Reductions

We want to compare the complexity of different problems.

A reduction from problem X to problem Y means that problem X is easier (or, more precisely, not harder) than problem Y. We write

$$X \leq Y$$

A reduction from X to Y means that if we have an algorithm for Y, we can use it to find an algorithm for X.

So we can use reductions to find algorithms.

But we can also use reductions to show that we cannot find algorithms for some problems. Such problems are called hard.

Also: Find the right reduction and win a million dollars!

Example 0

How do we solve $\textsc{IntervalScheduling}$?
(Given a set of intervals and a number $k > 0$, is there a non-overlapping set of intervals of size at least k?)

Solution:
Reduce it to $\textsc{WeightedIntervalScheduling}$.

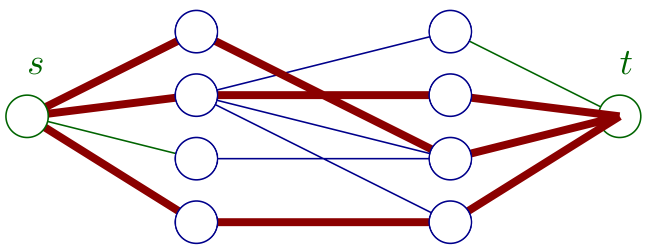
Give every interval weight one. There is a subset of intervals of size $\geq k$ iff there is a subset of intervals of weight $\geq k$.

And so we showed:

$$\textsc{IntervalScheduling} \leq \textsc{WeightedIntervalScheduling}$$

Example 1

How do we solve $\textsc{BipartiteMatching}$?
Given a bipartite graph $G = (U \cup V, E)$ and a number $k > 0$, does G have a matching of size $\geq k$?



Solution:
Reduce it to $\textsc{MaxFlow}$. G has a matching of size $\geq k$ iff there is a flow from s to t of value $\geq k$.

$$\textsc{BipartiteMatching} \leq \textsc{MaxFlow}$$

What is a reduction?

We work with decision problems

For decision problems X and Y, a reduction from X to Y is:

- an algorithm
- that takes an instance I_X of X as input,
- and returns an instance I_Y of Y as output,
- such that the solution (that is, yes or no) of I_Y is the same as the solution of I_X.

(Actually this is only one type of reduction, but this is the one we will use mostly.)

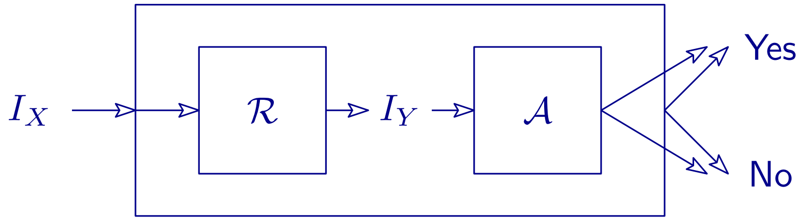
Using reductions to solve problems

Given a reduction $\mathcal{R}$ from X to Y and an algorithm $\mathcal{A}$ for Y:
We have an algorithm for X!

What we need

Using reductions to solve problems

Given a reduction $\mathcal{R}$ from X to Y and an algorithm $\mathcal{A}$ for Y:
We have an algorithm for X!



What we have!

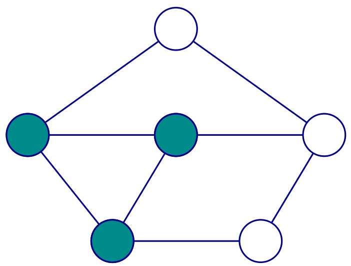
If $\mathcal{R}$ and $\mathcal{A}$ run in polynomial time, then the resulting algorithm for X is also a polynomial-time algorithm.

We write $X \leq Y$ iff there is a polynomial-time reduction from X to Y.

Independent Sets and Cliques

Given a graph $G = (V, E)$, a subset $S \subseteq V$ is:

- an independent set if no two vertices of S are connected by an edge.
- a clique if every pair of vertices in S is connected by an edge.



IndependentSet and Clique

IndependentSet:
Instance: A graph G and an integer k.
Question: Does G have an independent set of size $\geq k$?

Clique:
Instance: A graph G and an integer k.
Question: Does G have a clique of size $\geq k$?

We want to show

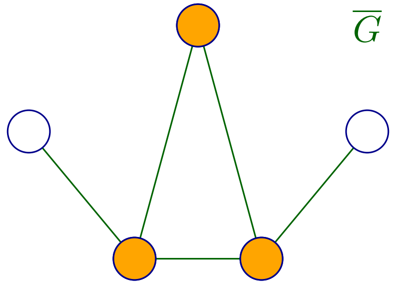
$$\text{IndependentSet} \leq \text{Clique}$$

The reduction needs to convert an instance of IndependentSet to an instance of Clique.

$$(\text{Graph } G, \text{ integer } k) \Longrightarrow (\text{Graph } G', \text{ integer } k')$$

The reduction

We set $G' = \overline{G}$, the complement of G, and $k' = k$.



