CS 4102: Algorithms

Lecture 24: P vs. NP

David Wu

Fall 2019

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

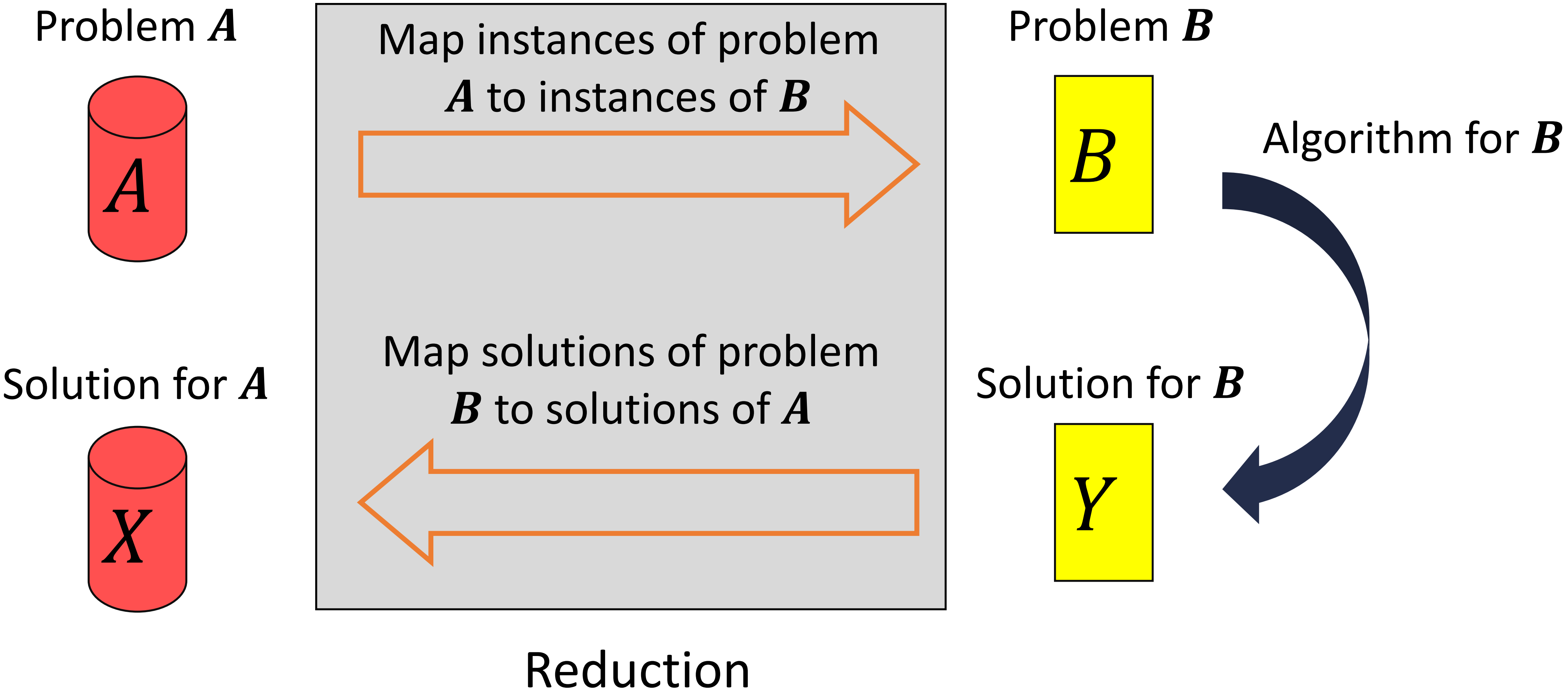
Monday, December 9, 7pm in Olsson 120

- 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

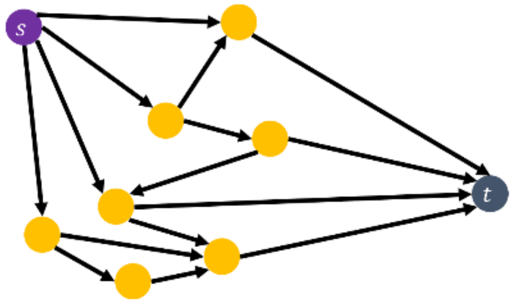
Reductions



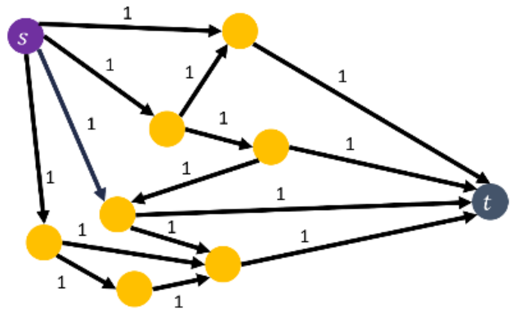
$A \leq B$: there is a reduction from A to B

Reduction Examples

edge-disjoint paths



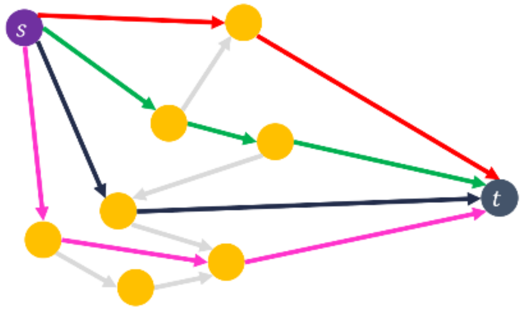
max flow

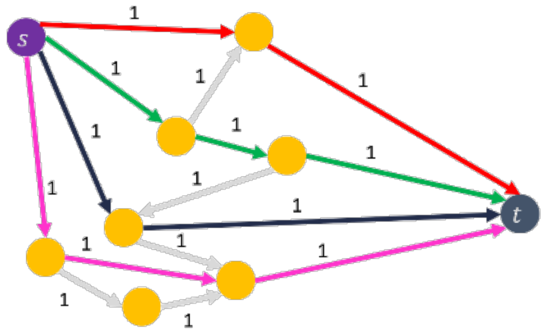


Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$

Ford-Fulkerson

Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$

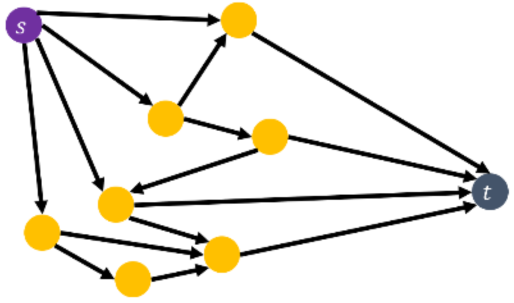




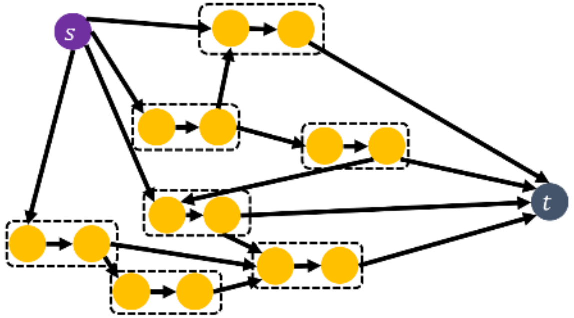
Reduction

Reduction Examples

vertex-disjoint paths

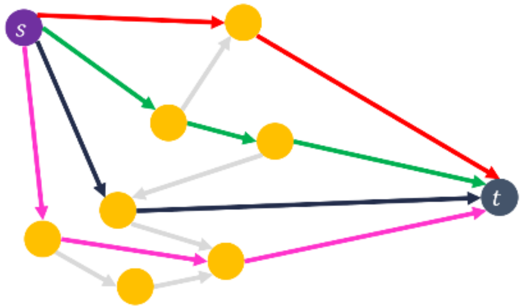


edge-disjoint paths

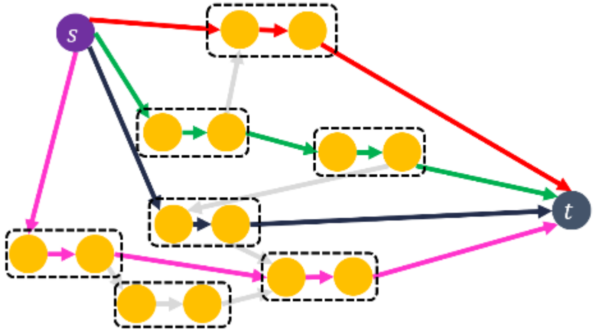


Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$

Edge-disjoint paths algorithm



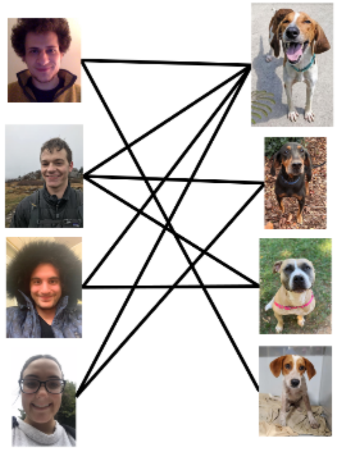
Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$

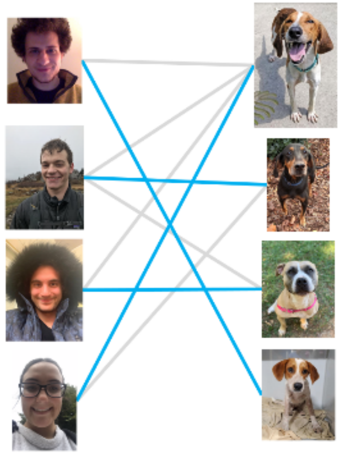


Reduction

Reduction Examples

maximum bipartite matching



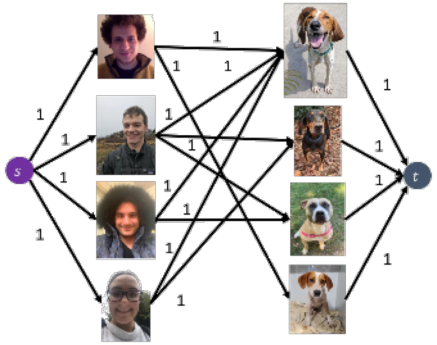


Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$

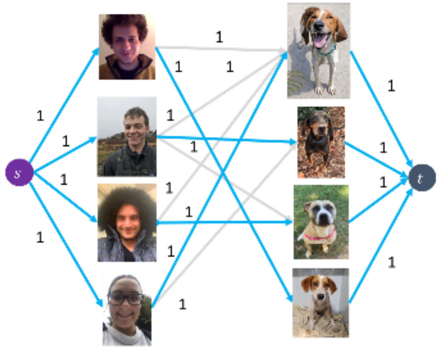
Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$

Reduction

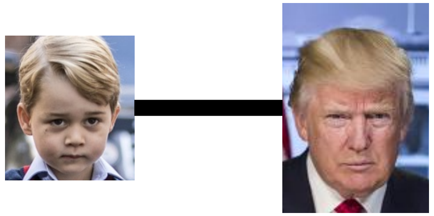
max flow



Ford-Fulkerson

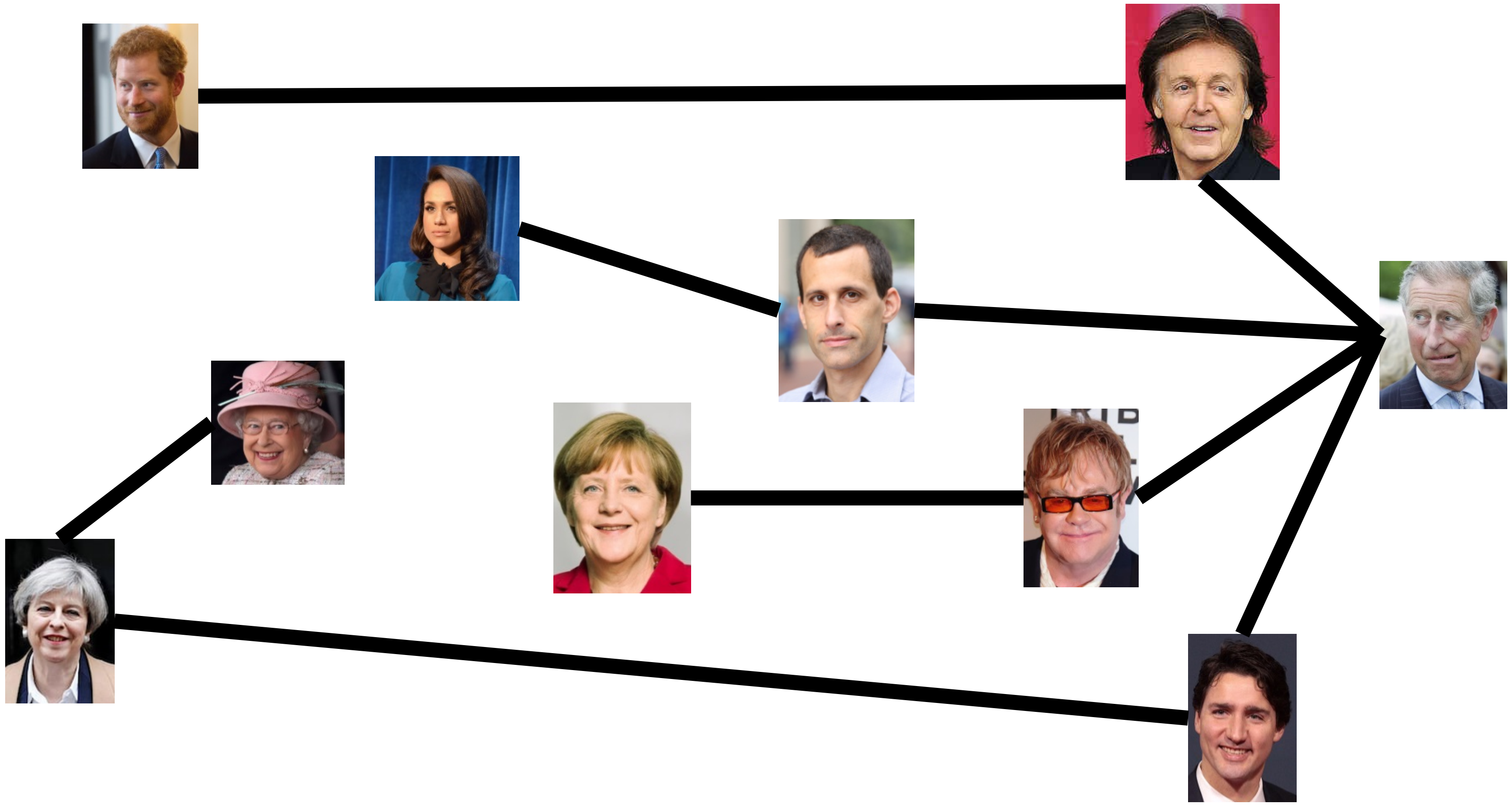


Party Problem



Draw edges between people who do not get along

Goal: Find the maximum number of people who get along

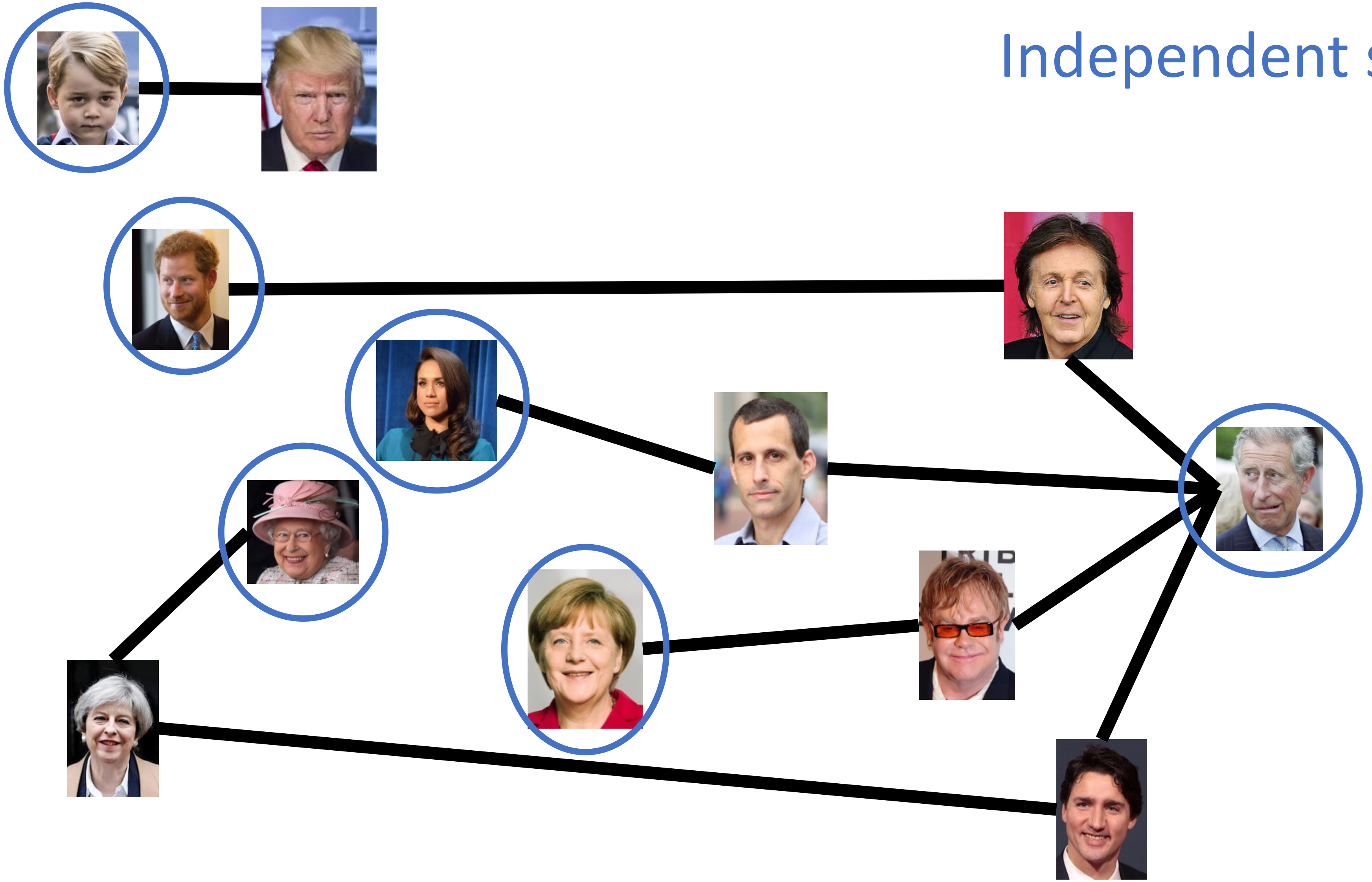


Maximum Independent Set

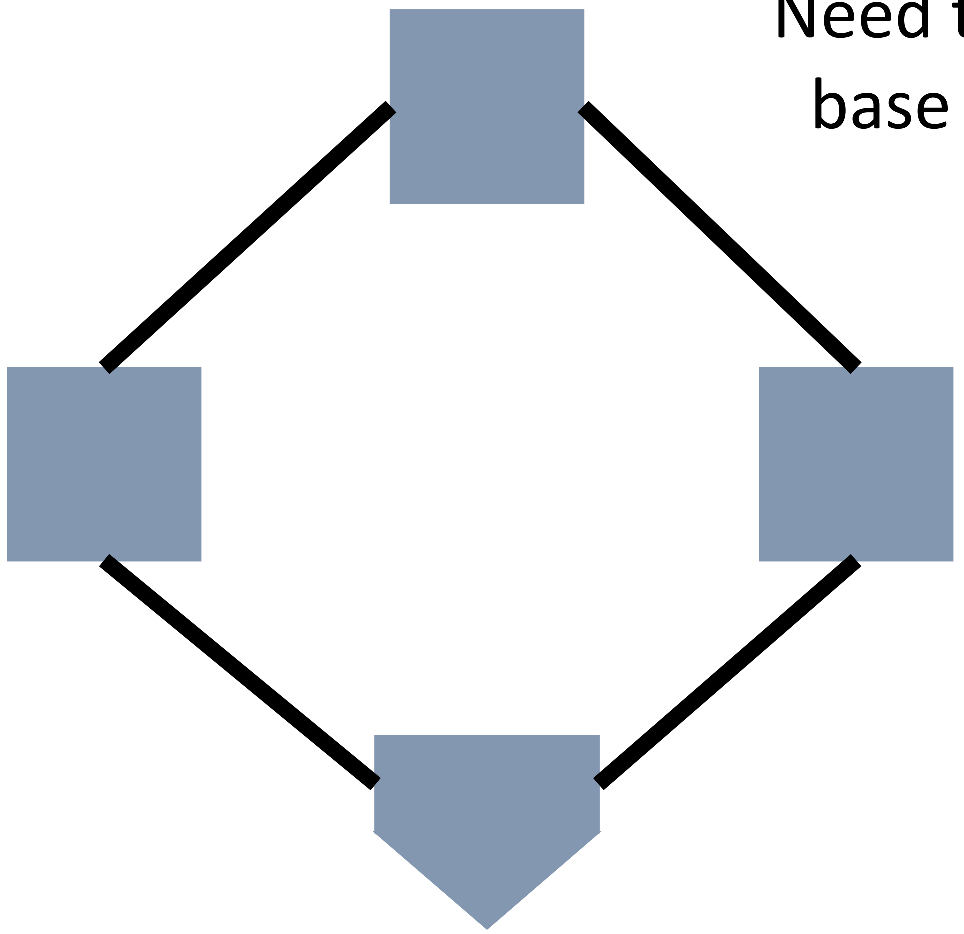
Independent set: $S \subseteq V$ is an independent set if no two nodes in S share an edge

