# 15-251: Great Theoretical Ideas In Computer Science

## Recitation 7 Solutions

## Integer Set Reductions

Definition reminder: **Reducing** problem A to problem B means giving an algorithm such that if we can solve problem B in polynomial time, we can solve problem A in polynomial time using that oracle for problem B.

First, we'll define some problems.

**PARTITION** : Given a set $S$ of integers, can we split that set into two subsets $P$, $Q$ with the same sum?

**SUBSETSUM** : Given a set $S$ of integers and an integer $k$, is there a subset of $S$ with sum $k$?

(a) Reduce PARTITION to SUBSETSUM

> Given a query PARTITION$(S)$, take the sum of that set $m$. Then call SUBSETSUM$(S, \frac{m}{2})$. If that returns True, we return True for PARTITION$(S)$ (if there's a subset of $S$ of sum $\frac{m}{2}$, the remaining elements of $S$ will also have sum $\frac{m}{2}$ and those two subsets partition $S$), otherwise we return False.

(b) Reduce SUBSETSUM to PARTITION

> Given a query SUBSETSUM$(S, k)$, first take the sum of $S$ $m$. Then define $S' = S \cup \{2k - m\}$; notice that the sum of $S'$ is $2k$. Now, call PARTITION$(S')$. If it returns True, there must be a partition of $S'$ into two subsets, each with sum $k$. Only one of those subsets can contain the new element we added, so one of those subsets of $S'$ must also be a subset of $S$ with sum $k$ and we return True. Otherwise, we return False.

## MOAR REDUCTIONS

(a) Recall the problem $k$-COLORING which takes a graph and asks whether that graph can be $k$-colored or not. Reduce 2-COLORING to 3-COLORING.

> Suppose we have a 3-COLORING oracle. Take any graph that you want to determine if it is 2-COLORABLE, and add a new vertex $x$. Place an edge between $x$ and every other vertex in the graph. If we can three color the new graph, $x$ must have a unique color so the rest of the graph must be properly two colored. If we could've two colored the original graph, giving $x$ the third color would three color the new graph. So the solution to 3-COLORING matches the desired solution to 2-COLORING.

(b) As you know, college students must occasionally go to class. Imagine we have the set of classes $C = \{c_1, \ldots, c_n\}$, and each student $s_1, \ldots, s_m$ has a set $S_i \subseteq C$ of classes that they are enrolled in. Clearly, a student can't be in two places at once — so if they're enrolled in two classes that meet at the same time, then we say their schedule is impossible. We would like to know if it is possible to fit all the classes into $k$ different time slots such that no student has an impossible schedule.

Prove 3-COLORING reduces to this scheduling problem.

Suppose we have an oracle $\mathcal{O}$ for the scheduling problem, and we are given a graph $G = (V, E)$ and want to know if it's 3-colorable. We'll translate this into an instance of the scheduling problem. We'll simply have one class for each vertex, so $C = V$. For each edge $uv$ in the graph, add a student $s_{uv}$ who wants to take classes $c_u$ and $c_v$. Now, feed this schedule to the oracle with $k = 3$, and output its answer.
We argue the created scheduling instance is possible iff $G$ is 3-colorable. First, if it's possible, we have some schedule where no student is enrolled in two classes set for the same time. That is, we have a function $f : C \to [3]$ such that for each student $s_{uv}$, $f(u) \neq f(v)$. Since we added a student for each edge, this means that for each edge $uv$, $f(u) \neq f(v)$, so $f$ is a proper 3-coloring of $G$.
Conversely, if we have a proper 3-coloring, $f : V \to [3]$, then for any student $s_{uv}$, $f(u) \neq f(v)$, so no student is enrolled in two classes in the same time slot.

(c) Reduce HAMILTONIAN (given a graph $G$, does $G$ have a Hamiltonian cycle?) to the Travelling Salesman Problem.

Given a graph $G$, define a new graph $G'$ that has all the vertices of $G$, and has all edges, with weight 1 if that edge exists in $G$ and weight 2 if that edge doesn't exist in $G$. Then call our Traveling Salesman oracle - if the minimum cost trip has cost $n$, then there is a Hamiltonian cycle in $G$. Otherwise, there is not.