Lemma: S is an independent set of G iff S is a clique of $\overline{G}$.

So the solution to $I_Y = (\overline{G}, k)$ is the same as the solution to $I_X = (G, k)$.

Reductions and hardness

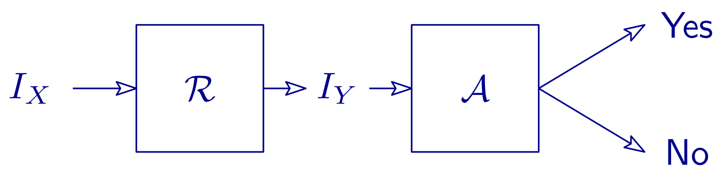
Recall: Efficient algorithms are polynomial-time algorithms

Lemma: If $X \leq Y$ and Y has an efficient algorithm, then X has an efficient algorithm.

- We believe INDEPENDENTSET has no efficient algorithm.
- We have INDEPENDENTSET $\leq$ CLIQUE.
- If CLIQUE had an efficient algorithm, so would INDEPENDENTSET!

Lemma: If $X \leq Y$ and X does not have an efficient algorithm, then Y cannot have an efficient algorithm.

Instance size



Running time of $\mathcal{R}$ is $p(|I_X|)$, for a polynomial p.

Running time of $\mathcal{A}$ is $q(|I_Y|)$, for a polynomial q.

What is $|I_Y|$?

Theorem: If $\mathcal{R}$ is a polynomial-time reduction, then the size of I_Y produced from I_X is polynomial in the size of I_X.

Proof: $\mathcal{R}$ can write at most $p(|I_X|)$ bits, and so $|I_Y| \leq p(|I_X|)$.

Polynomial-time reduction

A polynomial-time reduction (Karp reduction) from X to Y is an algorithm $\mathcal{R}$ such that:

- Given an instance I_X of X, $\mathcal{R}(I_X)$ is an instance I_Y of Y.
- $\mathcal{R}$ runs in time polynomial in $|I_X|$. This implies that $|I_Y|$ is polynomial in $|I_X|$.
- The answer to I_X is yes iff the answer to I_Y is yes.

Theorem: If $X \leq Y$ then a polynomial-time algorithm for Y implies a polynomial-time algorithm for X.

Theorem: Reductions are transitive. $X \leq Y$ and $Y \leq Z$ implies $X \leq Z$.

Important: $X \leq Y$ does not imply $Y \leq X$. Distinguish "from" and "to" in a reduction.

VertexCover

Instance: A graph $G = (V, E)$ and an integer $k > 0$.
Question: Does G have a vertex cover S of size $\leq k$?
($S \subset V$ is a vertex cover if every $e \in E$ has at least one endpoint in S.)

Theorem: $S \subset V$ is a vertex cover iff $V \setminus S$ is an independent set.

Therefore: G has independent set of size $\geq k$ iff G has vertex cover of size $\leq n - k$.

Reduction IndependentSet $\leq$ VertexCover:

If (G, k) is an instance of IndependentSet, then $(G, n - k)$ is an instance of VertexCover with the same answer.

SetCover

The SetCover problem:
Instance: A set U of n elements, a collection $S_1, S_2, \ldots, S_m$ of subsets of U, and an integer $k > 0$.
Question: Is there a collection of at most k of these sets S_i whose union is equal to U?

Let $U = \{1, 2, 3, 4, 5, 6, 7\}$, $k = 2$ with

$$\begin{array}{ll} S_1 = \{3, 7\} & S_2 = \{3, 4, 5\} \\ S_3 = \{1\} & S_4 = \{2, 4\} \\ S_5 = \{5\} & S_6 = \{1, 2, 6, 7\} \end{array}$$

VertexCover $\leq$ SetCover

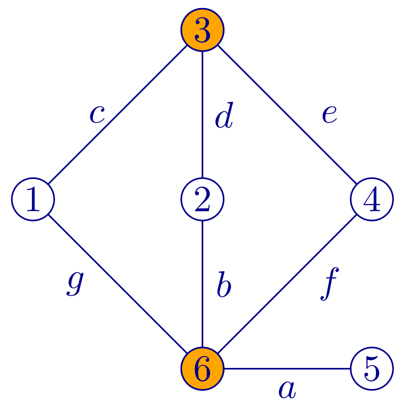
Given a VertexCover instance (G, k) we construct a SetCover instance $(U, \{S_1, \ldots, S_m\}, k')$.

- $k' = k$
- $U = E$
- For each $v \in V$, we have one set $S_v = \{e \mid e \text{ is incident on } v\}$.

The reduction can be computed in polynomial time.

G has a vertex cover of size k if and only if $U, \{S_v\}_{v \in V}$ has a set cover of size k.

Example



We have proven the reductions:

$$\textsc{IndependentSet} \le \textsc{VertexCover} \le \textsc{SetCover}$$

$$\textsc{IndependentSet} \le \textsc{Clique}$$