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

Maximum Independent Set Example

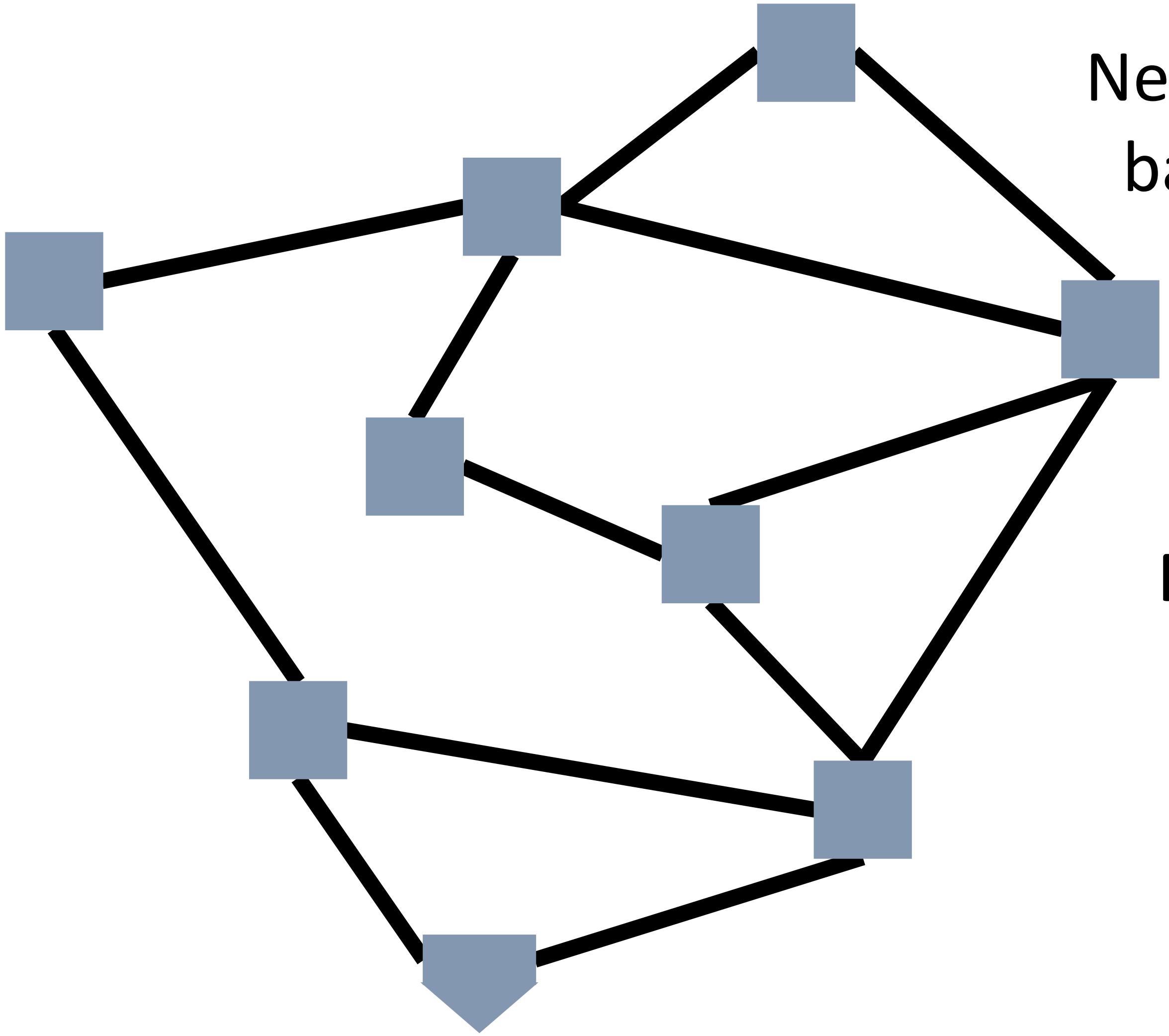


Classic Baseball



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

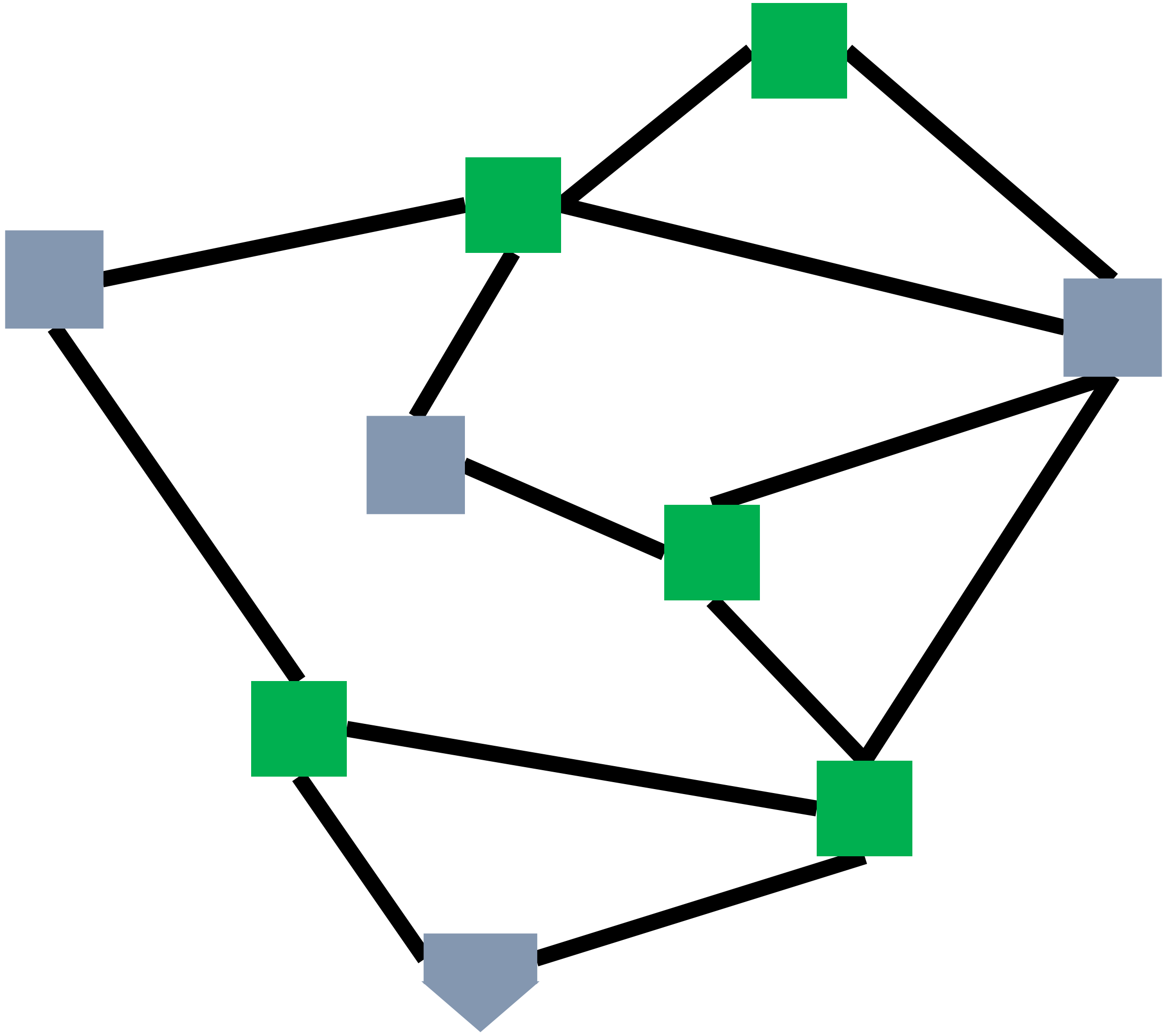
Problem: Fewest number of defenders required?

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

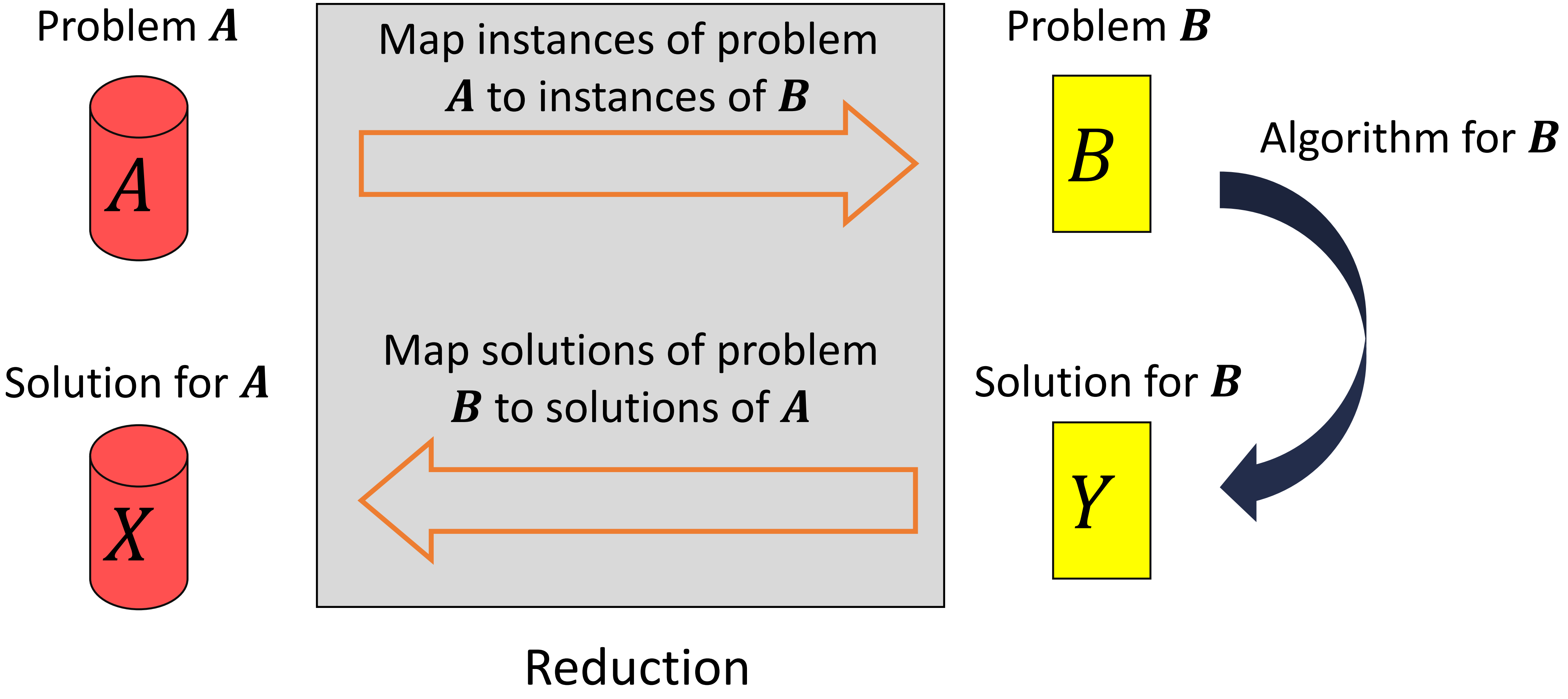
Vertex Cover Example



Vertex cover of size 5

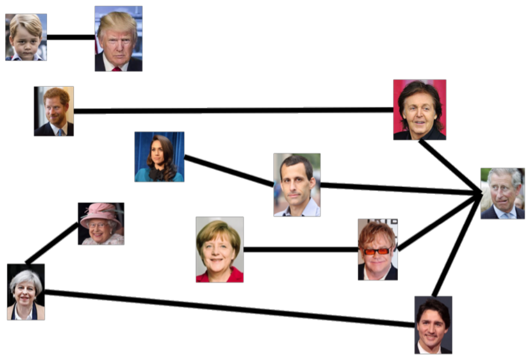
Turns out that problem of finding a minimum vertex cover is closely related to problem of finding a maximum independent set

Reductions

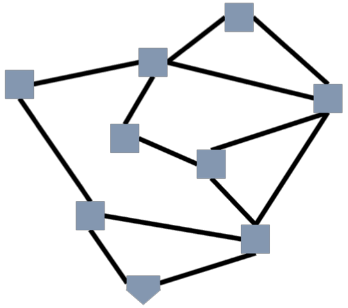


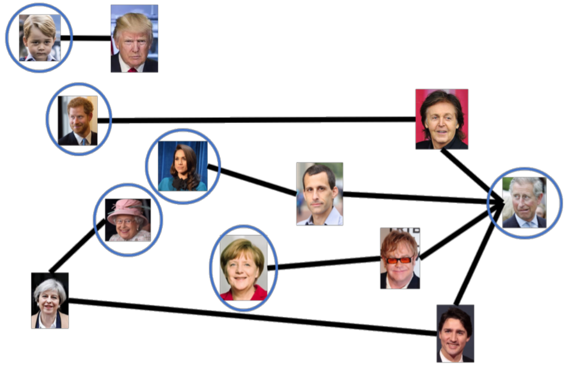
$\boldsymbol{A \leq B}$: there is a reduction from $\boldsymbol{A}$ to $\boldsymbol{B}$

Max Independent Set $\leq$ Min Vertex Cover

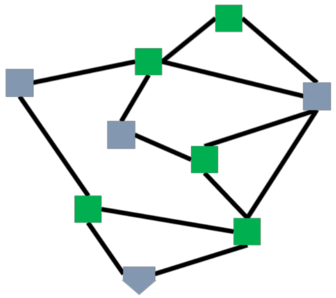


Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$





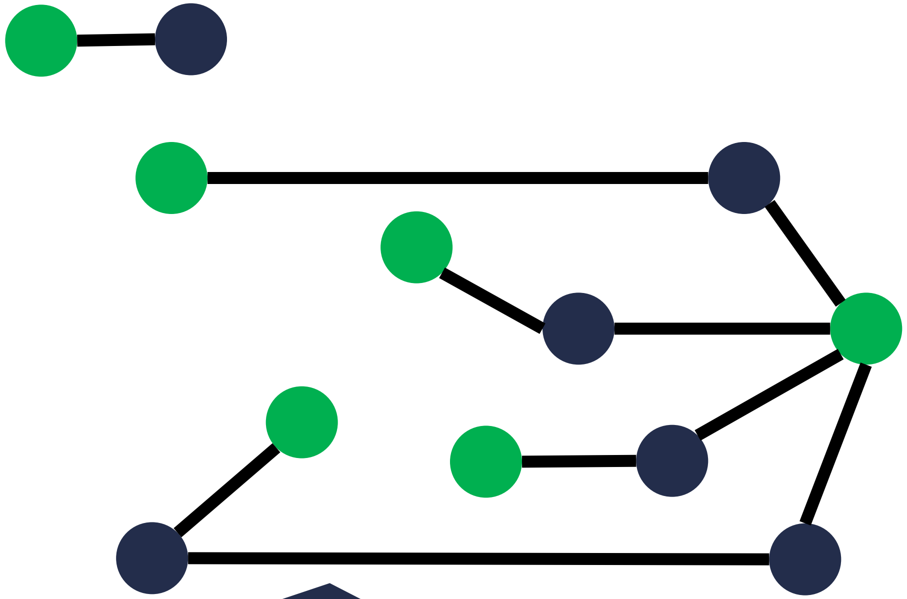
Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$



Reduction

$\boldsymbol{A} \leq \boldsymbol{B}$: there is a reduction from $\boldsymbol{A}$ to $\boldsymbol{B}$

