Today’s Keywords

Reductions
NP-Completeness
P vs. NP

**CLRS Readings:** Chapter 34
Homework

HW8 due Saturday, November 23, 11pm
  • Programming assignment (Python or Java)
  • Graph algorithms

HW9, HW10C out today (due Thursday, December 5)
  • Graphs, Reductions
  • Written (LaTeX)
Final Exam

Monday, December 9, 7pm in Olsson 120

- Practice exam coming next week
- Review session likely the weekend before

Exam conflicts: Sign-up by tomorrow (Friday, November 22)
- Alternative exam only for student with an conflicting exam at the same time
Reductions

Problem \( A \)

Solution for \( A \)

Map instances of problem \( A \) to instances of \( B \)

Map solutions of problem \( B \) to solutions of \( A \)

Problem \( B \)

Algorithm for \( B \)

Solution for \( B \)

\( A \leq B \): there is a reduction from \( A \) to \( B \)
Reduction Examples

Map instances of problem $A$ to instances of $B$

Map solutions of problem $B$ to solutions of $A$

Reduction

Ford-Fulkerson

edge-disjoint paths

max flow
Reduction Examples

vertex-disjoint paths

Edge-disjoint paths

Map instances of problem $A$ to instances of $B$

Map solutions of problem $B$ to solutions of $A$

Reduction

Edge-disjoint paths algorithm
Reduction Examples

maximum bipartite matching

Map instances of problem $A$ to instances of $B$

Map solutions of problem $B$ to solutions of $A$

Reduction

max flow

Ford-Fulkerson
Draw edges between people who do not get along

**Goal:** Find the maximum number of people who get along
Maximum Independent Set

**Independent set:** $S \subseteq V$ is an independent set if no two nodes in $S$ share an edge

**Maximum independent set problem:** Given a graph $G = (V, E)$, find the largest independent set $S$
Maximum Independent Set Example

Independent set of size 6
Need to place defenders on each base so each edge is defended.
Generalized Baseball

Need to place defenders on each base so each edge is defended

Problem: Fewest number of defenders required?
Minimum Vertex Cover

**Vertex cover:** $C \subseteq V$ is a vertex cover if every edge in $E$ has one of its endpoints in $C$

**Minimum vertex cover:** Given a graph $G = (V, E)$, find the smallest vertex cover $C$
Vertex Cover Example

Turns out that problem of finding a minimum vertex cover is closely related to problem of finding a maximum independent set.
Reductions

Problem \( A \)

Solution for \( A \)

\( A \leq B \): there is a reduction from \( A \) to \( B \)

Map instances of problem \( A \) to instances of \( B \)

Map solutions of problem \( B \) to solutions of \( A \)

Problem \( B \)

Algorithm for \( B \)

Solution for \( B \)
Max Independent Set \( \leq \) Min Vertex Cover

\[ A \leq B: \text{there is a reduction from } A \text{ to } B \]
Max Independent Set $\leq$ Min Vertex Cover

**Independent set**: set of nodes that do not share an edge

**Vertex cover**: set of nodes that cover all edges

**Claim**: $S$ is an independent set if and only if its complement $V - S$ is a vertex cover

**Important note**: a maximum independent set may not be a vertex cover
Max Independent Set $\leq$ Min Vertex Cover

**Independent set:** set of nodes that do not share an edge

**Vertex cover:** set of nodes that cover all edges

**Claim:** $S$ is an independent set $\Rightarrow V - S$ is a vertex cover
- Suppose $S$ is an independent set

**Important note:** a maximum independent set may **not** be a vertex cover
Max Independent Set $\leq$ Min Vertex Cover

**Independent set:** set of nodes that do not share an edge

**Vertex cover:** set of nodes that cover all edges

**Claim:** $S$ is an independent set $\Rightarrow V - S$ is a vertex cover
- Suppose $S$ is an independent set
- Take any edge $e = (u, v) \in E$
- Either $u \notin S$ or $v \notin S$ (otherwise, $u, v \in S$, and $S$ is no longer an independent set)
- Either $u \in V - S$ or $v \in V - S$, so $e$ is covered by $V - S$

