Learning Goals

- Define polynomial-time reductions.
- Understand consequences of polynomial-time reductions in terms of tractability of problems.
- Define decision problems and to use decision problems to solve optimization problems.
- Define independent sets, vertex coversi.
- State the decision problems INDEPENDENT SET, VERTEX COVER and SET COVER
- Understand the reductions between the three problems.

Polynomial-time Reductions

- Recall applications of the max flow min cut algorithm

Polynomial-time Reductions

- Recall applications of the max flow min cut algorithm
- *Black box* oracle access to an algorithm for a problem X: given an instance of problem X, the oracle returns correctly a solution to the instance in a single step.

Polynomial-time Reductions

- Recall applications of the max flow min cut algorithm
- *Black box* oracle access to an algorithm for a problem X: given an instance of problem X, the oracle returns correctly a solution to the instance in a single step.

Definition

A *polynomial-time reduction* from a problem Y to a problem X is an algorithm that solves any instance of Y by taking polynomially many standard computational steps plus a polynomial number of calls to a black box that solves problem X.

Polynomial-time Reductions

- Recall applications of the max flow min cut algorithm
- *Black box* oracle access to an algorithm for a problem X: given an instance of problem X, the oracle returns correctly a solution to the instance in a single step.

Definition

A *polynomial-time reduction* from a problem Y to a problem X is an algorithm that solves any instance of Y by taking polynomially many standard computational steps plus a polynomial number of calls to a black box that solves problem X.
When such a reduction exists, we say Y is polynomial-time reducible to X, denoted as $Y \leq_P X$.

Polynomial-time Reductions

- Recall applications of the max flow min cut algorithm
- *Black box* oracle access to an algorithm for a problem X: given an instance of problem X, the oracle returns correctly a solution to the instance in a single step.

Definition

A *polynomial-time reduction* from a problem Y to a problem X is an algorithm that solves any instance of Y by taking polynomially many standard computational steps plus a polynomial number of calls to a black box that solves problem X.
When such a reduction exists, we say Y is polynomial-time reducible to X, denoted as $Y \leq_P X$.

Example: Image Segmentation $\leq_P$ min cut

Consequences of polynomial-time reductions

Proposition

If $Y \leq_P X$, then

1. If X can be solved in polynomial time, then Y can be as well;
2. If Y cannot be solved in polynomial time, then X cannot be either.

Consequences of polynomial-time reductions

Proposition

If $Y \leq_P X$, then

1. If X can be solved in polynomial time, then Y can be as well;
2. If Y cannot be solved in polynomial time, then X cannot be either.

Proof.

1. Take a polynomial-time reduction from Y to X, replace each call of the black box oracle to X by the polynomially many steps needed to solve X. The resulting algorithm runs in polynomial time.

Consequences of polynomial-time reductions

Proposition

If $Y \leq_P X$, then

1. If X can be solved in polynomial time, then Y can be as well;
2. If Y cannot be solved in polynomial time, then X cannot be either.

Proof.

1. Take a polynomial-time reduction from Y to X, replace each call of the black box oracle to X by the polynomially many steps needed to solve X. The resulting algorithm runs in polynomial time.
2. This is the contrapositive of the first bullet.

$\square$

Consequences of polynomial-time reductions

Proposition

If $Y \leq_P X$, then

1. If X can be solved in polynomial time, then Y can be as well;
2. If Y cannot be solved in polynomial time, then X cannot be either.

Proof.

1. Take a polynomial-time reduction from Y to X, replace each call of the black box oracle to X by the polynomially many steps needed to solve X. The resulting algorithm runs in polynomial time.
2. This is the contrapositive of the first bullet.

$\square$

Example: Since Min Cut can be solved in polynomial time, so is Image Segmentation.

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

- Example: Given a flow network G and an integer K, decide whether there is a flow in G whose value is at least K.

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

- Example: Given a flow network G and an integer K, decide whether there is a flow in G whose value is at least K.
- Generally, there are decision versions of optimization problems.
    - Optimization problems ask for maximization or minimization of certain objectives, often subject to constraints.

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

- Example: Given a flow network G and an integer K, decide whether there is a flow in G whose value is at least K.
- Generally, there are decision versions of optimization problems.
  - Optimization problems ask for maximization or minimization of certain objectives, often subject to constraints.
- Generally, optimization problems can be poly-time reduced to their decision versions

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

- Example: Given a flow network G and an integer K, decide whether there is a flow in G whose value is at least K.
- Generally, there are decision versions of optimization problems.
  - Optimization problems ask for maximization or minimization of certain objectives, often subject to constraints.
- Generally, optimization problems can be poly-time reduced to their decision versions via binary search.