Max Independent Set $\leq$ Min Vertex Cover



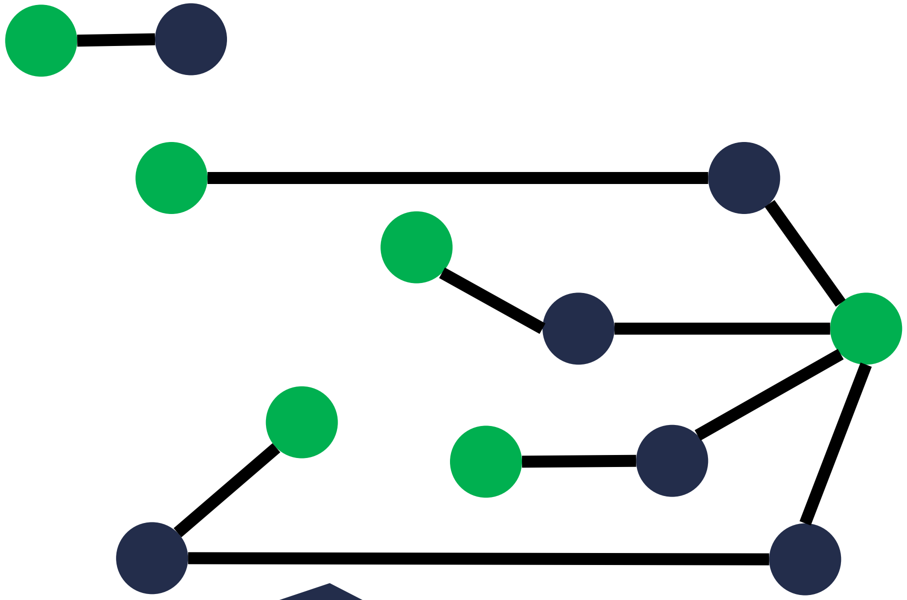
Important note: a maximum independent set may <u>not</u> be a vertex cover

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

Max Independent Set $\leq$ Min Vertex Cover



Important note: a maximum independent set may not be a vertex cover

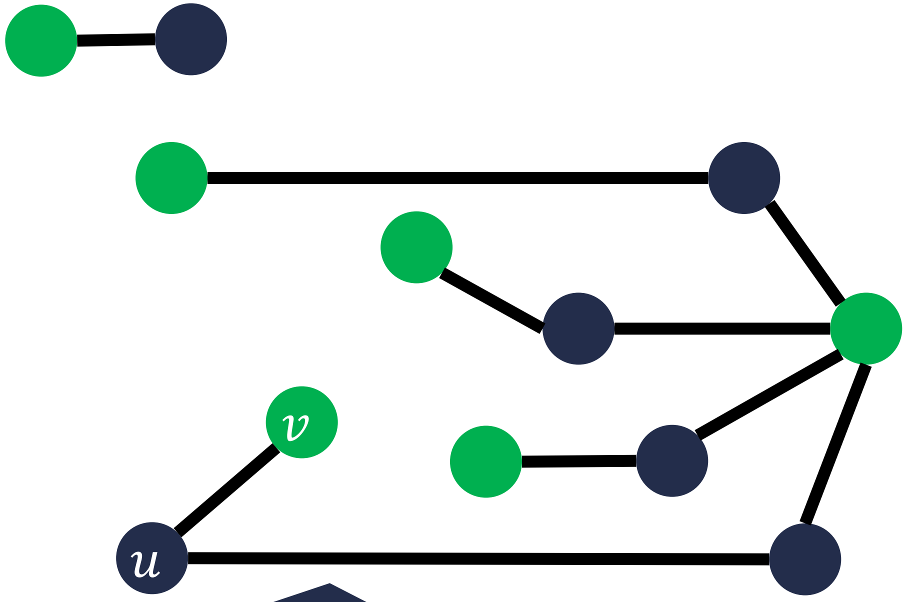
Independent set: set of nodes that do not share an edge

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

Max Independent Set $\leq$ Min Vertex Cover



Important note: a maximum independent set may <u>not</u> be a vertex cover

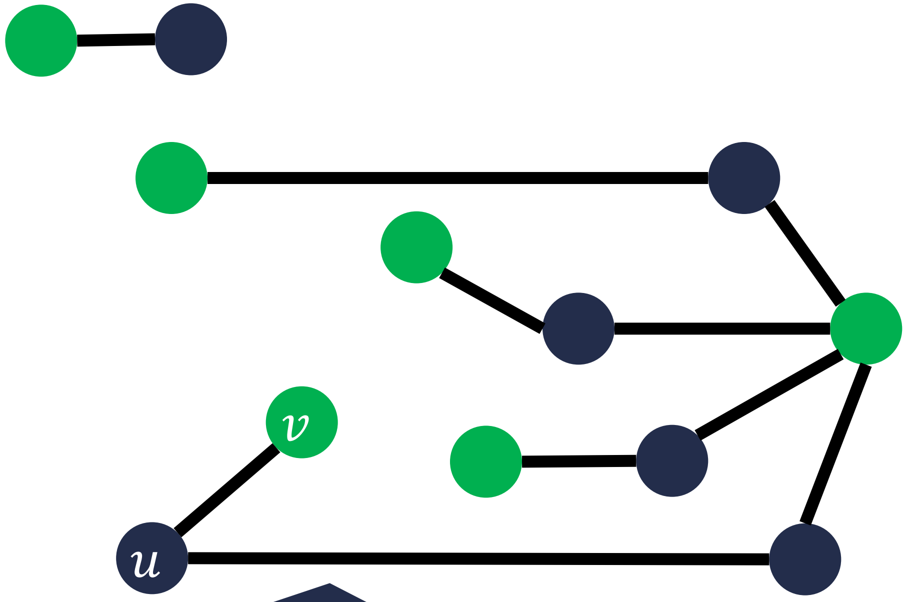
Independent set: set of nodes that do not share an edge

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 $V - S$

Max Independent Set ≤ Min Vertex Cover



Important note: a maximum independent set may <u>not</u> be a vertex cover

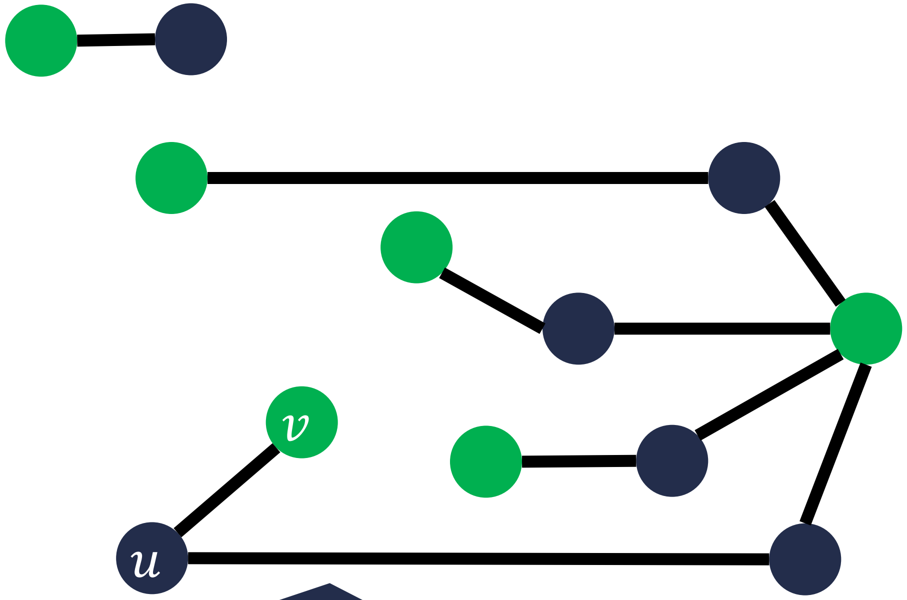
Independent set: set of nodes that do not share an edge

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 $u \notin S$ or $v \notin S$ (or both)
- Thus, there is no edge between any pair of nodes $u, v \in S$

Max Independent Set $\leq$ Min Vertex Cover



Important note: a maximum independent set may not be a vertex cover

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

Vertex cover: set of nodes that cover all edges

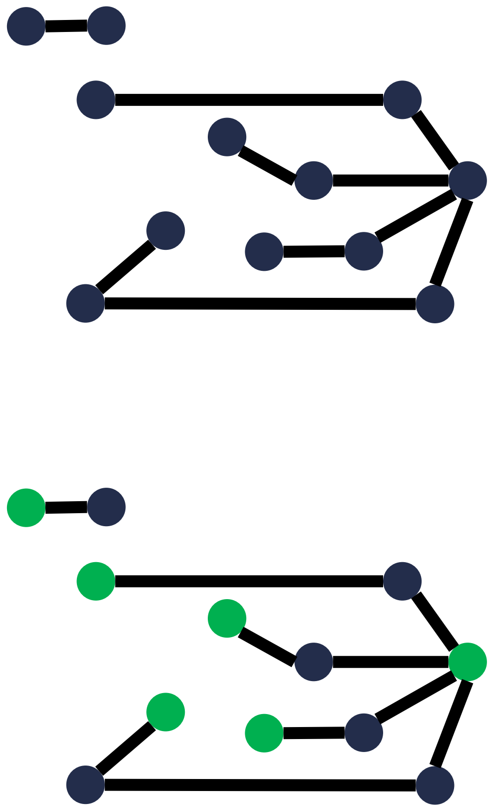
Claim: S is an independent set if and only if its complement $V - S$ is a vertex cover

Conclusions:

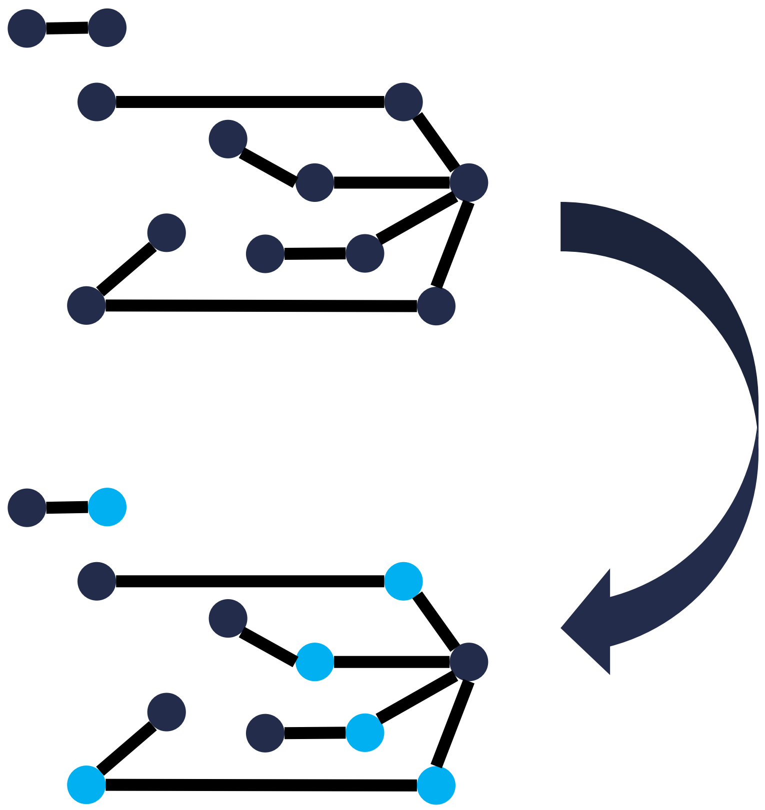
- There is a one-to-one correspondence between independent sets and vertex covers
- Independent sets and vertex covers are complements so maximizing one means minimizing the other

Max Independent Set $\leq$ Min Vertex Cover

maximum independent set



minimum vertex cover



Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$

$O(1)$ time

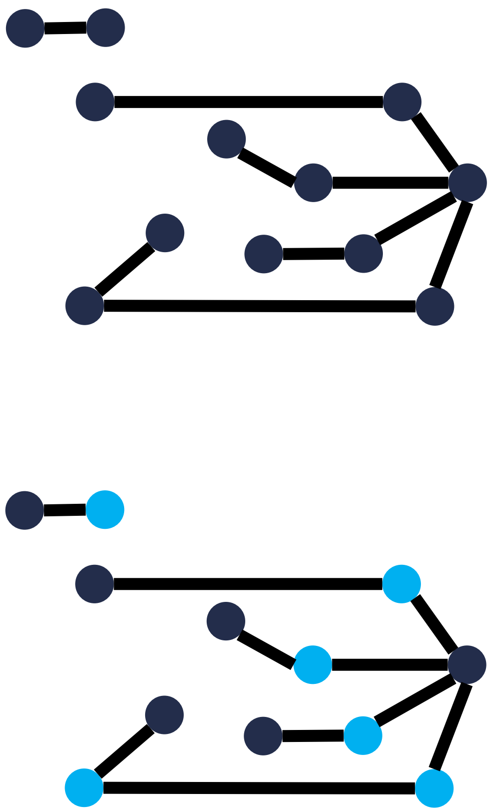
Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$

$O(|V|)$ time

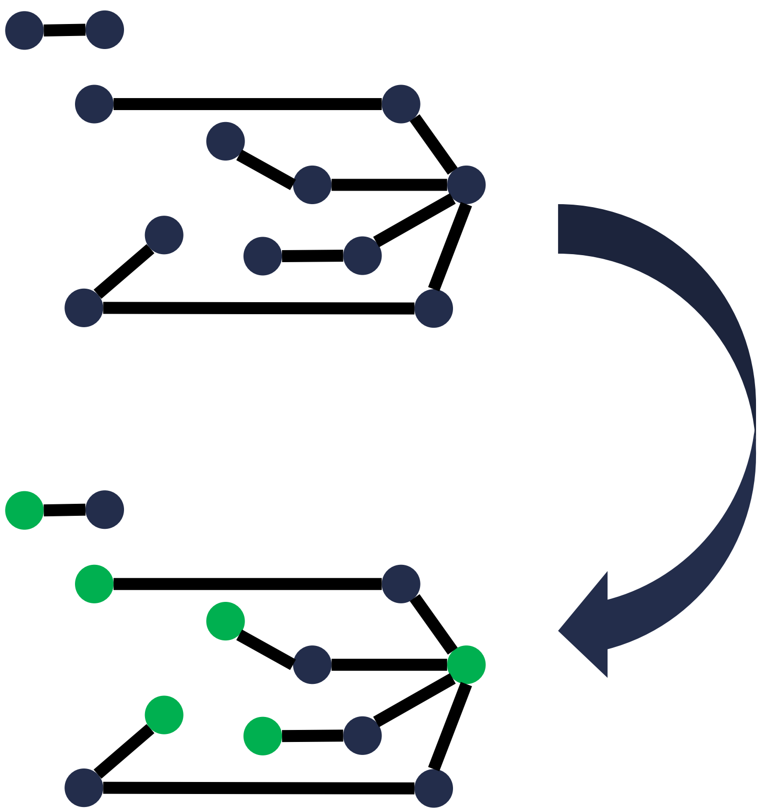
Reduction

Min Vertex Cover $\leq$ Max Independent Set

minimum vertex cover



maximum independent set



Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$

$O(1)$ time

Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$

$O(|V|)$ time

Reduction

Min Vertex Cover $\leq$ Max Independent Set

minimum vertex cover

Suppose there is no $O(|V|)$ algorithm for minimum vertex cover

Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$

$O(1)$ time

Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$

$O(|V|)$ time

Reduction

maximum independent set

Then, no $O(|V|)$ algorithm for maximum independent set

Max Independent Set $\leq$ Min Vertex Cover

maximum independent set

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

Map instances of problem $\boldsymbol{A}$ to instances of $\boldsymbol{B}$

$O(1)$ time

Map solutions of problem $\boldsymbol{B}$ to solutions of $\boldsymbol{A}$

$O(|V|)$ time

Reduction

minimum vertex cover

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 $\Omega\big(T(n)\big)$ time to solve either problem
  - Then it takes $\Omega\big(T(n)\big)$ 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)

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:

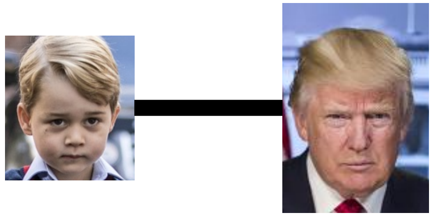
- Suppose the
  - Then the
- Suppose it ta
  - Then it ta

But we have no idea which is the case!

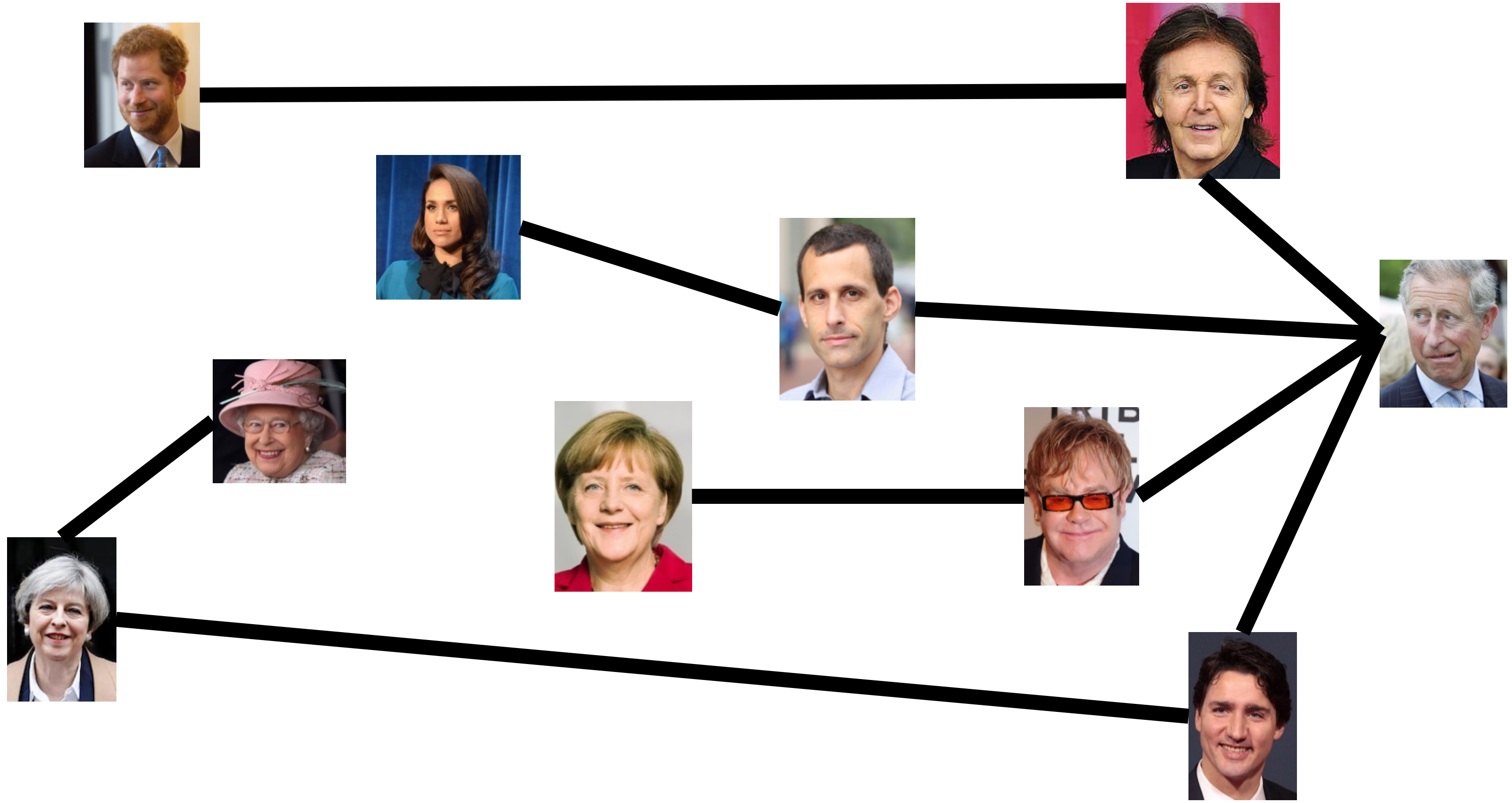
(but… likely that both are hard)