**Important note:** a maximum independent set may not be a vertex cover
Max Independent Set $\leq$ Min Vertex Cover

**Independent set:** set of nodes that do not share an edge

**Vertex cover:** set of nodes that cover all edges

**Claim:** $V - S$ is a vertex cover $\Rightarrow S$ is an independent set

- Suppose $V - S$ is a vertex cover
- Take any edge $e = (u, v) \in E$
- Since $V - S$ is a vertex cover, at least one of $u \in V - S$ or $v \in V - S$ should hold
- This means either $u \not\in S$ or $v \not\in S$ (or both)
- Thus, there is no edge between any pair of nodes $u, v \in S$

**Important note:** a maximum independent set may **not** be a vertex cover
**Max Independent Set \( \leq \) Min Vertex Cover**

**Independent set:** set of nodes that do not share an edge

**Vertex cover:** set of nodes that cover all edges

**Claim:** \( S \) is an independent set if and only if its complement \( V - S \) is a vertex cover

**Conclusions:**
- There is a one-to-one correspondence between independent sets and vertex covers
- Independent sets and vertex covers are complements so maximizing one means minimizing the other

**Important note:** a maximum independent set may **not** be a vertex cover
Max Independent Set \( \leq \) Min Vertex Cover

Maximum independent set

Minimum vertex cover

Map instances of problem \( A \) to instances of \( B \)

\( O(1) \) time

Map solutions of problem \( B \) to solutions of \( A \)

\( O(|V|) \) time

Reduction
Min Vertex Cover $\leq$ Max Independent Set

Minimum vertex cover

Maximum independent set

Map instances of problem $A$ to instances of $B$ in $O(1)$ time.

Map solutions of problem $B$ to solutions of $A$ in $O(|V|)$ time.

Reduction
Min Vertex Cover \leq Max Independent Set

Suppose there is no $O(|V|)$ algorithm for minimum vertex cover. Then, no $O(|V|)$ algorithm for maximum independent set.

Reduction:
- Map instances of problem $A$ to instances of $B$ in $O(1)$ time.
- Map solutions of problem $B$ to solutions of $A$ in $O(|V|)$ time.
Max Independent Set $\leq$ Min Vertex Cover

Suppose there is no $O(|V|)$ algorithm for maximum independent set.

Map instances of problem $A$ to instances of $B$

$O(1)$ time

Map solutions of problem $B$ to solutions of $A$

$O(|V|)$ time

Then, no $O(|V|)$ algorithm for minimum vertex cover.
Implications

Suppose $|V| = n$
- Maximum independent set reduces to minimum vertex cover in $O(n)$ time
- Minimum vertex cover reduces to maximum independent set in $O(n)$ time
- Any algorithm for either problems require $\Omega(n)$ time (why?)

Implications:
- Suppose there is a $T(n)$ algorithm for either problem
  - Then there is a $T(n)$ algorithm for both problems
- Suppose it takes $\Omega(T(n))$ time to solve either problem
  - Then it takes $\Omega(T(n))$ time to solve both problems

Interpretation: either both problems are easy (e.g., polynomial time) or both problems are hard (e.g., not polynomial time)
Suppose $|V| = n$

- Maximum independent set reduces to minimum vertex cover in $O(n)$ time
- Minimum vertex cover reduces to maximum independent set in $O(n)$ time
- Any algorithm for either problems require $\Omega(n)$ time (why?)

Implications:
- Suppose there is a $T_n$ algorithm for either problem
- Then there is a $T_n$ algorithm for the other
- Suppose it takes $\Omega(T_n)$ time to solve one of the two
- Then it takes $\Omega(T_n)$ time to solve the other

But we have no idea which is the case!

(But... likely that both are hard)

Interpretation: either both problems are easy (e.g., polynomial time) or both problems are hard (e.g., not polynomial time)
$k$-Independent Set

**Problem (Decision):** Given a graph $G = (V, E)$, is there an independent set with size $k$?
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

$k$-Independent set is an example of a decision problem:

- Answer is a single bit (e.g., true/false)
- Algorithm does not have to return the independent set (if there is one)

Can also define the search version of this problem:

- **Problem (Search):** Given a graph $G = (V, E)$, find an independent set with size $k$
- Output is an independent set of size $k$ (if there is one)

**Why should we care about the decision problem?**

We want to find the solution!
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

$k$-Independent set is an example of a decision problem:

- Answer is a single bit (e.g., true/false)
- Algorithm does not have to return the independent set (if there is one)

Can also define the search version of this problem:

- Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.
- Output is an independent set of size $k$ (if there is one)

Why should we care about the decision problem?

