Reduction of NP Problems
&
Property-Based Testing

Chenhao Zhang

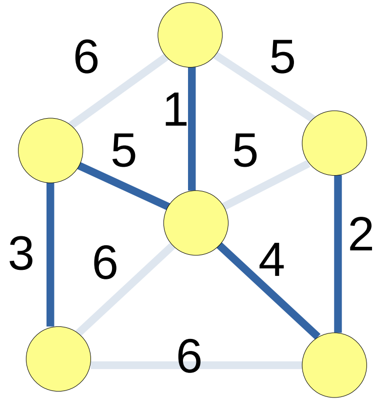
CS396 Spring 2023
Northwestern

Plan of the week

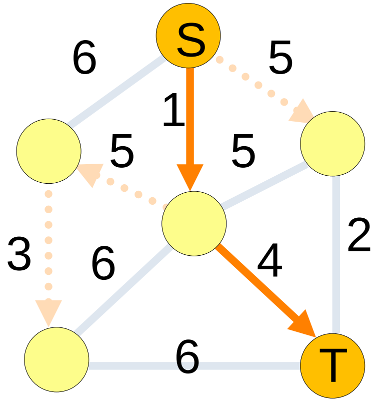
- **NP Problem & Reduction (Today)**
- Examples, Reduction in Karp -- Wednesday
- Lab, Assignment 4 -- Friday

Many problems have efficient algorithms

Minimum Spanning Tree



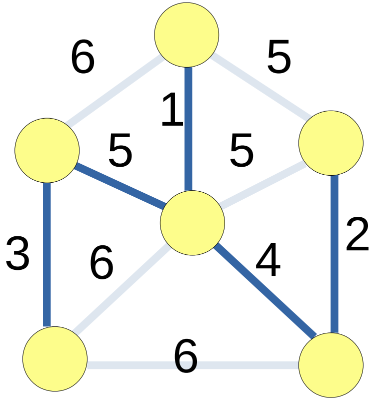
Shortest path



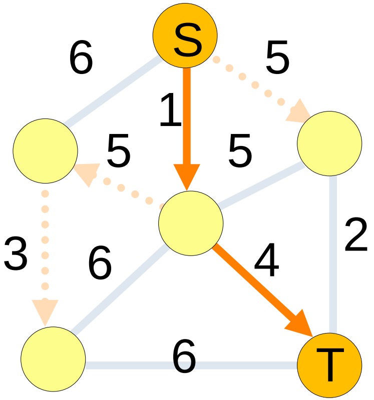
……

Many problems have efficient algorithms

__Minimum__ Spanning Tree



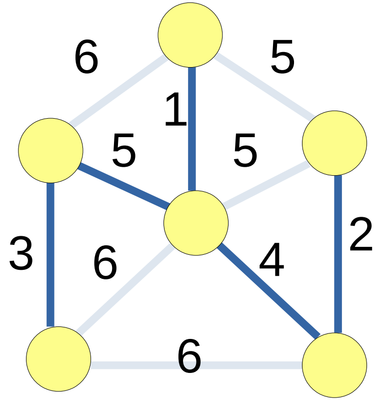
__Shortest__ path



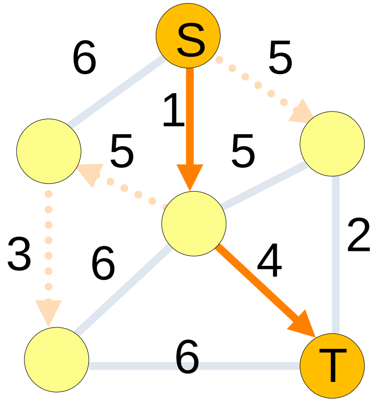
……

version with Yes/No answer

Has Spanning Tree w/ Cost <=15 ?



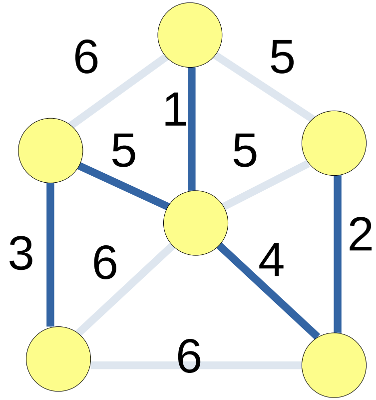
Has S-T path w/ Cost <=5 ?



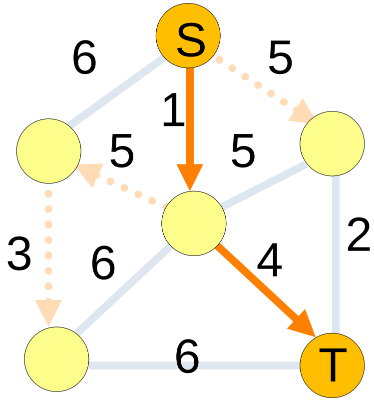
……

version with Yes/No answer – *decision problem*

Has Spanning Tree w/ Cost <=15 ?