Interpretation: either both problems are easy (e.g., polynomial time) or both problems are hard (e.g., not polynomial time)

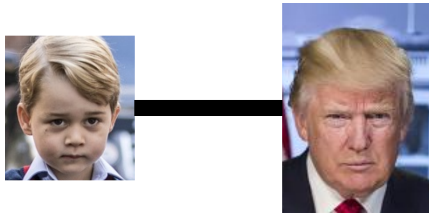
k-Independent Set



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



k-Independent Set



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)

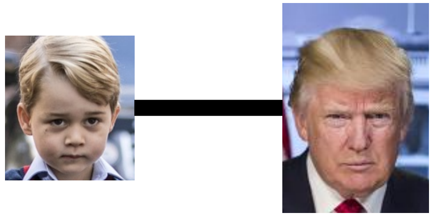
Can also define the search version of this problem:

- **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!

k-Independent Set



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

k-Independent set is an example of a <u>decision</u> problem:

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

Can also define the <u>sea</u>

- **Problem (Search):** G t set with size k.
- Output is an indepe

Because oftentimes, the search problem <u>reduces</u> to the decision problem, so it suffices to analyze the <u>simpler</u> problem

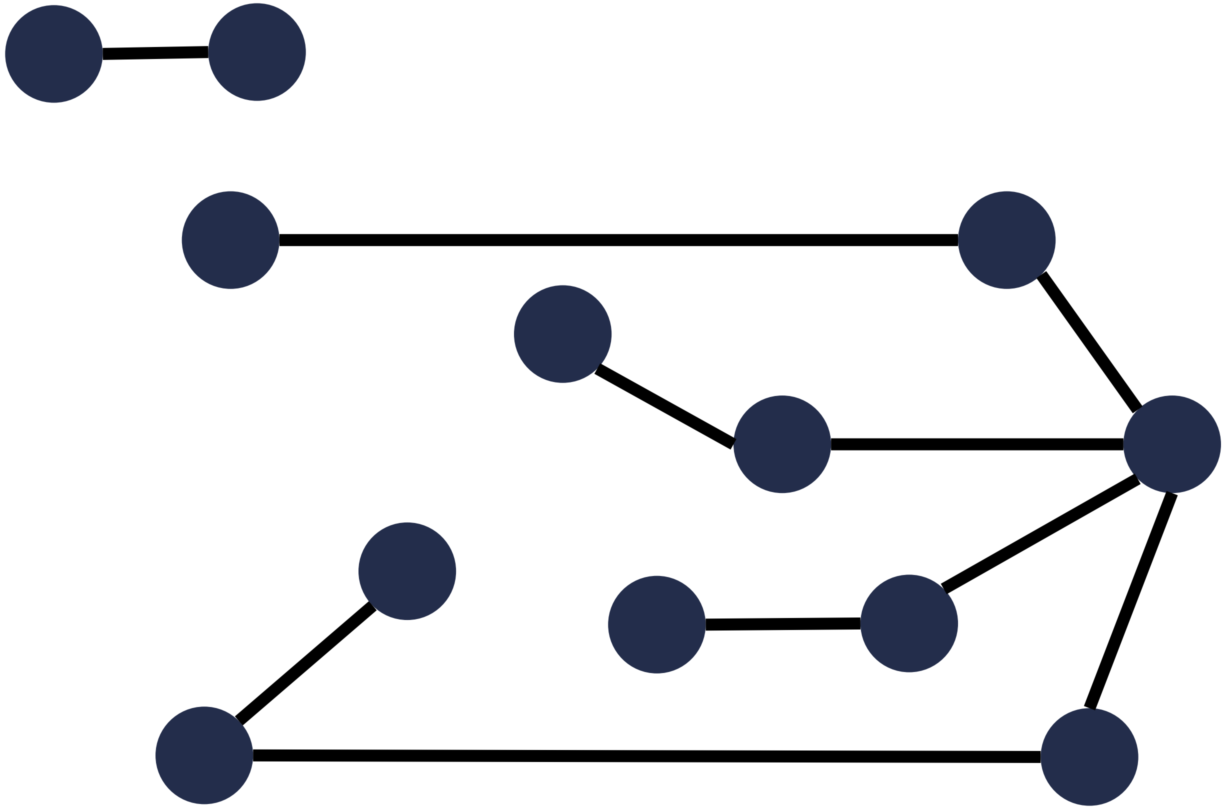
Why should we care about the decision problem?

We want to <u>find</u> 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

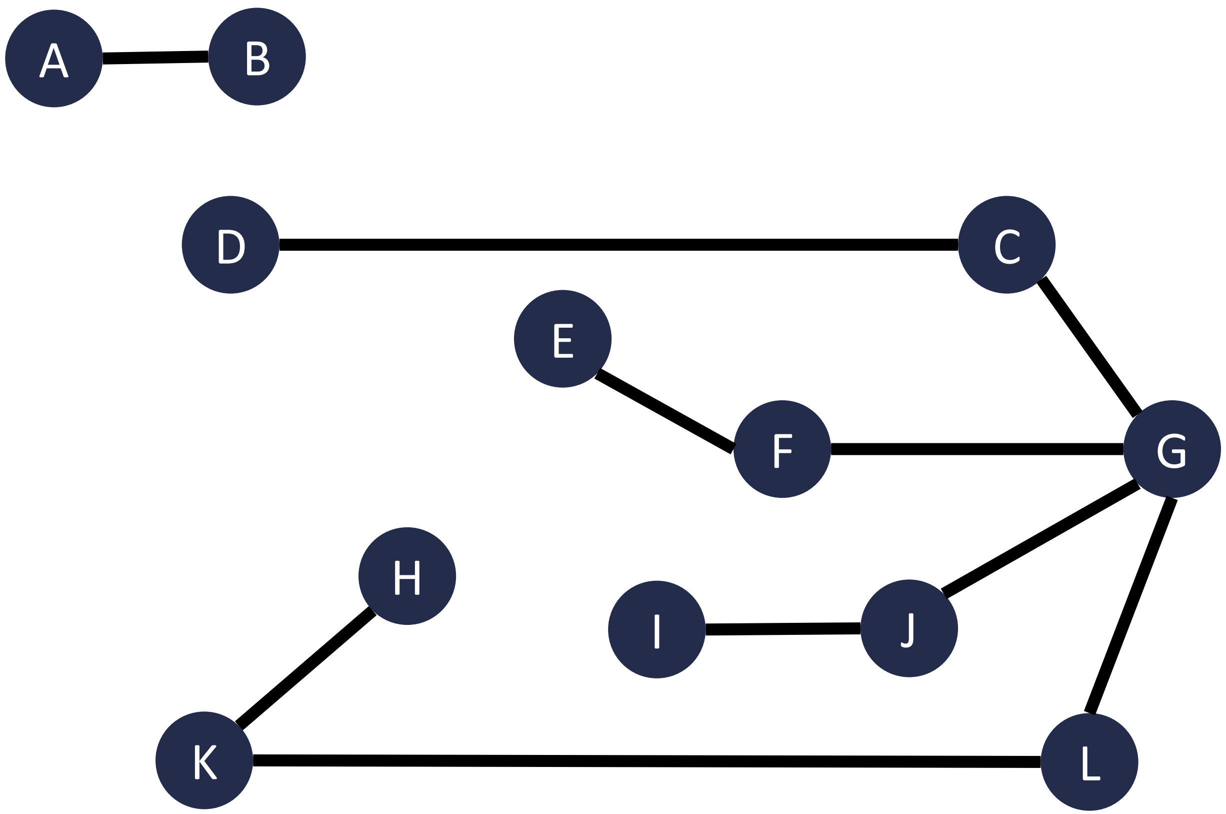
Idea: Remove a node, and check if the resulting graph still has an independent set of size k