We want to find the solution!
Search-to-Decision Reduction

Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Goal: Find an independent set of size $k$

Idea: Remove a node, and check if the resulting graph still has an independent set of size $k$

- If not, then the node must be part of the independent set (so add the node to the set, and search for an independent set of size $k - 1$ in the remaining graph)
- If yes, then we do not need the node and continue searching over the remaining graph
**Problem (Decision):** Given a graph $G = (V, E)$, is there an independent set with size $k$?

**Problem (Search):** Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

Initially: $S = \emptyset$

Run decision algorithm with $k = 6$

**Output:** True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \emptyset$

Run decision algorithm with $k = 6$

Output: True

Remove a node from the graph
Search-to-Decision Reduction Example

Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?  
Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \emptyset$

Run decision algorithm with $k = 6$

**Output**: False
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B\}$

Run decision algorithm with $k = 5$

Output: True
Search-to-Decision Reduction Example

Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B\}$

Run decision algorithm with $k = 5$

Output: True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?
Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B\}$

Run decision algorithm with $k = 5$

Output: True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D\}$

Run decision algorithm with $k = 4$

Output: True
Search-to-Decision Reduction Example

Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D\}$

Run decision algorithm with $k = 4$

Output: True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D\}$

Run decision algorithm with $k = 4$

Output: True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D\}$

Run decision algorithm with $k = 4$

Output: False
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E\}$

Run decision algorithm with $k = 3$

Output: True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E\}$

Run decision algorithm with $k = 3$

**Output:** True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E\}$

Run decision algorithm with $k = 3$

**Output:** False
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?
Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E, I\}$

Run decision algorithm with $k = 2$

Output: True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E, I\}$

Run decision algorithm with $k = 2$

Output: False
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?
Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E, I, L\}$

Run decision algorithm with $k = 1$

Output: True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E, I, L\}$

Run decision algorithm with $k = 1$

Output: True
Search-to-Decision Reduction Example

Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E, I, L\}$

Run decision algorithm with $k = 1$

Output: False
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Consider $k = 6$

$S = \{B, D, E, I, L, H\}$

Invocations of decision algorithm: $O(|V|)$

Cost of reduction: $O(|V| + |E|)$
Search-to-Decision Reduction

Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

search

transform search problem into many instances of decision problem

decision

construct solution one node at a time

True
Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?
Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$. 

Search-to-Decision Reduction

transform search problem into many instances of decision problem

often times called a "self-reduction"

construct solution one node at a time
Search vs. Decision Problems

Problem (Decision): Given a graph $G = (V, E)$, is there an independent set with size $k$?

Problem (Search): Given a graph $G = (V, E)$, find an independent set with size $k$.

Search problems: “Find a solution to the problem”

Decision problems: “Does a solution to the problem exist?”

For many problems like $k$-independent set and $k$-vertex cover, there is a search-to-decision reduction (i.e., a self-reduction)
  • As we will see, this will be the case for any NP-complete problem

This is a key reason why we focus on decision problems rather than search problems
The Class NP

Given a graph $G$ and a set $S$, it is easy to check if $S$ is a $k$-independent set.

**Running Time:** $O(|E| + |V|)$

Given a graph $G$ and a set $S$, it is easy to check if $S$ is a $k$-vertex cover.

**Running Time:** $O(|E| + |V|)$

For both of these problems, it is easy to check a candidate solution.
The Class NP

Complexity class NP:
- **Decision problems** whose solutions can be checked efficiently (i.e., in polynomial time)
- Formally, we define problems in terms of languages $\mathcal{L} \subseteq \{0,1\}^*$ (i.e., infinite set of bitstrings)
  - $k$-independent set: $\mathcal{L} = \{G : G$ has an independent set of size $k\}$
  - $k$-vertex cover: $\mathcal{L} = \{G : G$ has a vertex cover of size $k\}$
- Solving a decision problem equates to deciding whether an instance $x$ (called a statement) is contained in the language $\mathcal{L}$ (i.e., deciding if $x \in \mathcal{L}$)
- A language $\mathcal{L} \in \text{NP}$ if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that $x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x, w) = 1$

Given a graph $G$ and a set $S$, it is **easy** to check if $S$ is a $k$-independent set.