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

- Example: Given a flow network G and an integer K, decide whether there is a flow in G whose value is at least K.
- Generally, there are decision versions of optimization problems.
  - Optimization problems ask for maximization or minimization of certain objectives, often subject to constraints.
- Generally, optimization problems can be poly-time reduced to their decision versions via binary search.

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

- Example: Given a flow network G and an integer K, decide whether there is a flow in G whose value is at least K.
- Generally, there are decision versions of optimization problems.
    - Optimization problems ask for maximization or minimization of certain objectives, often subject to constraints.
- Generally, optimization problems can be poly-time reduced to their decision versions via binary search.
    - For maximization problems, there are often a natural upper bound U and a natural lower bound L;

Decision Problems

Definition

A problem is a *decision problem* if its answer is either TRUE or FALSE.

- Example: Given a flow network G and an integer K, decide whether there is a flow in G whose value is at least K.
- Generally, there are decision versions of optimization problems.
  - Optimization problems ask for maximization or minimization of certain objectives, often subject to constraints.
- Generally, optimization problems can be poly-time reduced to their decision versions via binary search.
  - For maximization problems, there are often a natural upper bound U and a natural lower bound L;
  - Use a black box oracle to the decision version allows one to perform binary search between L and U.

Karp reductions

Definition

Given two decision problems A and B, a *Karp reduction* is a polynomial-time algorithm φ with

- Input: an instance a of A
- Output: an instance $\varphi(a)$ of B
- Guarantee: the answer to a is TRUE $\Leftrightarrow$ the answer to $\varphi(a)$ is TRUE.

Karp reductions

Definition

Given two decision problems A and B, a *Karp reduction* is a polynomial-time algorithm φ with

- Input: an instance a of A
- Output: an instance $\varphi(a)$ of B
- Guarantee: the answer to a is TRUE $\Leftrightarrow$ the answer to $\varphi(a)$ is TRUE.

Remark

- A Karp reduction is a polynomial-time reduction.
  - A general polynomial-time reduction is sometimes referred to as a *Cook* reduction or *Turing* reduction.

Karp reductions

Definition

Given two decision problems A and B, a *Karp reduction* is a polynomial-time algorithm φ with

- Input: an instance a of A
- Output: an instance $\varphi(a)$ of B
- Guarantee: the answer to a is TRUE $\Leftrightarrow$ the answer to $\varphi(a)$ is TRUE.

Remark

- A Karp reduction is a polynomial-time reduction.
  - A general polynomial-time reduction is sometimes referred to as a *Cook* reduction or *Turing* reduction.
- A Karp reduction calls the oracle for B only once.

Karp reductions

Definition

Given two decision problems A and B, a *Karp reduction* is a polynomial-time algorithm φ with

- Input: an instance a of A
- Output: an instance $\varphi(a)$ of B
- Guarantee: the answer to a is TRUE $\Leftrightarrow$ the answer to $\varphi(a)$ is TRUE.

Remark

- A Karp reduction is a polynomial-time reduction.
    - A general polynomial-time reduction is sometimes referred to as a *Cook* reduction or *Turing* reduction.
- A Karp reduction calls the oracle for B only once.
- In this class we always do Karp reductions.

Independent Set

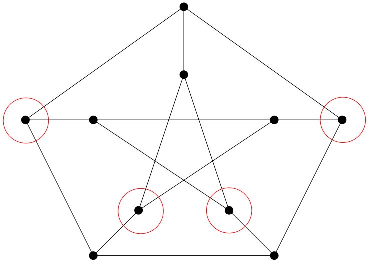
Definition

Given an undirected 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.

Independent Set

Definition

Given an undirected 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.



INDEPENDENT SET

Definition

Given an undirected 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

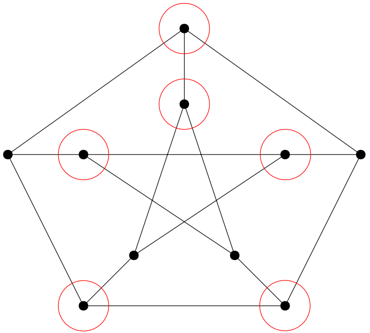
In the *INDEPENDENT SET* problem, we are given an undirected graph $G = (V, E)$ and an integer k. We must answer whether G has an independent set of size at least k.

Vertex Cover

Definition

Given an undirected 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.

Recall Question 4 of Problem Set 2.



VERTEX COVER

Definition

Given an undirected 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 an undirected graph $G = (V, E)$ and an integer k. We must answer whether G has a vertex cover of size at most k.