- 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

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 $k = 6$

Initially: $S = \emptyset$

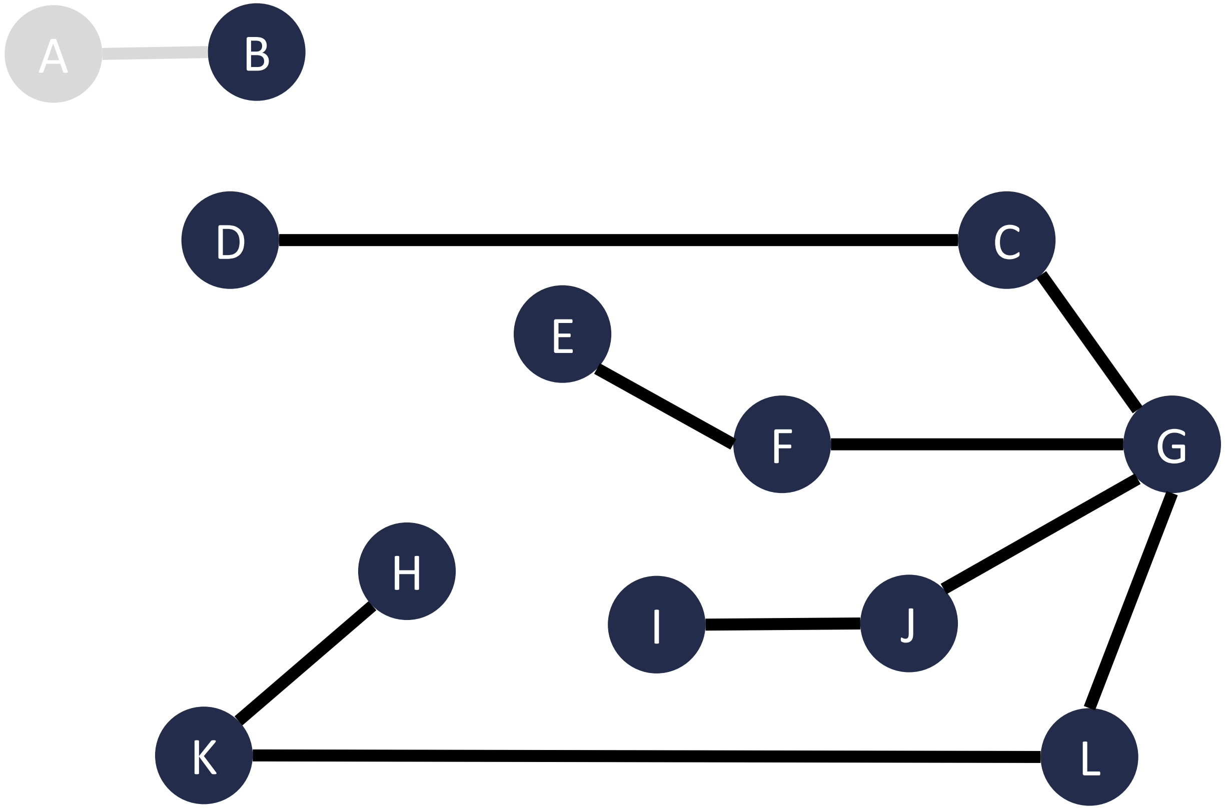
Run decision algorithm with $k = 6$

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 $k = 6$

$S = \emptyset$

Run decision algorithm with $k = 6$

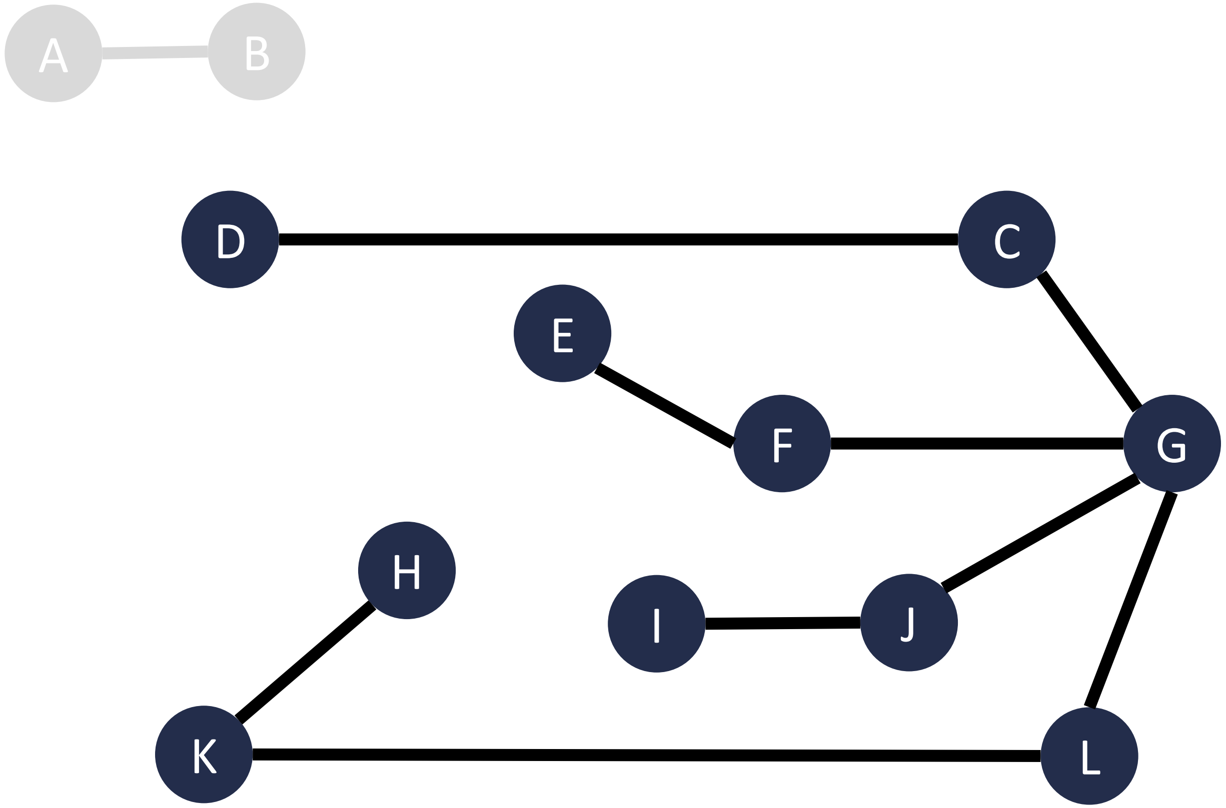
Output: True

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 $k = 6$

$S = \emptyset$

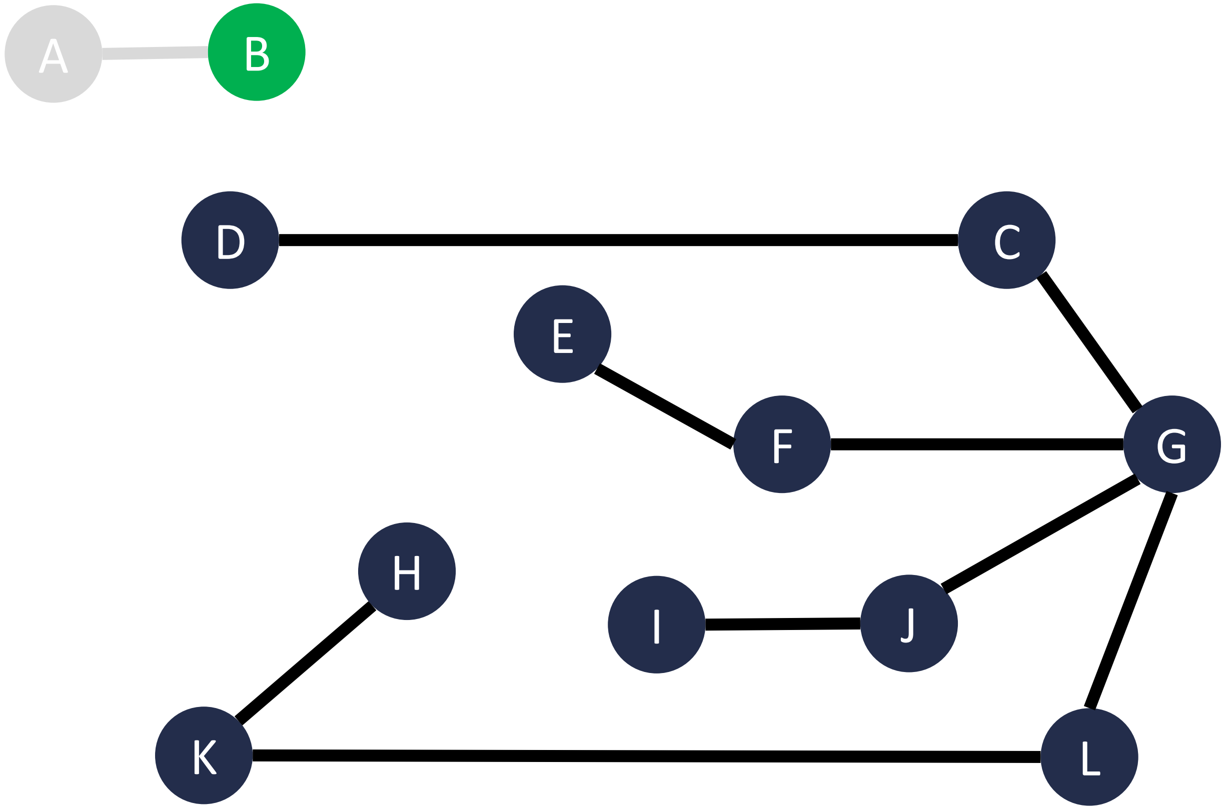
Run decision algorithm with $k = 6$

Output: False

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 $k = 6$

$S = \{B\}$

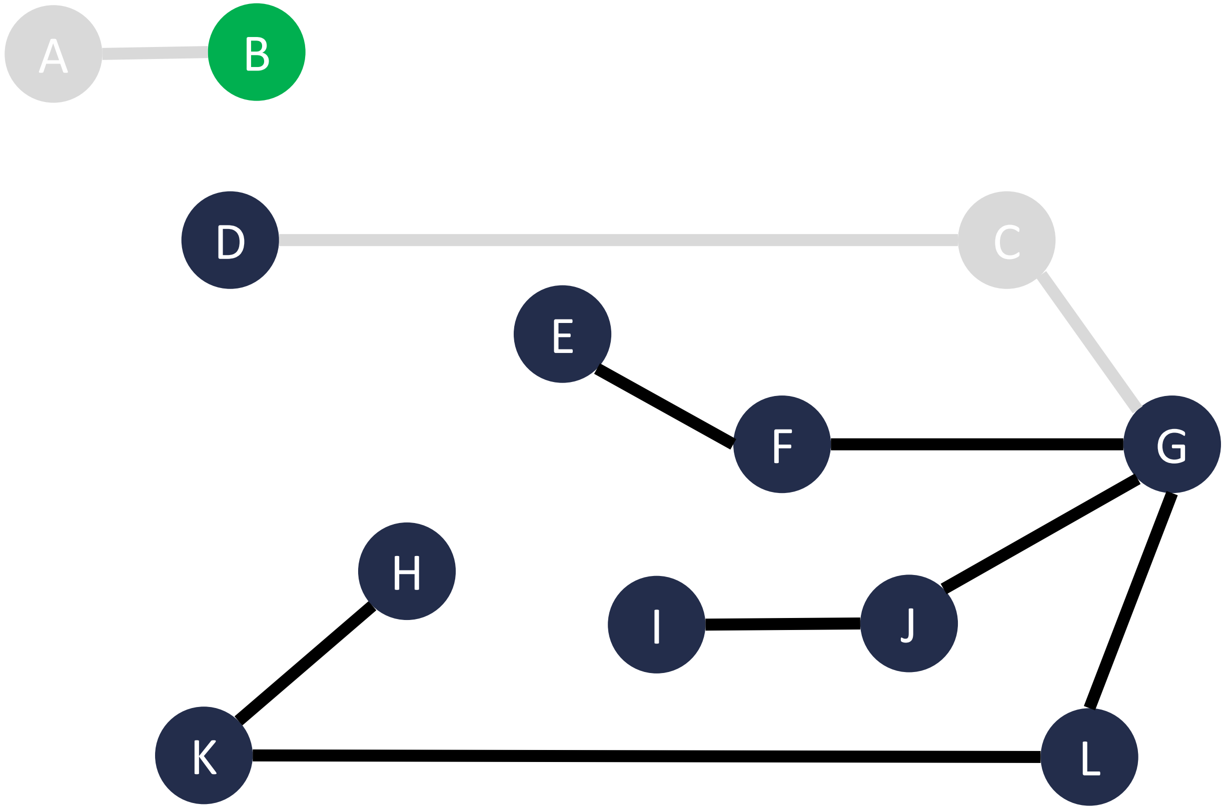
Run decision algorithm with $k = 5$

Output: True

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 $k = 6$

$S = \{B\}$

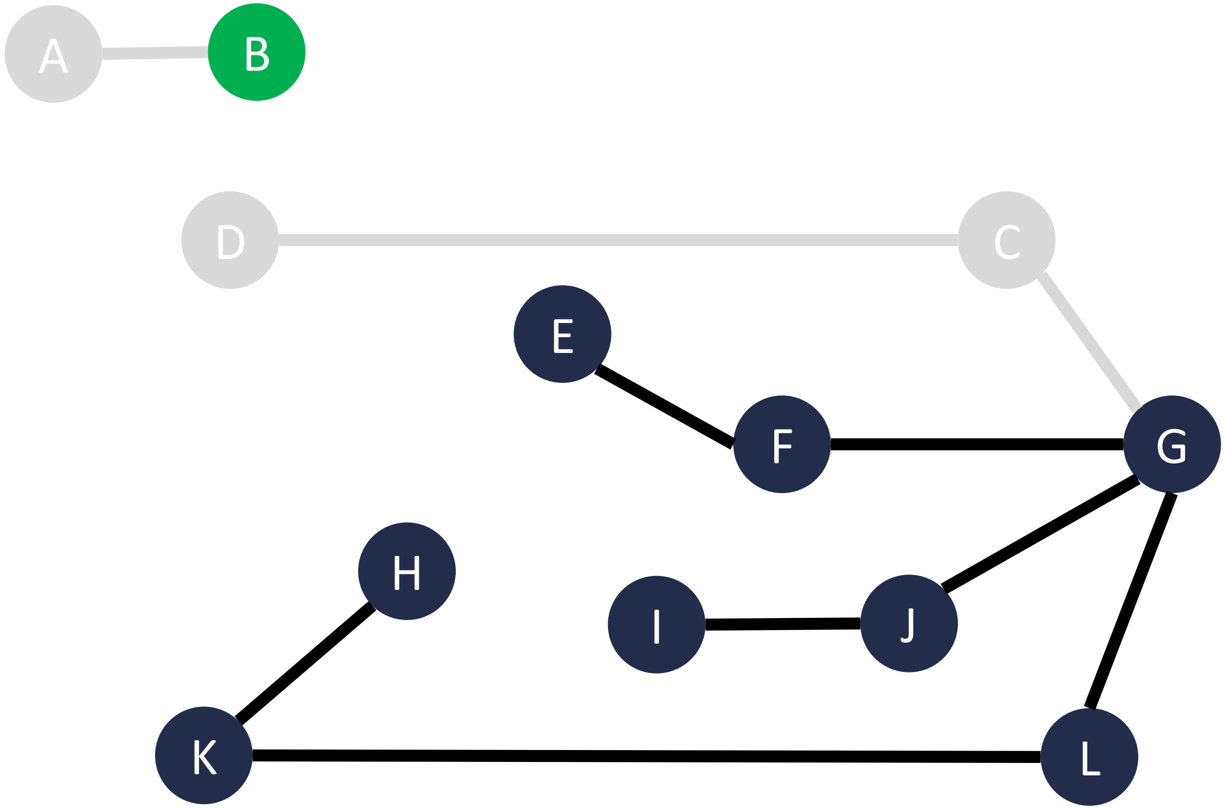
Run decision algorithm with $k = 5$

Output: True

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 $k = 6$

$S = \{B\}$

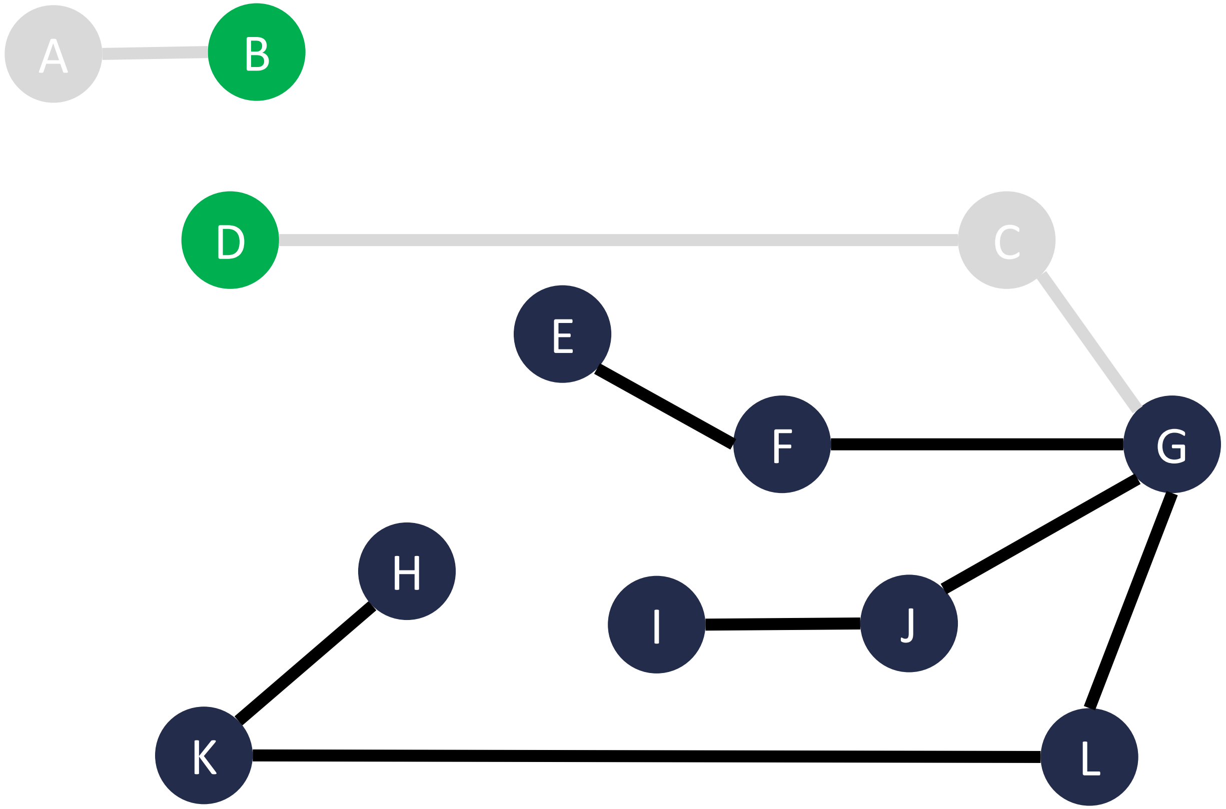
Run decision algorithm with $k = 5$

Output: True

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 $k = 6$

$S = \{B, D\}$

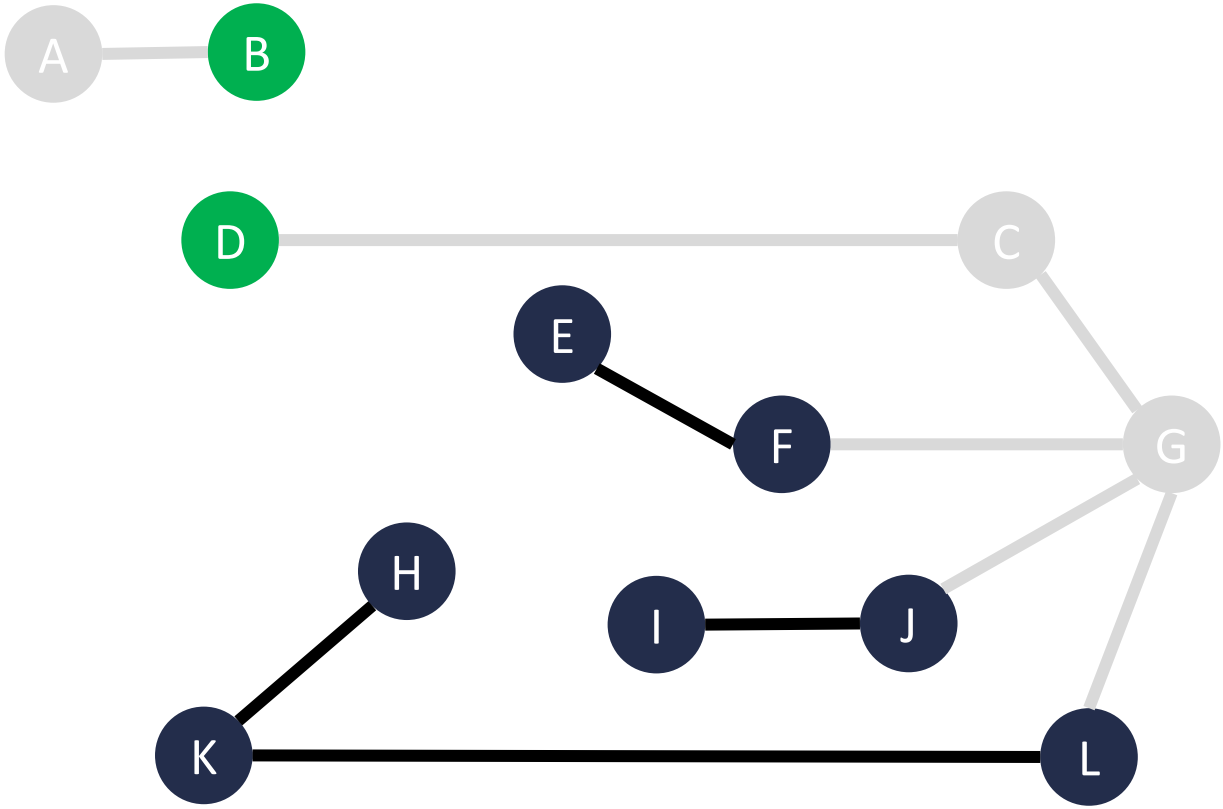
Run decision algorithm with $k = 4$

Output: True

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 $k = 6$

$S = \{B, D\}$

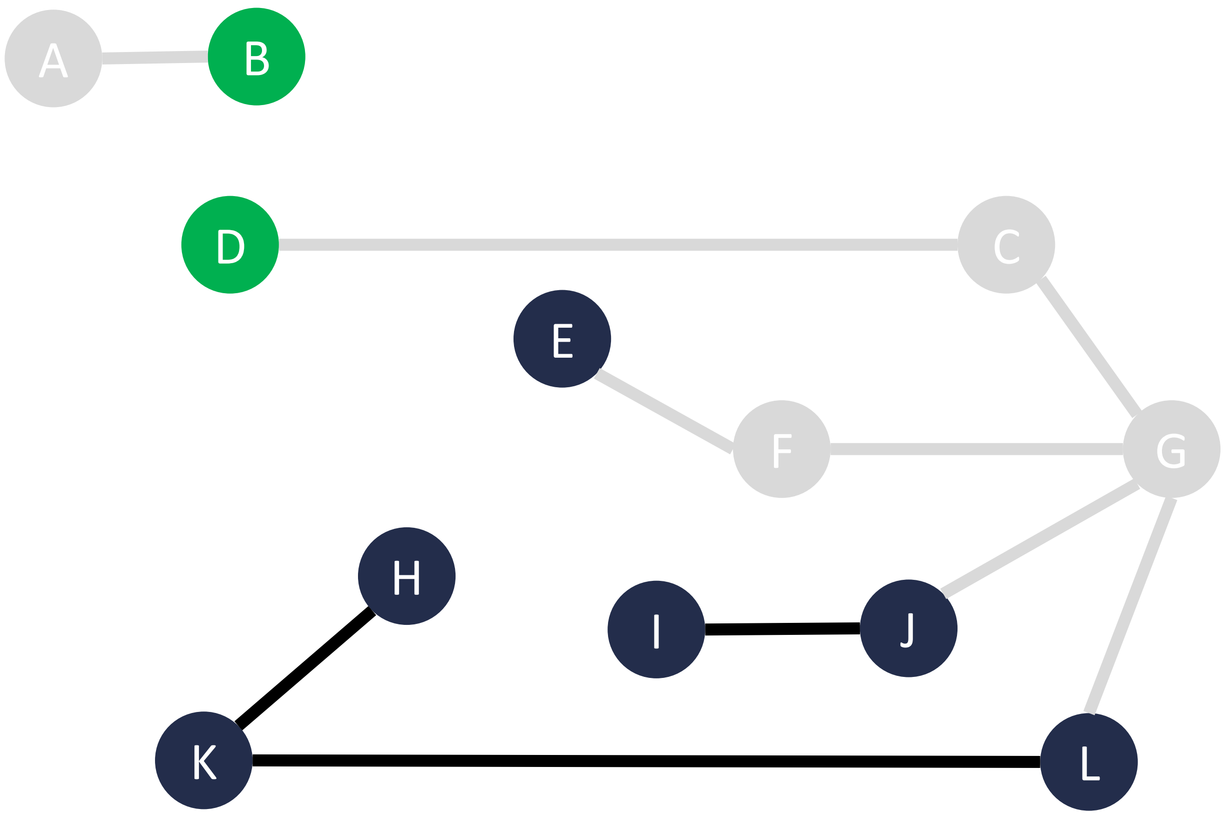
Run decision algorithm with $k = 4$

Output: True

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 $k = 6$