## If CHARIZARD is NP-Hard, so are VENUSAUR and BLASTOISE

Consider a graph $G = (V, E)$. Let CHARIZARD be the problem of placing as many Charizards as possible on the vertices graph, knowing that any Charizard will burn the vertex it occupies as well as all neighboring vertices, such that no vertex gets burned twice. (Formally, you want the largest set $C \subseteq V$ s.t. $d(x, y) \geq 3$ for all $x \neq y \in C$)

(a) Give a reduction from CHARIZARD to CLIQUE

Note that saying $d_G(a, b) \geq 3$ is the same as saying that for all $x \in V$, $\{a, x\}$ or $\{b, x\}$ is an edge in $\overline{G}$ (otherwise, $\{a, x\}$ and $\{b, x\}$ are edges in $G$ and $d(a, b) = 2$) Note that $\{a, a\}$ can't be an edge, so this correctly implies $\{a, b\}$ is an edge in $\overline{G}$.

Construct $G'$ as follows: Use the same vertex set. For each $a$ and $b$ in $V$ let $\{a, b\}$ be an edge if and only if for every $x \in V$ $\{a, x\}$ or $\{b, x\}$ (or both) are edges in $\overline{G}$. Thus $\{a, b\} \in E_{G'}$ iff $d_G(a, b) \geq 3$. Note that this construction only takes $n^3$ time. Also note that any legal placing of Charizards will create a clique in $G'$ (as every vertex in the Charizard set must have distance three or more). Thus we can use the CLIQUE oracle to find a legal Charizard set and solve CHARIZARD

(b) Give a reduction from CLIQUE to CHARIZARD

Instead let's show how to quickly solve INDEPENDENT SET, as we already know how to quickly solve CLIQUE if we can do this.
Let $G$ be the graph we want to solve INDEPENDENT SET on.
Start with $G'$ being the same as $G$. Now for each edge $\{a, b\}$ in $E(G)$, remove said edge from $E(G')$ and add vertex $ab$ as well as edges $\{a, ab\}$ and $\{b, ab\}$. Also place all added vertices into a big clique.
Suppose $d_G(a, b) \geq 2$. Then along that same path, $d_G(a, b)$ must be four (as we split each edge into two parts). So any path from $a$ to $b$ shorter than four must use one of the edges in the clique. However, we must first go from $a$ to some $x$, then to some vertex in the clique $y$, then to $b$ which makes $d_{G'}(a, b) \geq 3$.
Let $a$ and $b$ be vertices from $V(G)$. Suppose $d_{G'}(a, b) \geq 3$. Then $(a, b)$ weren't adjacent in $G$ (otherwise $d_{G'}(a, b) = 2$. If $a, b \in V(G')$ are both in the added clique, $d_{G'}(a, b) = 1$. So the only case to consider is $a, b \in V(G')$ s.t. $a \in V(G)$ but $b$ is an added vertex. If there is a $c$ in $V(G)$ s.t. $\{a, c\} \in E(G)$ then the vertex $ac$ is in the clique, and so $d_{G'}(a, b) = 2$. Thus in order for $d_{G'}(a, b)$ to be $\geq 3$, $a$ must be disconnected from the rest of the graph. Thus, if $b$ is in the CHARIZARD-set, there can be no other vertex in $b$'s compontent in the CHARIZARD set. This means there is an equally valid CHARIZARD set using a vertex from $V(G)$ (namely one of $b$'s neighbors not in the clique).

So we have shown that $d_{G'}(a, b) \geq 2$ implies $d_G(a, b) \geq 3$ and vice versa, so if we solve CHARIZARD for $G'$ we solve INDEPENDENT SET (and thus CLIQUE) for $G$.

Let CHARIZARD-k be the problem of whether or not we can fit $k$ CHARIZARD's on the graph under the "distance at least three" constraint.

(c) Give a reduction from CHARIZARD to CHARIZARD-k (so we have an oracle that can answer the question "On graph $G$, can we place at least $k$ Charizards?" for any $G, k$ and want to know the most Charizards we can place)

We know the size of the CHARIZARD set is at most $n$, the total number of vertices. So perform a binary search. Start with the CHARIZARD-$\frac{n}{2}$ oracle. If it returns FALSE we know we can't fit that many Charizard, so try again with CHARIZARD-$\frac{n}{4}$. If it returns TRUE, we know we can fit that many Charizard, so try again with CHARIZARD-$\frac{3n}{4}$. Repeating this process will give us a solution after $\log_2 n$ calls to CHARIZARD-k

(d) Give a reduction from CHARIZARD-k to CHARIZARD

One running of CHARIZARD gives us more information. It tells us the maximal number of Charizard that fit, thus we can simply compare that result to $k$. If it is smaller than $k$ we know we can't fit $k$ Charizard, otherwise we know we can.