**Running Time:** $O(|E| + |V|)$
The Class NP

Complexity class NP:
- **Decision problems** whose solutions can be checked efficiently (i.e., in polynomial time)
- Formally, we define problems in terms of **languages** $\mathcal{L} \subseteq \{0,1\}^*$ (i.e., infinite set of bitstrings)
  - $k$-independent set: $\mathcal{L} = \{ G : G \text{ has an independent set of size } k \}$
  - $k$-vertex cover: $\mathcal{L} = \{ G : G \text{ has a vertex cover of size } k \}$

Solving a decision problem equates to deciding whether an instance $x$ (called a statement) is contained in the language $\mathcal{L}$ (i.e., deciding if $x \in \mathcal{L}$)

A language $\mathcal{L} \in \text{NP}$ if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that

$$x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x, w) = 1$$

Given a graph $G$ and a set $S$, it is easy to check if $S$ is a $k$-independent set.

**Running Time:** $O(|E| + |V|)$
The Class NP

Complexity class NP:

- **Decision problems** whose solutions can be checked efficiently (i.e., in polynomial time)
- Formally, we define problems in terms of languages \( \mathcal{L} \subseteq \{0,1\}^* \) (i.e., infinite set of bitstrings)
  - \( k \)-independent set:
    \( \mathcal{L} = \{G: G \text{ has an independent set of size } k\} \)
  - \( k \)-vertex cover:

Given a graph \( G \) and a set \( S \), it is easy to check if \( S \) is a \( k \)-independent set

**Running Time:** \( O(|E| + |V|) \)

\( w \) is a “witness” or proof (of polynomial length) that the statement \( x \in \mathcal{L} \)

A language \( \mathcal{L} \in \text{NP} \) if there exists a deterministic polynomial-time algorithm \( \mathcal{R} \) such that

\[ x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x, w) = 1 \]
The Class NP

Given a graph $G$ and a set $S$, it is easy to check if $S$ is a $k$-independent set.

**Running Time:** $O(|E| + |V|)$

**Complexity class NP:**
- Decision problems whose solutions can be checked efficiently (i.e., in polynomial time)
- Formally, we define problems in terms of languages $\mathcal{L} \subseteq \{0,1\}^*$ (i.e., infinite set of bitstrings)
  - $k$-independent set:
    $\mathcal{L} = \{ G : G$ has an independent set of size $k \}$
  - $k$-vertex cover:
    $\mathcal{L} = \{ G : G$ has a vertex cover of size $k \}$
- Solving a decision problem equates to deciding whether an instance $x$ (called a statement) is contained in the language $\mathcal{L}$ (i.e., deciding if $x \in \mathcal{L}$)
- A language $\mathcal{L} \in$ NP if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that $x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x, w) = 1$

$\mathcal{R}$ is the “NP relation” or “solution-checker;” given an instance $x$ and a candidate solution $w$, $\mathcal{R}$ decides whether the solution is valid or not in polynomial time.
The Class P

A language $\mathcal{L} \in \text{NP}$ if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that

$$x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x,w) = 1$$

NP is the class of decision problems with efficiently-verifiable solutions

- Does not say anything about being able to find the solutions (i.e., we do not require that there is a polynomial-time algorithm to find $w$)

The class $\text{P}$ is the class of decision problems where solutions can be found efficiently (e.g., there is a polynomial-time algorithm that computes $w$ from $x$)

A language $\mathcal{L} \in \text{P}$ if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that

$$x \in \mathcal{L} \iff \mathcal{R}(x) = 1$$
The Class $P$

A language $\mathcal{L} \in \text{NP}$ if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that

$$x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x, w) = 1$$

NP is the class of decision problems with efficiently-verifiable solutions

• Does not say anything about being able to find the solutions (i.e., we do not require that there is a polynomial-time algorithm to find $w$)