$S = \{B, D\}$

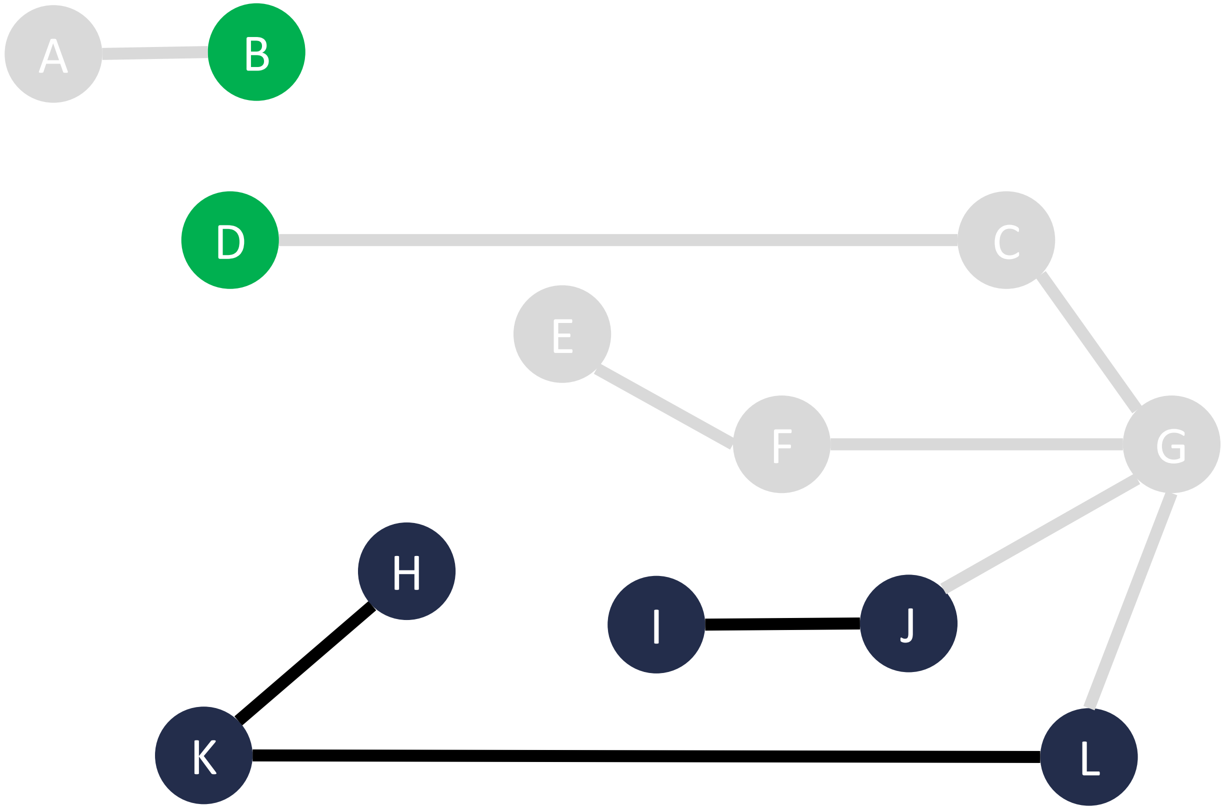
Run decision algorithm with $k = 4$

Output: True

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 $k = 6$

$S = \{B, D\}$

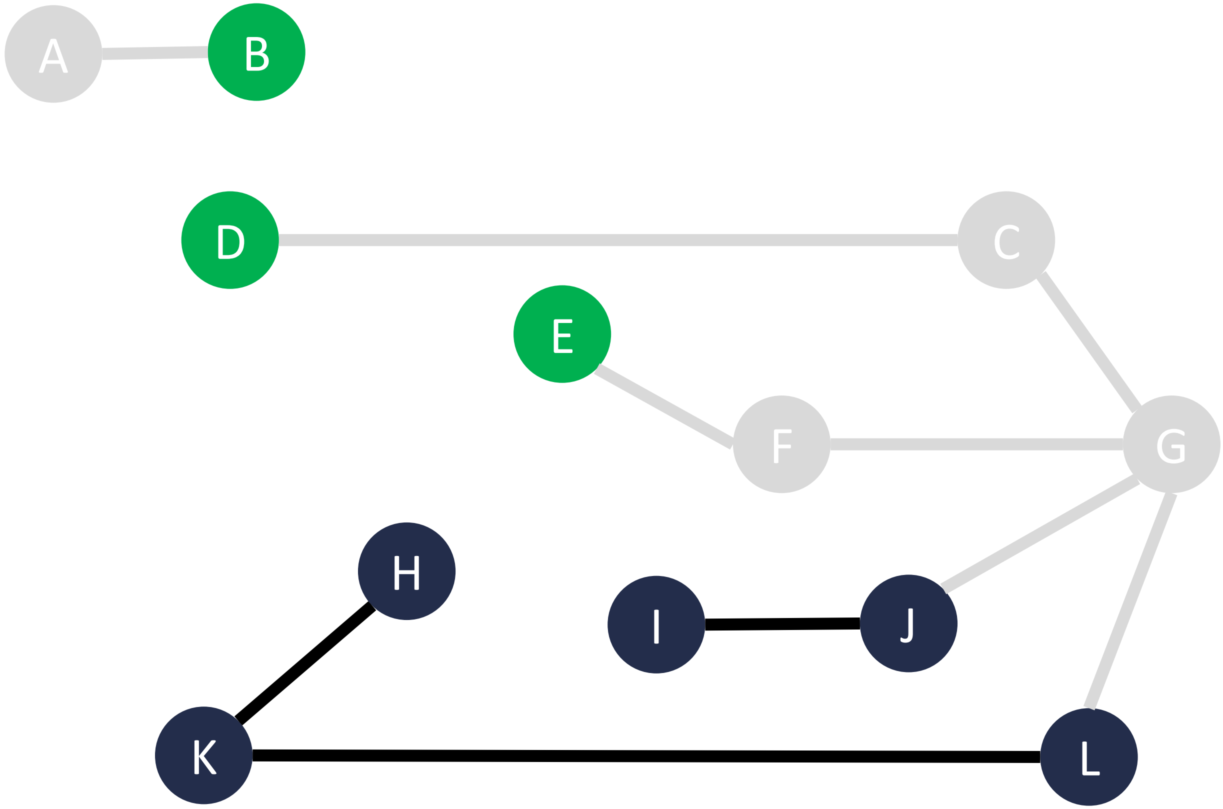
Run decision algorithm with $k = 4$

Output: False

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 $k = 6$

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

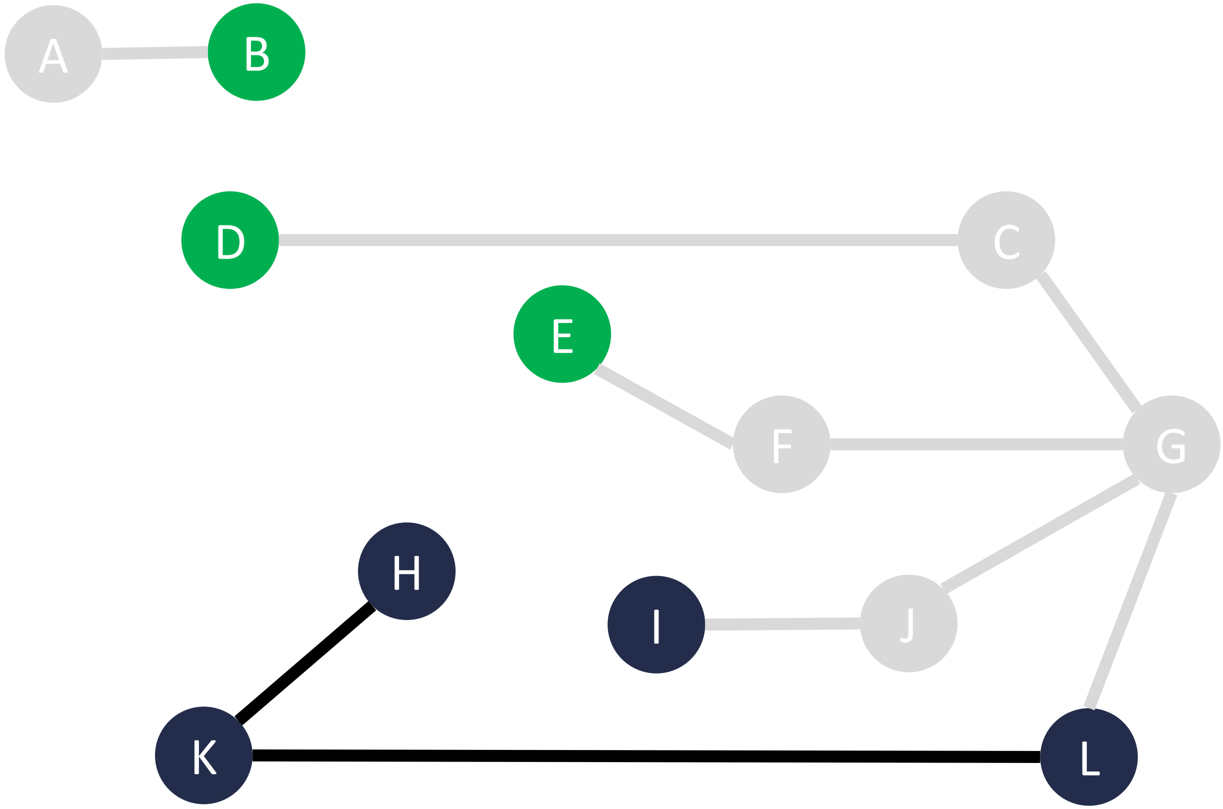
Run decision algorithm with $k = 3$

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 $k = 6$

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

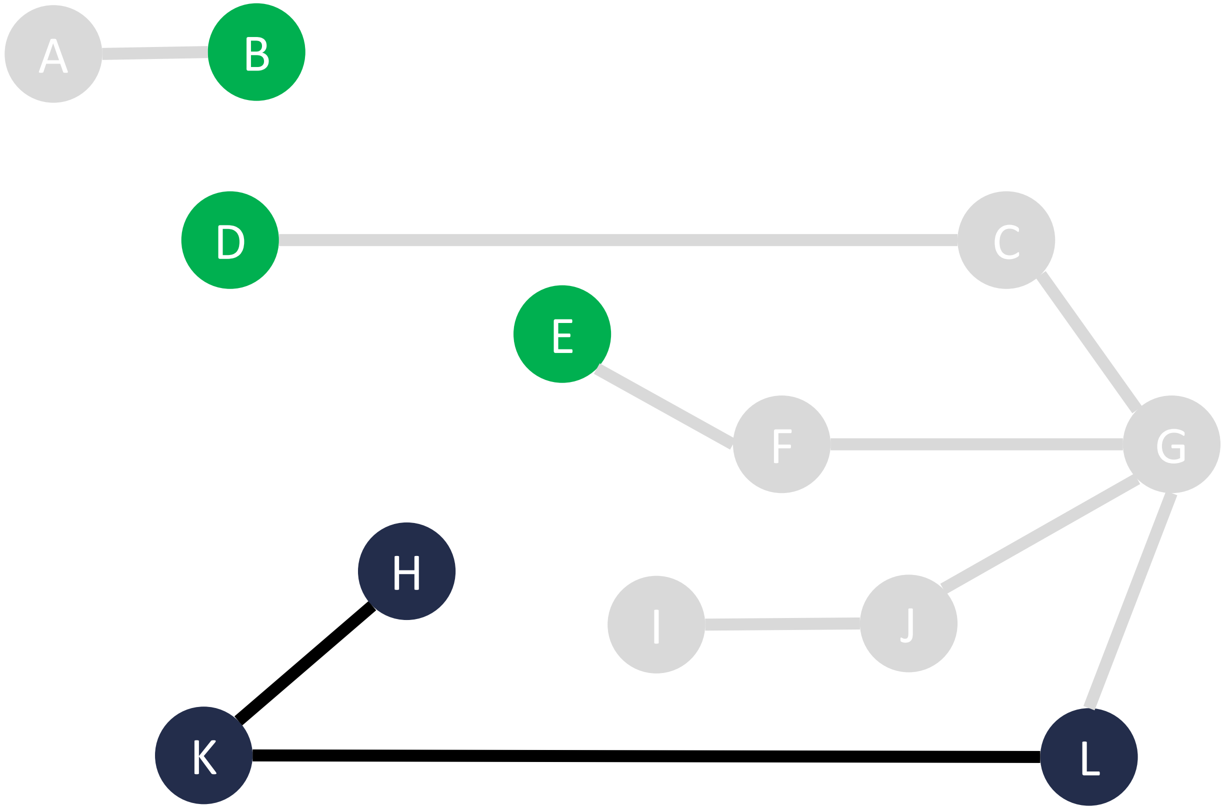
Run decision algorithm with $k = 3$

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 $k = 6$

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

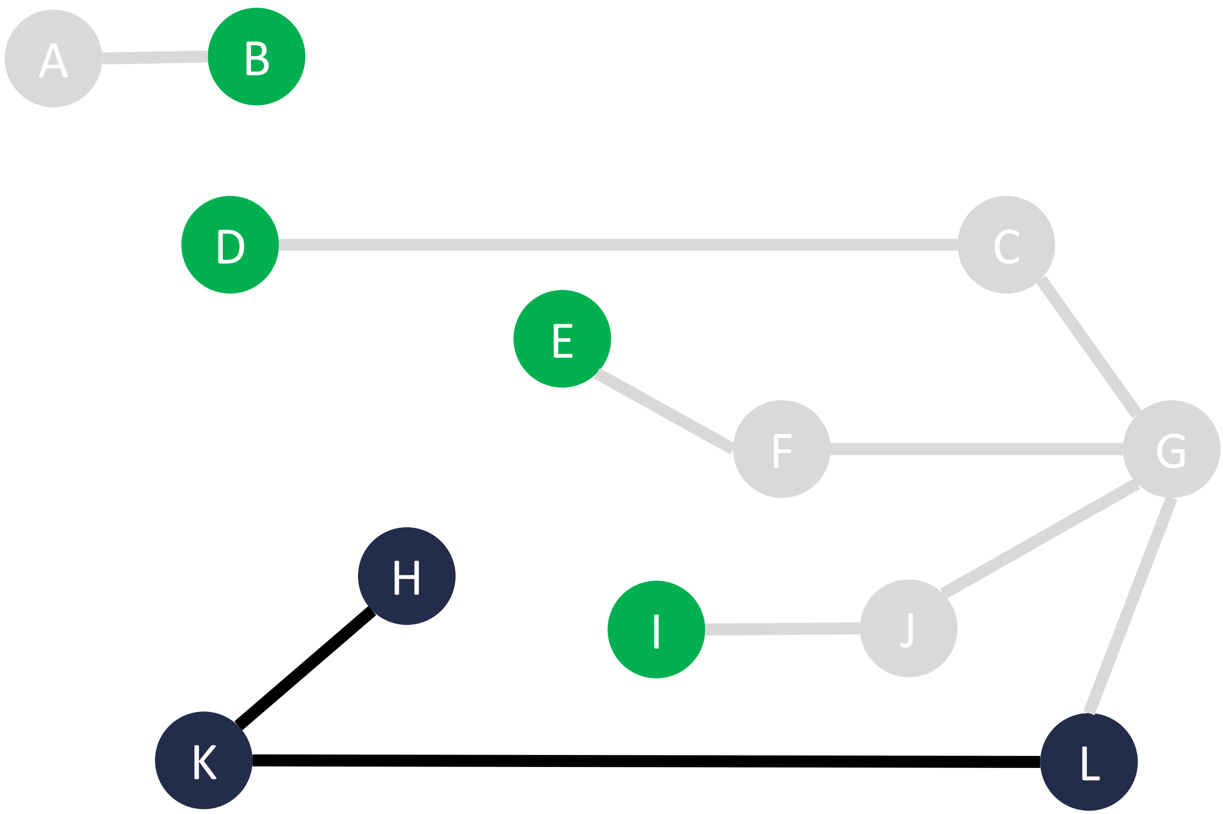
Run decision algorithm with $k = 3$

Output: False

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 $k = 6$

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

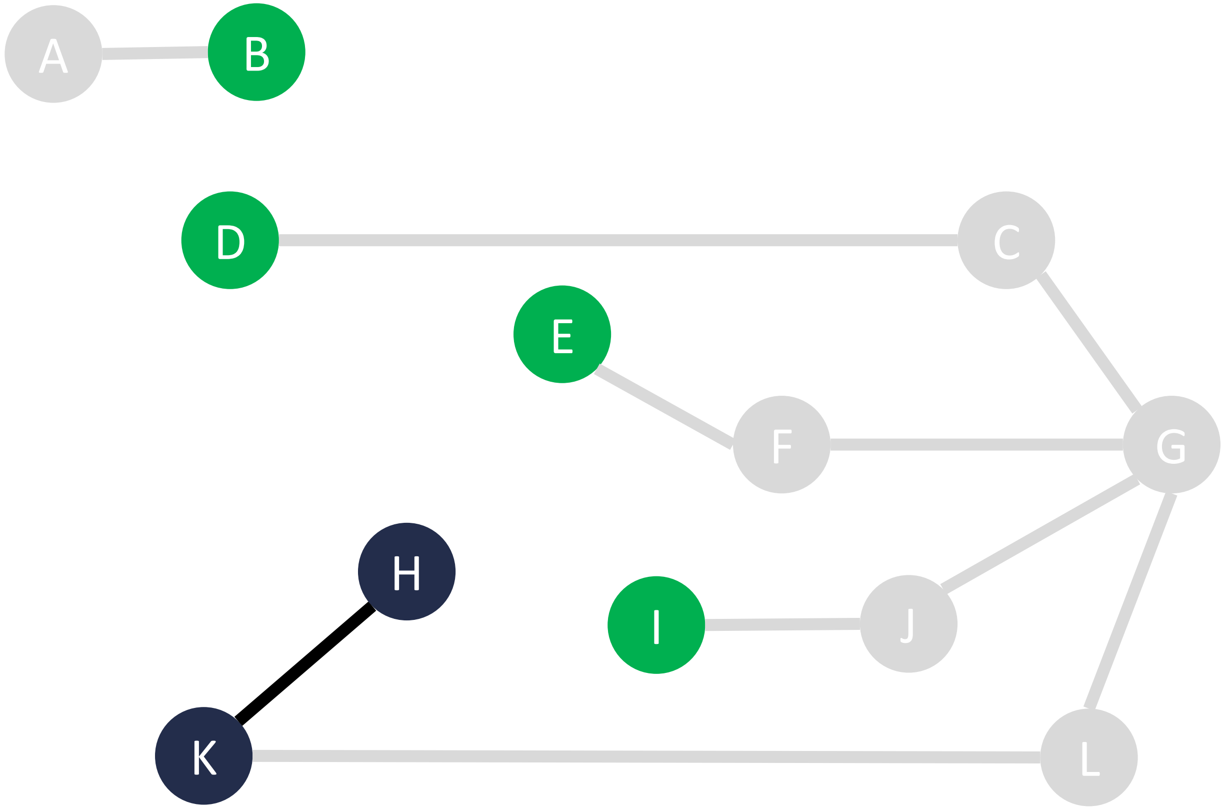
Run decision algorithm with $k = 2$

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 $k = 6$

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

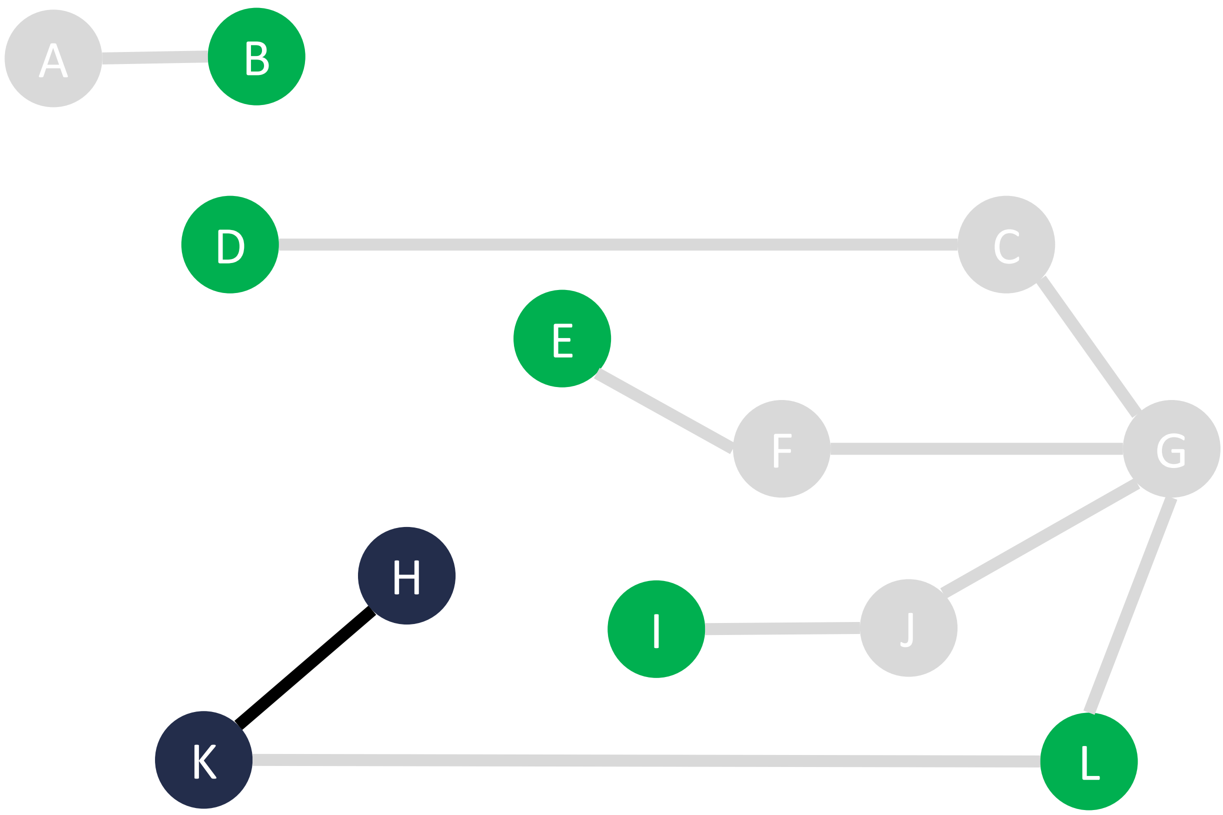
Run decision algorithm with $k = 2$

Output: False

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 $k = 6$

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

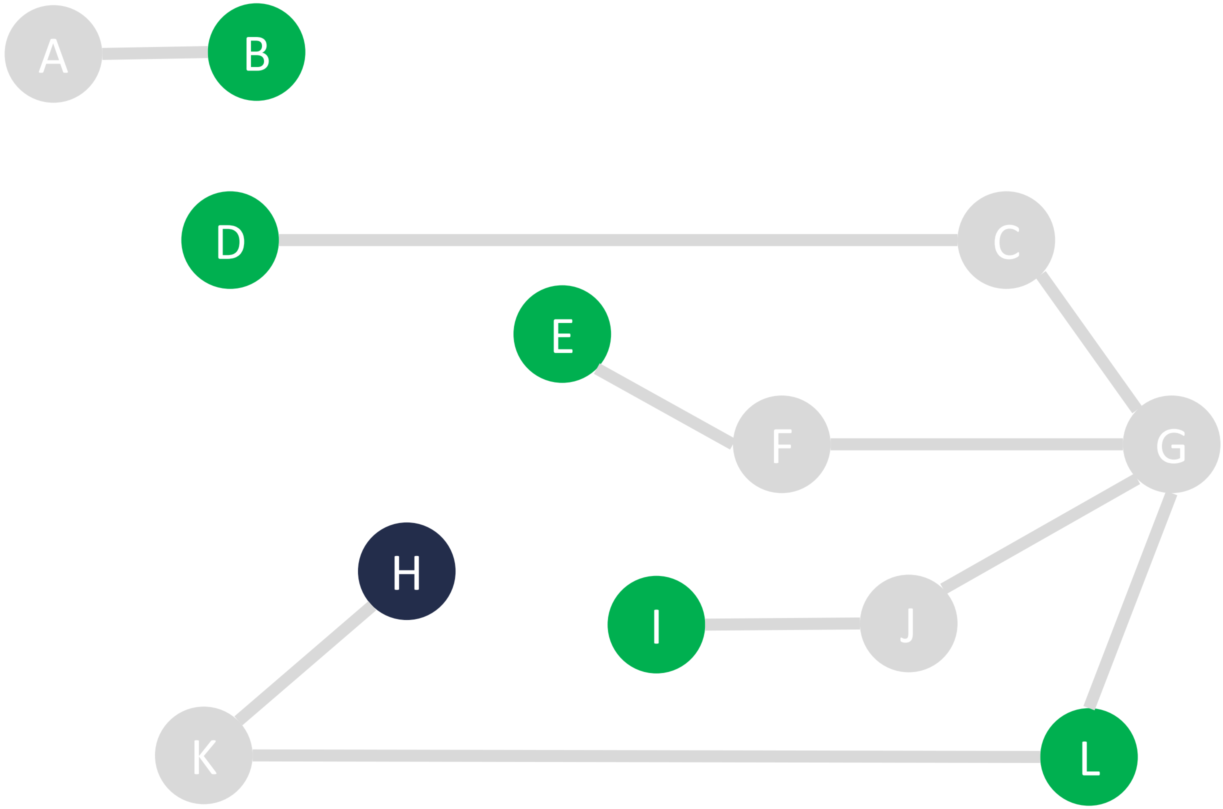
Run decision algorithm with $k = 1$