Polynomial-time reduction example

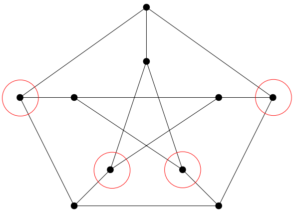
Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

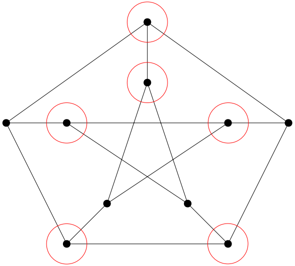
Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.



An independent set



A vertex cover

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Lemma

For any graph $G = (V, E)$, if $S \subseteq V$ is an independent set, then $V - S$ is a vertex cover.

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Lemma

For any graph $G = (V, E)$, if $S \subseteq V$ is an independent set, then $V - S$ is a vertex cover.

Proof.

For any edge $e = (u, v) \in E$, either $u \notin S$ or $v \notin S$ (otherwise S cannot be independent).

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Lemma

For any graph $G = (V, E)$, if $S \subseteq V$ is an independent set, then $V - S$ is a vertex cover.

Proof.

For any edge $e = (u, v) \in E$, either $u \notin S$ or $v \notin S$ (otherwise S cannot be independent).
Therefore $V - S$ covers every edge, i.e., it is a vertex cover. □

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Lemma

For any graph $G = (V, E)$, if $S \subseteq V$ is a vertex cover, then $V - S$ is an independent set.

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Lemma

For any graph $G = (V, E)$, if $S \subseteq V$ is a vertex cover, then $V - S$ is an independent set.

Proof.

For any two vertices $u, v \in V - S$, there cannot be an edge (u, v) (otherwise S is not a vertex cover.

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Lemma

For any graph $G = (V, E)$, if $S \subseteq V$ is a vertex cover, then $V - S$ is an independent set.

Proof.

For any two vertices $u, v \in V - S$, there cannot be an edge (u, v) (otherwise S is not a vertex cover.
Therefore $V - S$ is a vertex cover. $\square$

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Proof.

There is an independenset set of size at least k if and only if there is a vertex cover of size at most $|V| - k$.

Polynomial-time reduction example

Proposition

INDEPENDENT SET $\leq_P$ VERTEX COVER.
VERTEX COVER $\leq_P$ INDEPENDENT SET.

Proof.

There is an independenset set of size at least k if and only if there is a vertex cover of size at most $|V| - k$.
The Karp reduction from INDEPENDENT SET to VERTEX COVER:

1. Input: graph $G = (V, E)$ and integer k.
2. Output (as an instance of VERTEX COVER): same graph G and integer $|V| - k$.

□

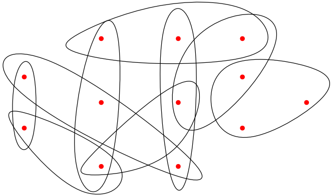
Review of last lecture

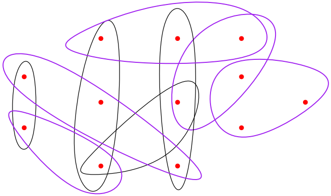
- Definition: Polynomial-time reduction
- If $A \leq_{\mathrm{P}} B$, then B is "harder" than A.
- Definition: Decision problems, Karp reductions
- Definition: Independent set, vertex cover
- INDEPENDENT SET $\leq_{\mathrm{P}}$ VERTEX COVER; VERTEX COVER $\leq_{\mathrm{P}}$ INDEPENDENT SET

SET COVER

Definition

In the *SET COVER* problem, we are given a set U of n elements, a collection $S_1, \cdots, S_m$ of subsets of U, and a number k, and we must answer whether there is a collection of at most k of these sets whose union is equal to U.





SET COVER

Proposition

VERTEX COVER $\leq_P$ SET COVER.

SET COVER

Proposition

VERTEX COVER $\leq_P$ SET COVER.

Proof.

Given a VERTEX COVER problem $G = (V, E)$ and integer k, create the following SET COVER instance:

- $U = E$;
- For every vertex $v \in V$, create a set $S_v \subseteq U$ which is the set of edges incident to v.

SET COVER

Proposition

VERTEX COVER $\leq_P$ SET COVER.

Proof.

Given a VERTEX COVER problem $G = (V, E)$ and integer k, create the following SET COVER instance:

- $U = E$;
- For every vertex $v \in V$, create a set $S_v \subseteq U$ which is the set of edges incident to v.

There is a vertex cover in G of size at most k if and only if there is a set cover of size at most k in the instance we created. $\square$

Last remark

Remark

This is a reduction from a special problem to a more general problem. A VERTEX COVER problem is precisely a SET COVER problem when every element of U is contained in two given subsets.