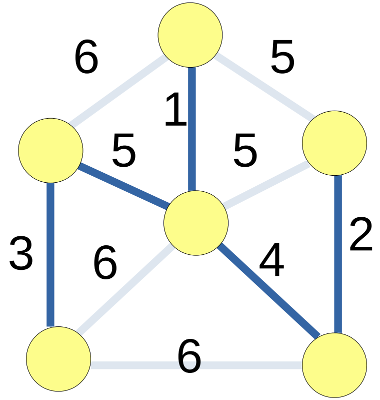
Has S-T path w/ Cost <=5 ?



......

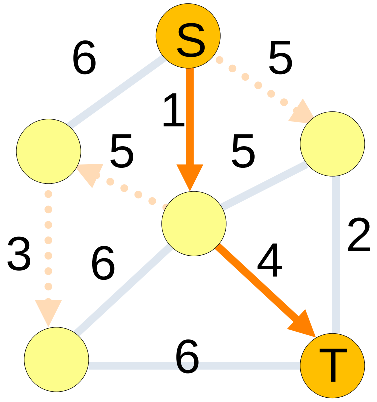
version with Yes/No answer – *decision problem*

Has Spanning Tree w/ Cost <=15 ?



1+5+3+4+2=15

Has S-T path w/ Cost <=5 ?

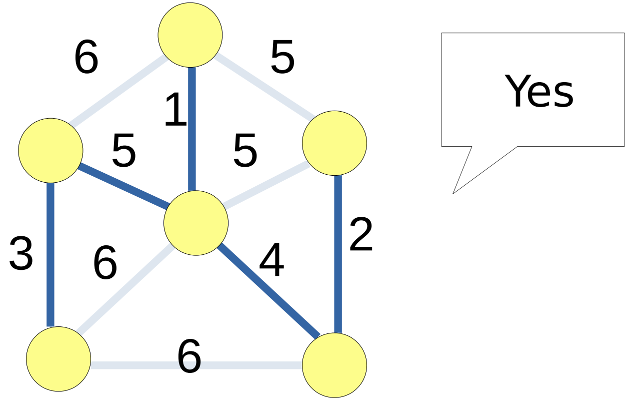


1+4=5

……

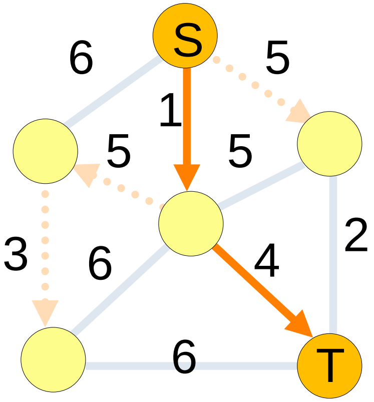
version with Yes/No answer – *decision problem*

Has Spanning Tree w/ Cost <=15 ?



1+5+3+4+2=15

Has S-T path w/ Cost <=5 ?

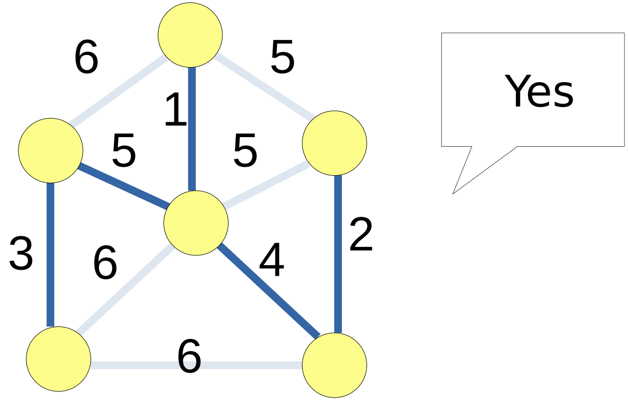


1+4=5

……

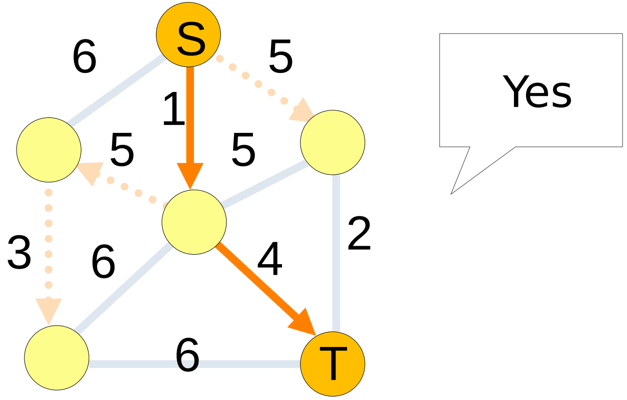
version with Yes/No answer – *decision problem*

Has Spanning Tree w/ Cost <=15 ?



1+5+3+4+2=15

Has S-T path w/ Cost <=5 ?