Output: 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 $k = 6$

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

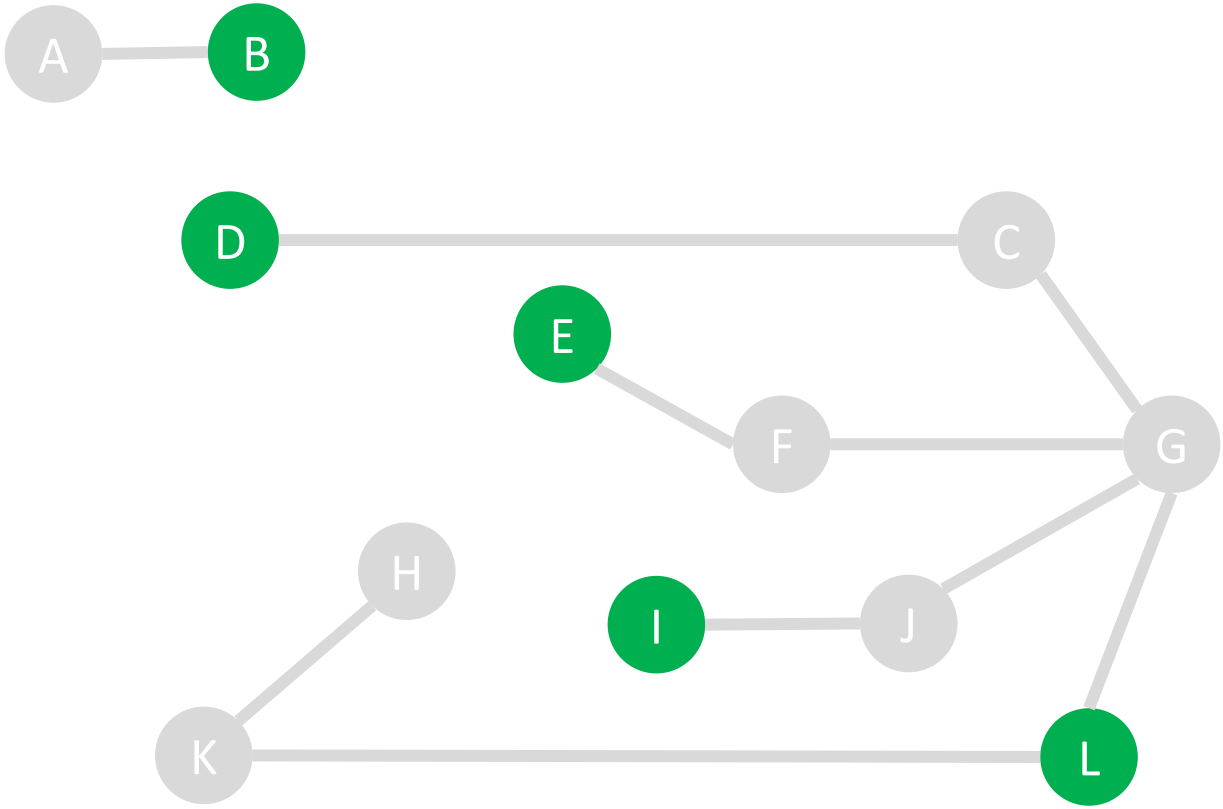
Run decision algorithm with $k = 1$

Output: 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 $k = 6$

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

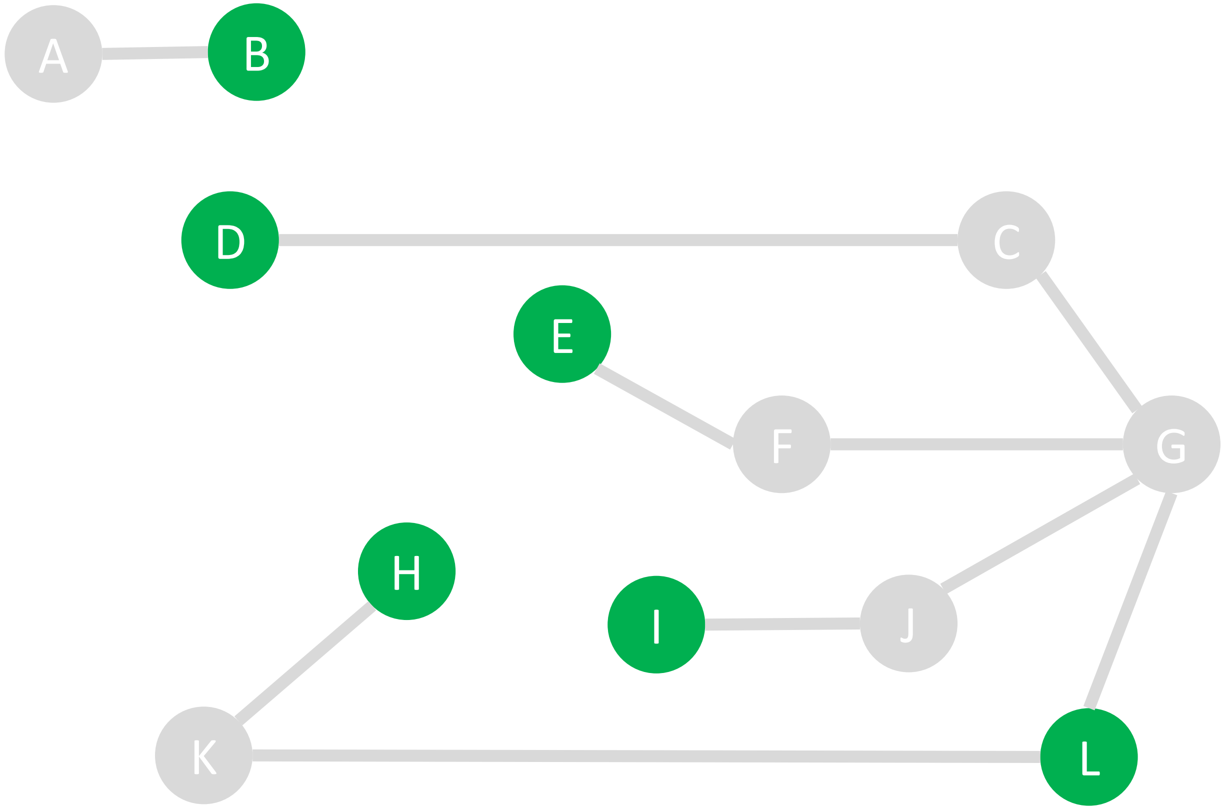
Run decision algorithm with $k = 1$

  Output: False

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 $k = 6$

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

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

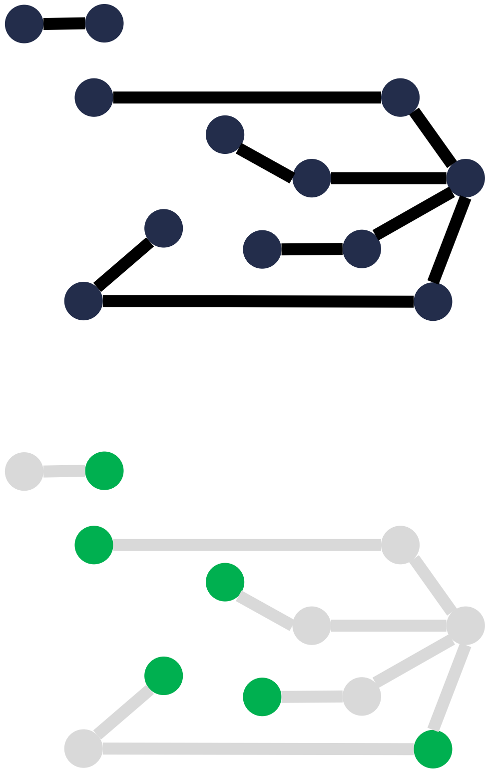
Cost of reduction: $O(|V| + |E|)$

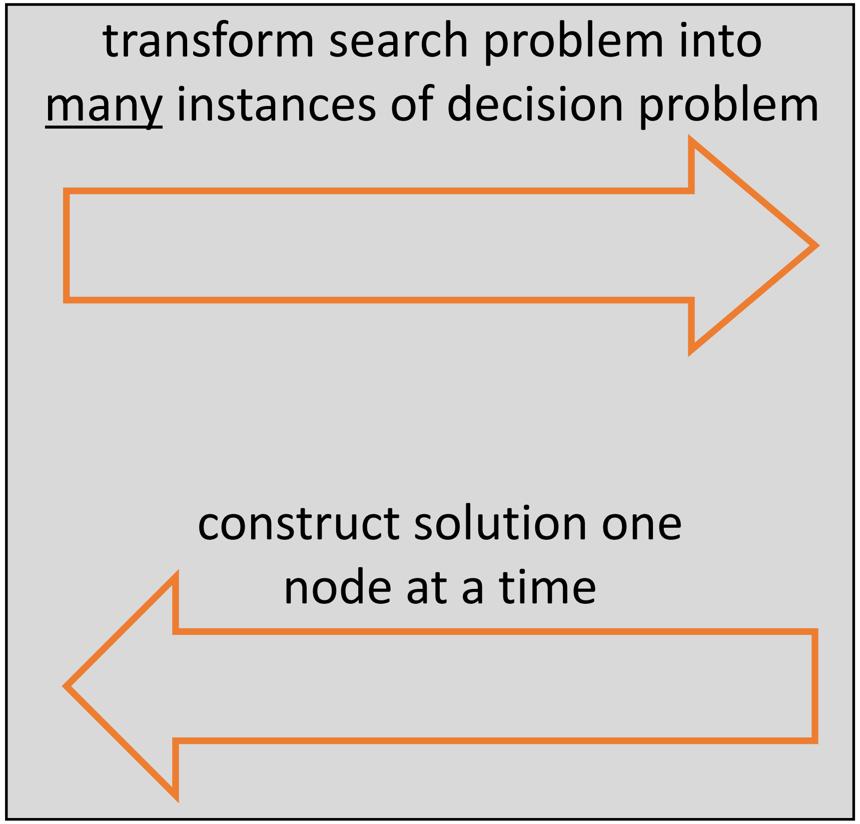
Search-to-Decision Reduction

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

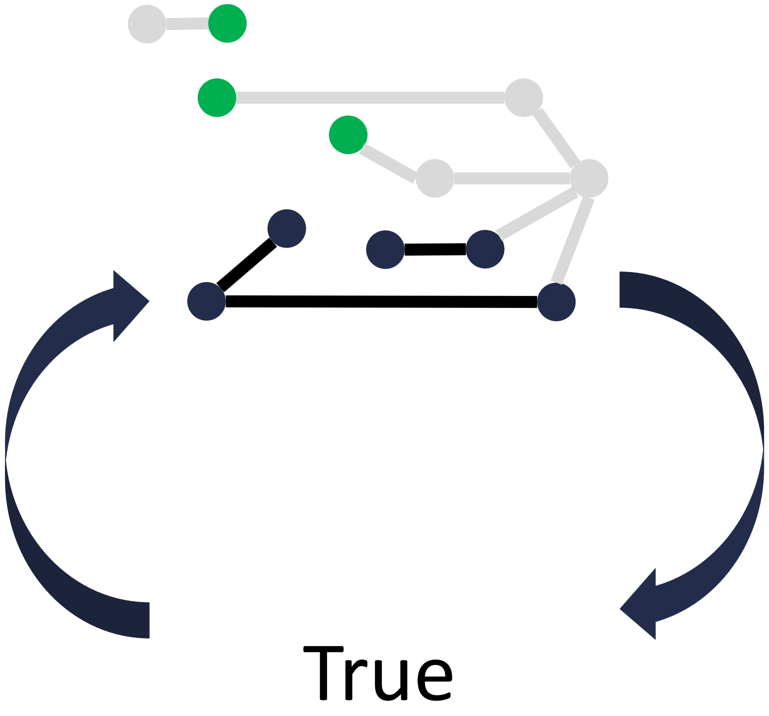
Problem (Search): Given a graph $G = (V, E)$, find an independent set with size k.

search





decision

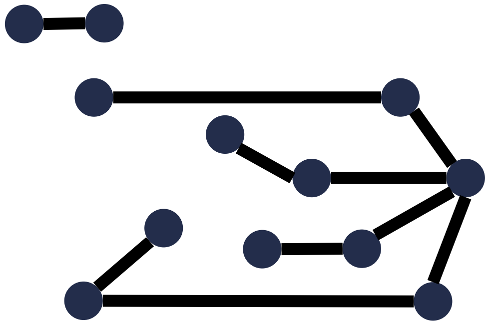


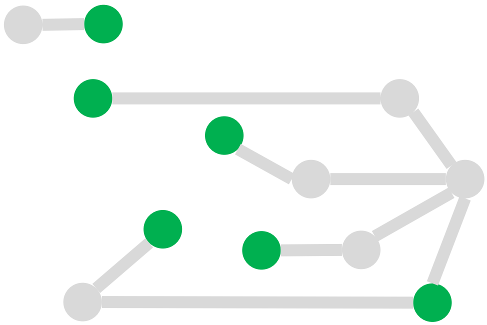
Search-to-Decision Reduction

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



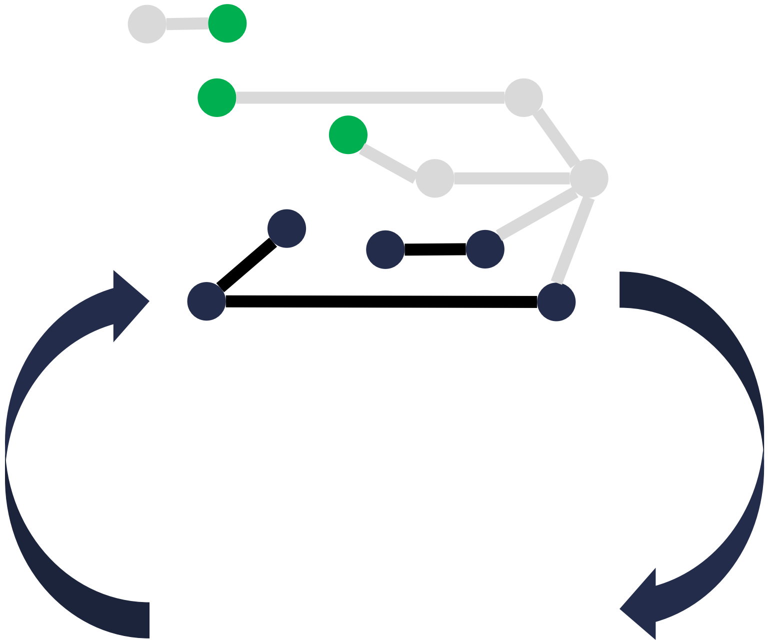


transform search problem into
<u>many</u> instances of decision problem

oftentimes called a
"<u>self-reduction</u>"

construct solution one
node at a time

decision



True

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"*

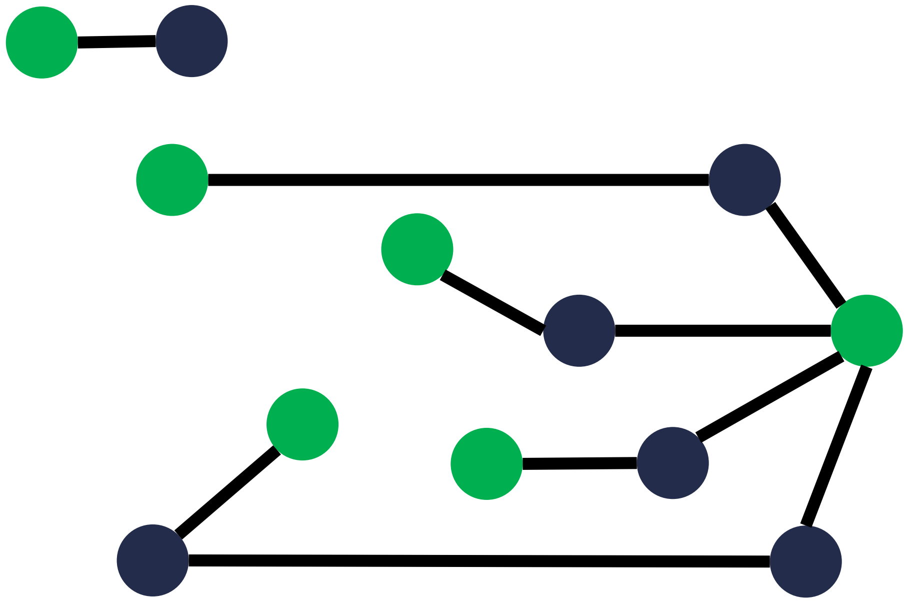
Decision problems: *"Does a solution to the problem exist?"*

