Polynomial-time Reductions

Disclaimer: Many definitions in these slides should be taken as "the intuitive meaning", as the precise meaning of some of the terms are hard to pin down without introducing the formal machinery of computational models (the Turing machine in particular).

- A problem is a *decision problem* if its answer is either yes or no.

Polynomial-time Reductions

Disclaimer: Many definitions in these slides should be taken as "the intuitive meaning", as the precise meaning of some of the terms are hard to pin down without introducing the formal machinery of computational models (the Turing machine in particular).

- A problem is a *decision problem* if its answer is either yes or no.
- A decision problem A is *polynomial-time reducible* to a decision problem B if there is a polynomial-time algorithm φ that takes any instance a of problem A and returns an instance of problem B, such that $\varphi(a)$ has the same answer as a; that is, the answer to $\varphi(a)$ is yes if and only if the answer to a is yes.
  - We denote this by $A \leq_{\mathrm{P}} B$.

Polynomial time reduction example

Definition

Given a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is an *independent set* if no two nodes in S are connected by an edge.

Polynomial time reduction example

Definition

Given a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is an *independent set* if no two nodes in S are connected by an edge.

Definition

In the *independent set* problem, we are given a graph $G = (V, E)$ and an integer k. We need to answer whether there exists an independent set of G of size at least k.

Polynomial time reduction example

Definition

Given a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is an *independent set* if no two nodes in S are connected by an edge.

Definition

In the *independent set* problem, we are given a graph $G = (V, E)$ and an integer k. We need to answer whether there exists an independent set of G of size at least k.

Recall: Given a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is a *vertex cover* if every edge is incident to at least one node in S.

Polynomial time reduction example

Definition

Given a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is an *independent set* if no two nodes in S are connected by an edge.

Definition

In the *independent set* problem, we are given a graph $G = (V, E)$ and an integer k. We need to answer whether there exists an independent set of G of size at least k.

Recall: Given a graph $G = (V, E)$, a set of nodes $S \subseteq V$ is a *vertex cover* if every edge is incident to at least one node in S.

Definition

In the *vertex cover problem*, we are given a graph $G = (V, E)$ and an integer k. We need to answer whether there exists a vertex cover of size at most k.

The classes P and NP

- The class P is the set of all decision problems that can be solved in polynomial time.

The classes P and NP

- The class P is the set of all decision problems that can be solved in polynomial time.
- The class NP (standing for *nondeterministic polynomial time*) is the set of decision problems for which a *polynomial-time verifier* algorithm V exists for the following the task: if the answer to an instance of the problem is yes, then there exists a *polynomial-length certificate*, such that V, when provided with both the instance and the certificate, will return yes; on the other hand, if the answer to an instance of the problem is no, then V returns no, no matter what certificate it is given.

The classes P and NP

- The class P is the set of all decision problems that can be solved in polynomial time.
- The class NP (standing for *nondeterministic polynomial time*) is the set of decision problems for which a *polynomial-time verifier* algorithm V exists for the following the task: if the answer to an instance of the problem is yes, then there exists a *polynomial-length certificate*, such that V, when provided with both the instance and the certificate, will return yes; on the other hand, if the answer to an instance of the problem is no, then V returns no, no matter what certificate it is given.
- Obviously, $\mathrm{P} \subseteq \mathrm{NP}$.

The classes P and NP

- The class P is the set of all decision problems that can be solved in polynomial time.
- The class NP (standing for *nondeterministic polynomial time*) is the set of decision problems for which a *polynomial-time verifier* algorithm V exists for the following the task: if the answer to an instance of the problem is yes, then there exists a *polynomial-length certificate*, such that V, when provided with both the instance and the certificate, will return yes; on the other hand, if the answer to an instance of the problem is no, then V returns no, no matter what certificate it is given.
- Obviously, $\mathrm{P} \subseteq \mathrm{NP}$.
- The most famous question in (theoretical) computer science: $\mathrm{NP} \subseteq \mathrm{P}$?

NP Completeness

- A problem A in a class $\mathcal{C}$ of problems is said to be $\mathcal{C}$-*complete* if all problems in $\mathcal{C}$ be reduced to A by an "appropriate" reduction.

NP Completeness

- A problem A in a class $\mathcal{C}$ of problems is said to be $\mathcal{C}$-*complete* if all problems in $\mathcal{C}$ be reduced to A by an "appropriate" reduction.
- For class NP, the "appropriate" reductions are the polynomial-time reductions. A problem is NP-complete if it is in NP and if all other problems in NP can be reduced to it in polynomial time.

NP Completeness

- A problem A in a class $\mathcal{C}$ of problems is said to be $\mathcal{C}$-*complete* if all problems in $\mathcal{C}$ be reduced to A by an "appropriate" reduction.
- For class NP, the "appropriate" reductions are the polynomial-time reductions. A problem is NP-complete if it is in NP and if all other problems in NP can be reduced to it in polynomial time.

Theorem (Cook-Levin)

SAT is NP*-complete.*