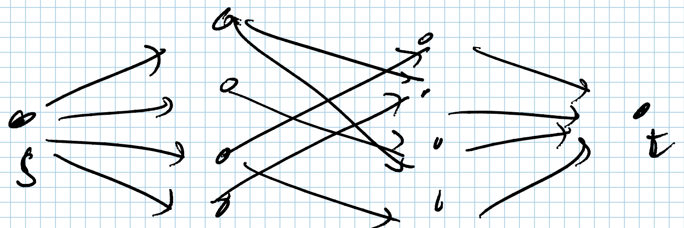
Fact: $P \subseteq NP$

Proof: Run poly time algo to verify solution.

A problem A is said to be polynomially reduce to B (denoted $A \le B$) if given an instance of problem A, we can produce an instance of problem B s.t. there is a polynomial time algorithm that can decide the instance given a decision on the instance of B.



E.g. Bipartite Matching $\le$ Max Flow



Maxflow $\ge k$

$\Updownarrow$

Matching $\ge k$

We will use reductions to establish hardness. If A reduces to B ($A \le B$), then B is at least as hard as A (upto a polynomial).

A problem is said to be NP-hard if all problems in NP reduce to the problem and NP-complete if it additionally belongs to NP.

Theorem (Cook-Levin): SAT is NP-complete.

SAT: $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_4 \vee \neg x_1)$

Is this formula (in CNF) satisfiable ?

Reductions

If $A \leq B$ and A is NP-hard, then so too is B.

Examples:

(1) SAT $\leq$ 3-SAT

$(l_1 \vee l_2 \vee l_3 \vee \dots \vee l_k)$ is isfied by an assignment iff there exists a setting of variables $x_1, x_2, \dots$ such that

$(l_1 \vee l_2 \vee x_1) \wedge (\neg x_1 \vee l_3 \vee x_2) \wedge (\neg x_2 \vee l_4 \vee x_3) \wedge \dots \wedge (\neg x_{k-3} \vee l_{k-1} \vee l_k)$

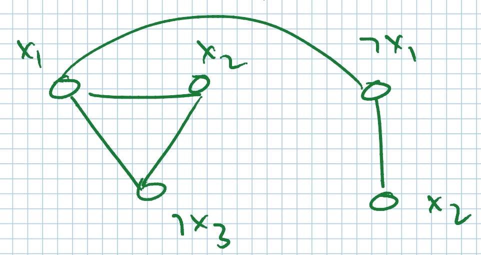
is satisfied.

(2) SAT → Integer programming

$(x_1 \vee x_2 \vee \neg x_3) \rightarrow x_1 + x_2 + (1 - x_3) \geq 1$

(3) SAT → Independent set

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



Independent set $\geq m$ (# of clauses)

$\Updownarrow$

SAT formula is satisfiable

(4) Independent set → Clique

G has independent set of size k iff $\bar{G}$ has clique of size k

(5) Independent set → Vertex cover

G has independent set of size k iff G has vertex cover of size $|V| - k$

(C is an independent set iff $V - C$ is a vertex cover)

(6) Vertex cover → Dominating set

Place a vertex on every edge; size of dominating set in new graph equals the size of a vertex cover on original graph

Approximation Algorithms

An α-approximation algorithm for a minimization (resp. maximization)

An α-approximation algorithm for a minimization (resp. maximization) problem is guaranteed to produce a solution ALGO of value $\leq \alpha \cdot OPT$ (resp. $\geq \frac{OPT}{\alpha}$).

Examples:

(1) Vertex cover: pick both ends of an edge, remove all incident edges, and repeat. 2-approx (best known!!)

(2) Set cover (generalizes vertex cover, hence NP-hard):

sets $S \subseteq U$. Find min. collection of sets that covers all U.

Greedy Algo generalizing VC algo: pick set with max new elements covered

Analysis: Suppose k elements left to be covered. Then cost per element is at most OPT/k.

$\Rightarrow H_n$-approx $= O(\log n)$-approx.

(3) Max coverage: given a budget of k sets, how many elements can we cover?

Greedy algo: pick set that maximizes # of new elements covered

Analysis: If $OPT - ALGO = t_i$ currently, then $\exists$ a set that covers t_i/k elements, ie, $t_{i+1} \leq t_i\left(1 - \frac{1}{k}\right)$

Thus, $t_k \leq t_{k-1}\left(1-\frac{1}{k}\right) \leq \dots \leq t_0\left(1-\frac{1}{k}\right)^k = OPT\left(1-\frac{1}{k}\right)^k$

As $k \to \infty$, $t_k \leq \frac{1}{e} OPT$

$\Rightarrow \left(1-\frac{1}{e}\right)$ approx.

(4) Metric TSP: Walk on spanning tree so that each edge is traversed at most twice — 2-approx