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

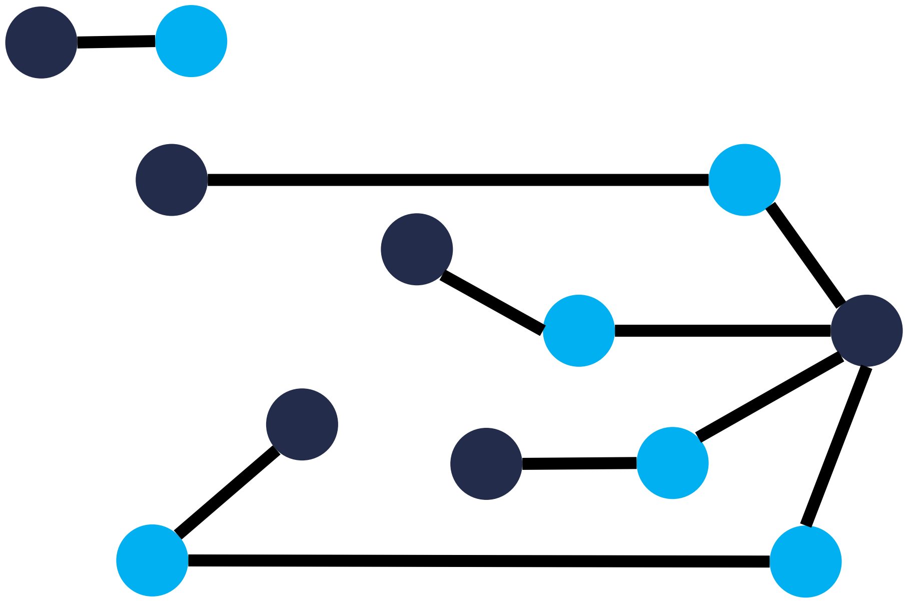
This is a key reason why we focus on decision problems rather than search problems

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: $O(|E| + |V|)$

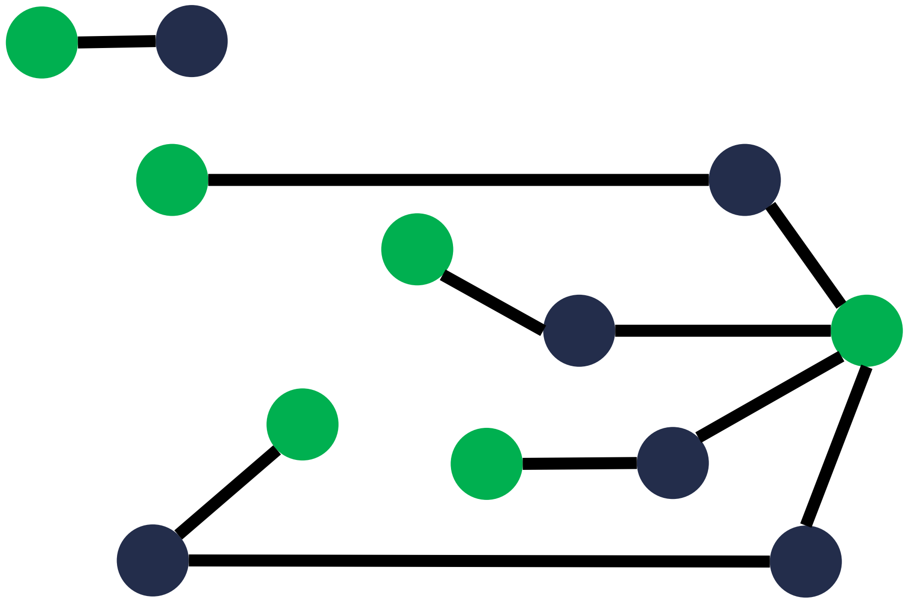


Given a graph G and a set S, it is easy to check if S is a k-vertex cover

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

For both of these problems, it is easy to check a candidate solution

The Class NP



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

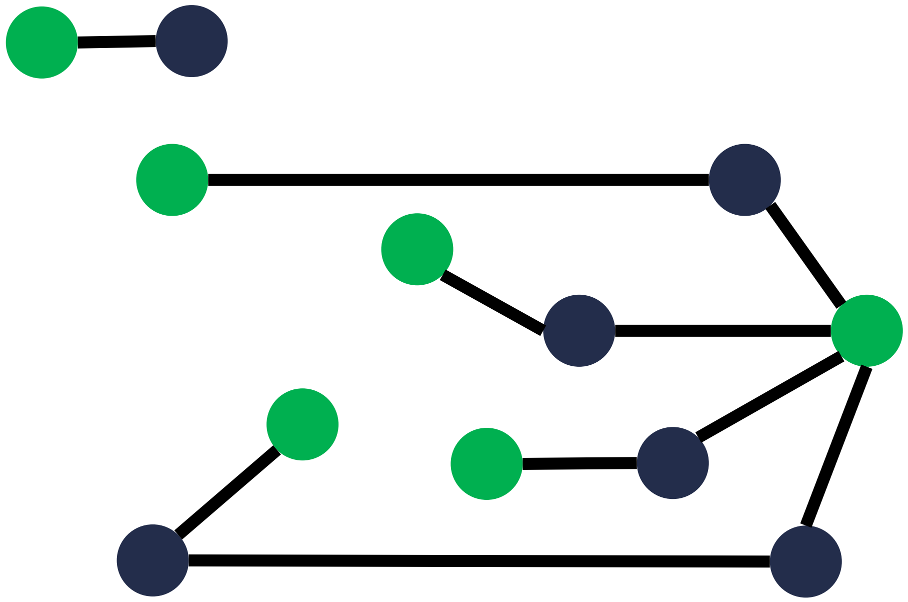
Running Time: $O(|E| + |V|)$

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:
    $\mathcal{L} = \{G: G \text{ has an independent set of size } k\}$
  - 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 $\mathcal{L} \in \mathrm{NP}$ if there exists a deterministic polynomial-time algorithm $\mathcal{R}$ such that

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

The Class NP



Given a graph G and a set S, it is easy t
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 $k\}$

    $k\}$

    ding

    is

    f $x \in \mathcal{L}$)

    inistic

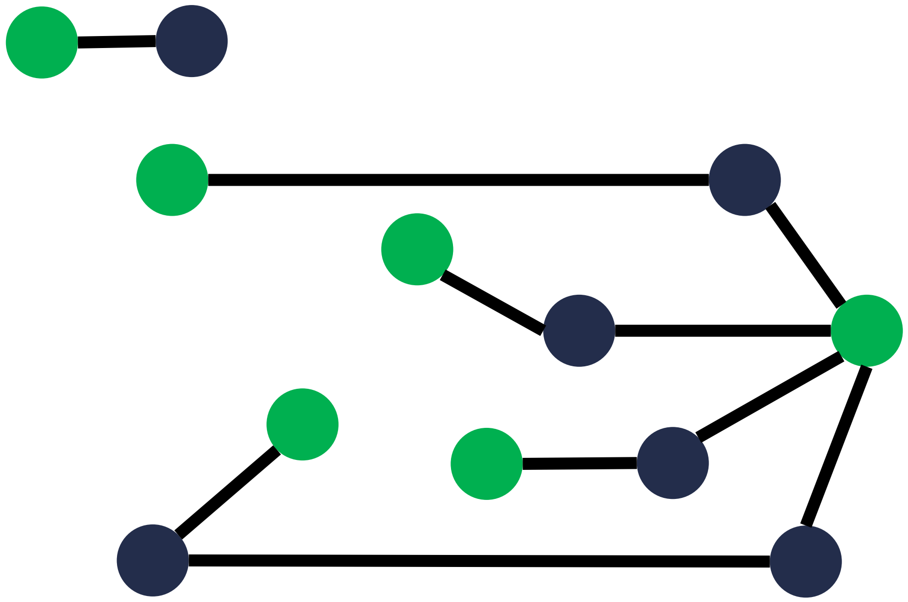
  pol al-time algorithm $\mathcal{R}$ such that

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

x is a statement and $\mathcal{L}$ is the language:

- x could be the description of a graph G
- $\mathcal{L}$ could be the set of graphs with a k-independent set
- Deciding whether $x \in \mathcal{L}$ is deciding whether G has a k-independent set or not

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: $O(|E| + |V|)$

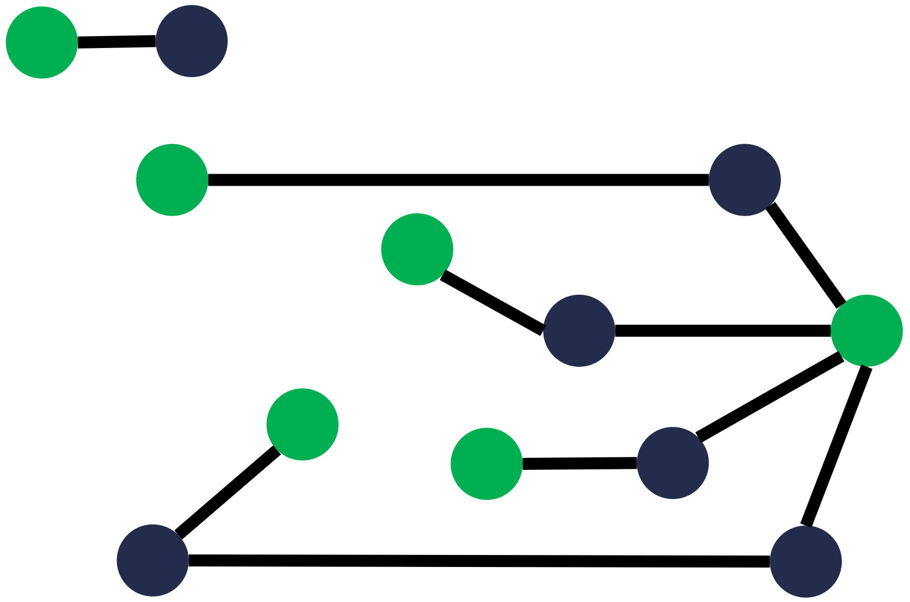
Complexity class NP:

- 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:
    $\mathcal{L} = \{G : G \text{ has an independent set of size } k\}$
  - k-vertex cover:
- S … ng v … c … $x \in \mathcal{L}$)
- A … stic polynomial-time algorithm $\mathcal{R}$ such that

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

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

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:
    $\mathcal{L} = \{G : G \text{ has an independent set of size } k\}$
  - k-vertex cover:
- S
  v
  c
- A
  polynomial time algorithm $\mathcal{R}$ su

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

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

$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathrm{poly}(|x|)}\colon \mathcal{R}(x, w) = 1$$

NP is the class of decision problems with <u>efficiently-verifiable</u> solutions

- Does not say anything about being able to <u>find</u> the solutions (i.e., we do not require that there is a polynomial-time algorithm to <u>find</u> 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 $\mathcal{L} \in \mathrm{P}$ if there exists a <u>deterministic</u> polynomial-time algorithm $\mathcal{R}$ such that

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

The Class P

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

$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathrm{poly}(|x|)}\colon \mathcal{R}(x, w) = 1$$

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 there is a polynomial-time algorithm that co

Polynomial in the input length (i.e., $\mathrm{poly}(|x|) = O\left(|x|^d\right)$ for some $d \in \mathbb{N}$

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

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

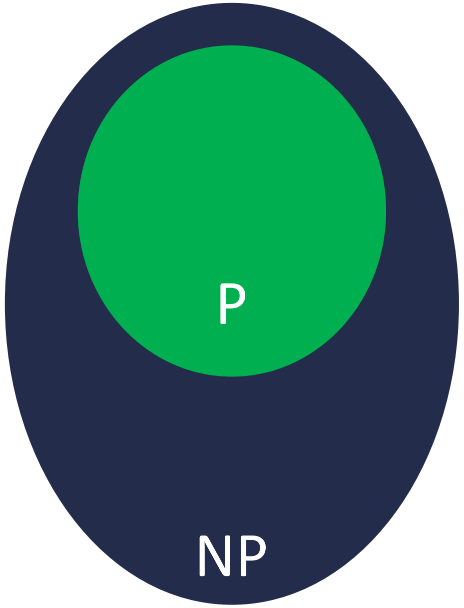
P vs. NP

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

$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathrm{poly}(|x|)}\colon \mathcal{R}(x, w) = 1$$

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

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



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

$$\mathrm{P} \subseteq \mathrm{NP}$$

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

$$\mathrm{P} = \mathrm{NP} \text{ or } \mathrm{P} \neq \mathrm{NP}$$

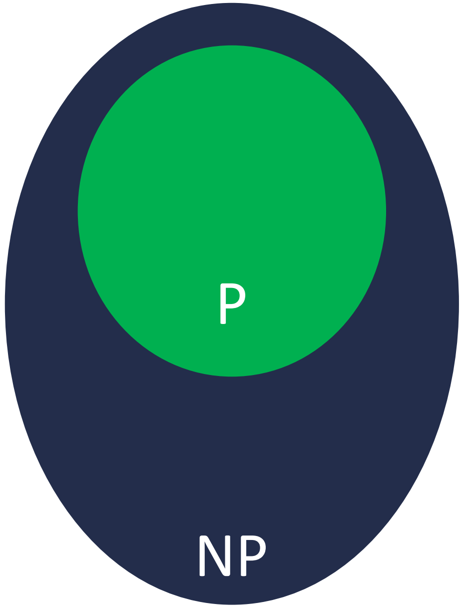
P vs. NP

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

$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathrm{poly}(|x|)} \colon \mathcal{R}(x, w) = 1$$

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

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



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

$$\mathrm{P} \subseteq \mathrm{NP}$$

One of the seven Millennium Prize problems!

$\mathrm{P} = \mathrm{NP}$ or $\mathrm{P} \neq \mathrm{NP}$

P vs. NP

A langua

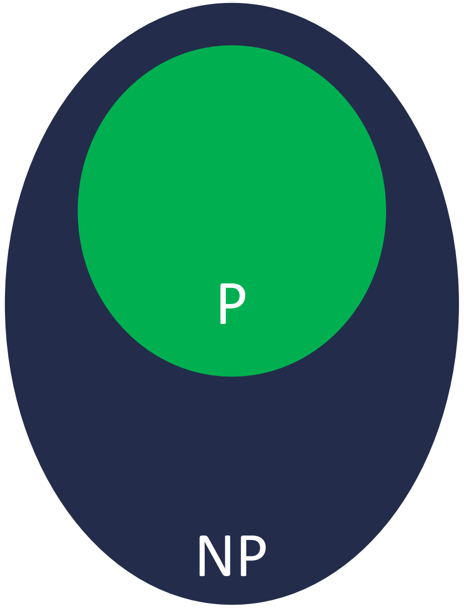
P: “polynomial time”

eterministic polynomial-time “verifier” $\mathcal{R}$ such that

$v \in \{0,1\}^{\mathrm{poly}(|x|)}$: $\mathcal{R}(x, w) = 1$

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

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



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

$$\mathrm{P} \subseteq \mathrm{NP}$$

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

$$\mathrm{P} = \mathrm{NP} \text{ or } \mathrm{P} \neq \mathrm{NP}$$

P vs. NP

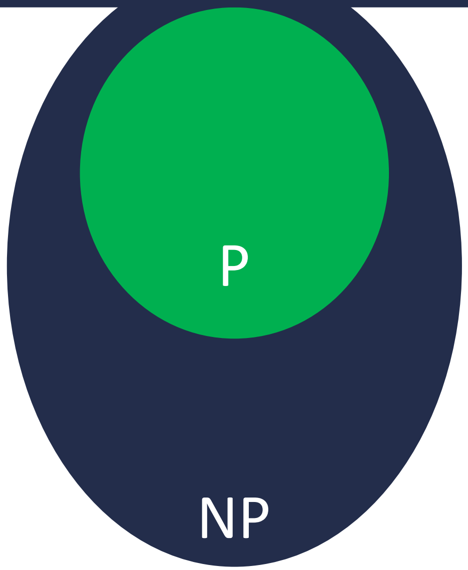
A language $\mathcal{L} \in \mathrm{NP}$ if there exists a deterministic polynomial-time "verifier" $\mathcal{R}$ such that

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

NP: "<u>non-deterministic</u> polynomial time"

- Non-deterministic has <u>nothing</u> 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 <u>polynomial time</u>)

A lan … hat



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

$$\mathrm{P} \subseteq \mathrm{NP}$$

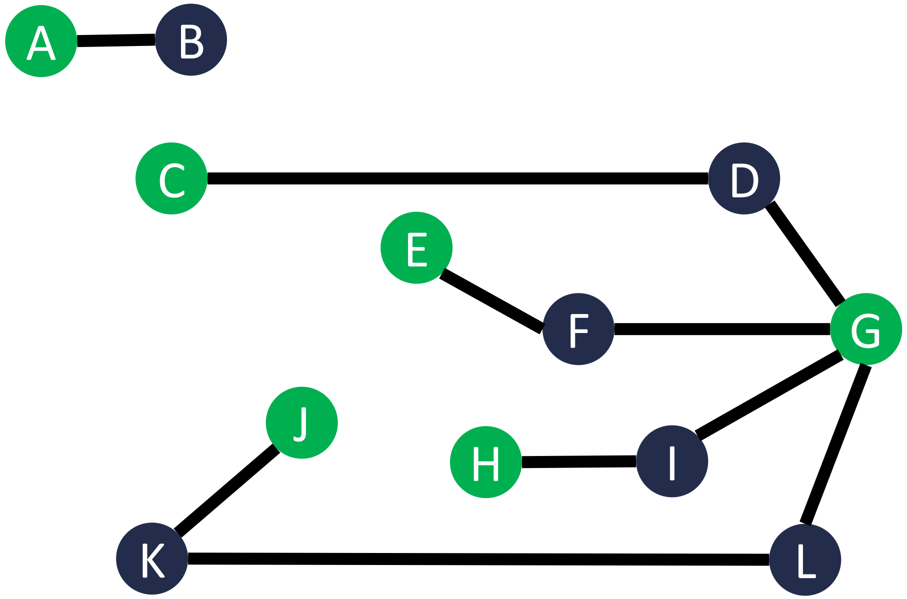
Biggest open problem in computer science: is this containment strict?

$$\mathrm{P} = \mathrm{NP} \text{ or } \mathrm{P} \neq \mathrm{NP}$$

k-Independent Set is in 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)



Graph G

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

Checking the witness:

- Check that $|S| = k$ $\quad O(k) = O(|V|)$
- Check that every edge is incident on at most one node in S $\quad O(|V| + |E|)$

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