The class $P$ is the class of decision problems where solutions can be found efficiently (e.g., there is a polynomial-time algorithm that computes $w$ from $x$)

A language $\mathcal{L} \in \text{P}$ if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that

$$x \in \mathcal{L} \iff \mathcal{R}(x) = 1$$

Polynomial in the input length (i.e., $\text{poly}(|x|) = O(|x|^d)$ for some $d \in \mathbb{N}$)
A language \( \mathcal{L} \in \text{NP} \) if there exists a deterministic polynomial-time "verifier" \( \mathcal{R} \) such that
\[
x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x, w) = 1
\]

A language \( \mathcal{L} \in \text{P} \) if there exists a deterministic polynomial-time "solver" \( \mathcal{R} \) such that
\[
x \in \mathcal{L} \iff \mathcal{R}(x) = 1
\]

If we can decide a problem in polynomial time, we can verify a solution to the problem in polynomial time:
\[
P \subseteq \text{NP}
\]

Biggest open problem in computer science: is this containment strict?

\[
P = \text{NP} \text{ or } P \neq \text{NP}
\]
A language $\mathcal{L} \in \text{NP}$ if there exists a deterministic polynomial-time “verifier” $\mathcal{R}$ such that
\[ x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)}: \mathcal{R}(x, w) = 1 \]

A language $\mathcal{L} \in \text{P}$ if there exists a deterministic polynomial-time “solver” $\mathcal{R}$ such that
\[ x \in \mathcal{L} \iff \mathcal{R}(x) = 1 \]

If we can decide a problem in polynomial time, we can verify a solution to the problem in polynomial time:
\[ \text{P} \subseteq \text{NP} \]

One of the seven Millennium Prize problems!

$\text{P} = \text{NP}$ or $\text{P} \neq \text{NP}$
A language $\mathcal{L} \in \text{NP}$ if there exists a deterministic polynomial-time “verifier” $\mathcal{R}$ such that $w \in \{0,1\}^{\text{poly}(|x|)}$: $\mathcal{R}(x, w) = 1$

A language $\mathcal{L} \in \text{P}$ if there exists a deterministic polynomial-time “solver” $\mathcal{R}$ such that $x \in \mathcal{L} \iff \mathcal{R}(x) = 1$

If we can decide a problem in polynomial time, we can verify a solution to the problem in polynomial time:

$\text{P} \subseteq \text{NP}$

**Biggest open problem in computer science**: is this containment strict?

$\text{P} = \text{NP}$ or $\text{P} \neq \text{NP}$
A language $\mathcal{L} \in \text{NP}$ if there exists a deterministic polynomial-time “verifier” $\mathcal{R}$ such that

$$x \in \mathcal{L} \iff \exists w \in \{0, 1\}^*$ : \mathcal{R}(x, w) = 1$$

A language $\mathcal{L} \in \text{P}$ if there exists a deterministic polynomial-time “solver” $\mathcal{R}$ such that

$$x \in \mathcal{L} \iff \mathcal{R}(x) = 1$$

NP: “non-deterministic polynomial time”
- Non-deterministic has nothing to do with randomness
- Non-deterministic refers to a computation taking many possible paths (e.g., $\mathcal{R}(x, \cdot)$ can be viewed as a non-deterministic algorithm that tries every possible value of $w$ and sees if any of the branches accept – there can be exponentially-many branches, but checking each branch is polynomial time)

If we can decide a problem in polynomial time, we can verify a solution to the problem in polynomial time:

$$\text{P} \subseteq \text{NP}$$

Biggest open problem in computer science: is this containment strict?

$$\text{P} = \text{NP} \text{ or } \text{P} \neq \text{NP}$$
**Show:** For any graph $G$:
- There is a short witness (i.e., proof) that $G$ has a $k$-independent set
- The proof can be checked efficiently (in polynomial time)

**Witness for $G$:** $S = \{A, C, E, G, H, J\}$
(nodes in the $k$-independent set)

**Checking the witness:**
- Check that $|S| = k$
- Check that every edge is incident on at most one node in $S$

**Total time:** $O(|E| + |V|) = \text{poly}(|V| + |E|)$