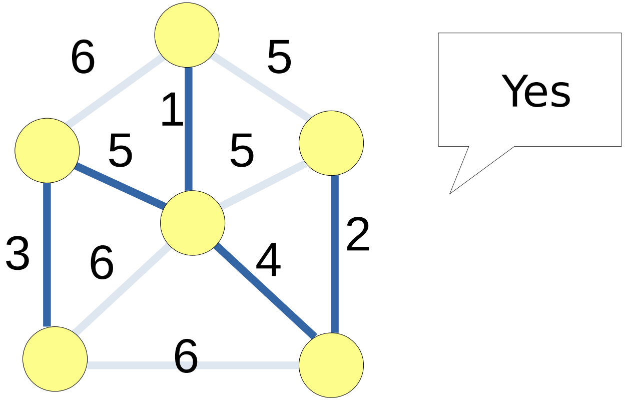


1+4=5

……

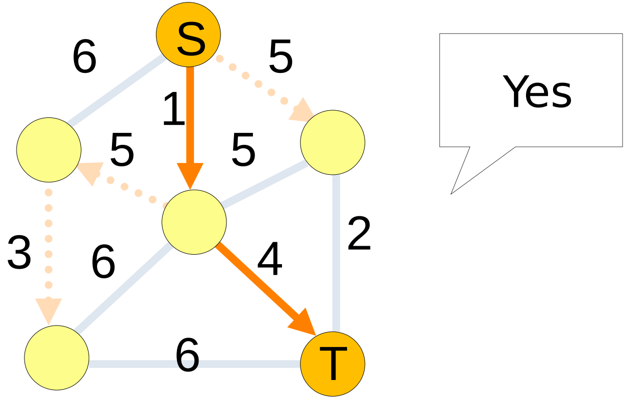
Yes-Instance has a *certificate*, i.e., proof of yes

Has Spanning Tree w/ Cost <=15 ?



1+5+3+4+2=15

Has S-T path w/ Cost <=5 ?

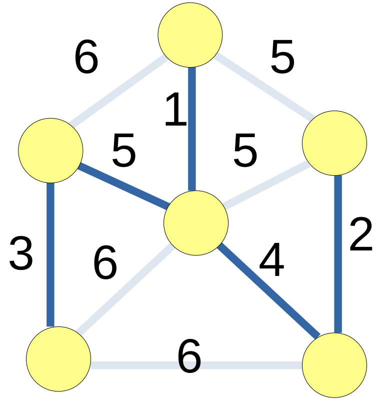


1+4=5

……

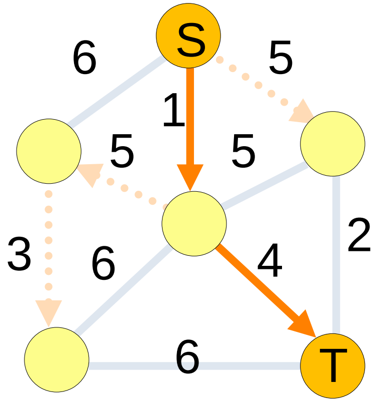
No-Instance has no *certificate*, ~~proof of yes~~

Has Spanning Tree w/ Cost <=**14** ?



1+5+3+4+2=15 > 14

Has S-T path w/ Cost <=**4** ?

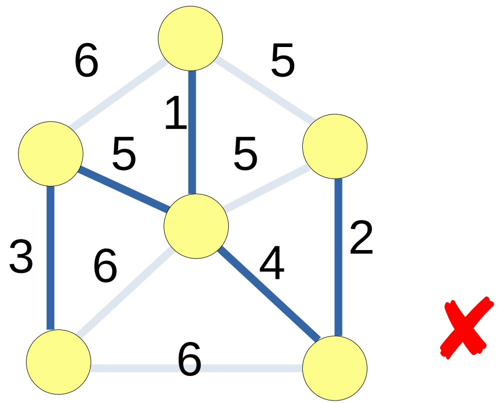


1+4=5 > 4

……

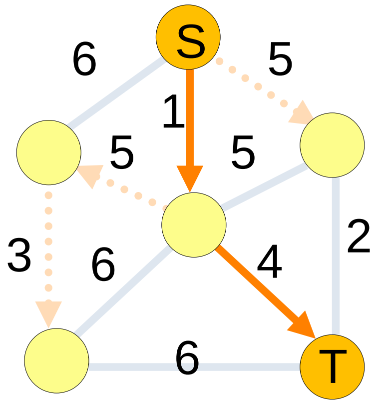
No-Instance has no *certificate*, ~~proof of yes~~

Has Spanning Tree w/ Cost <=**14** ?



1+5+3+4+2=15 > 14

Has S-T path w/ Cost <=**4** ?

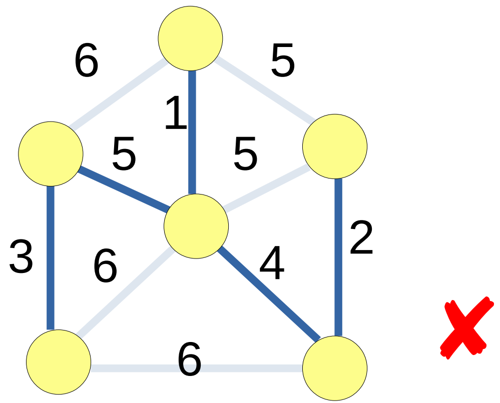


1+4=5 > 4

……

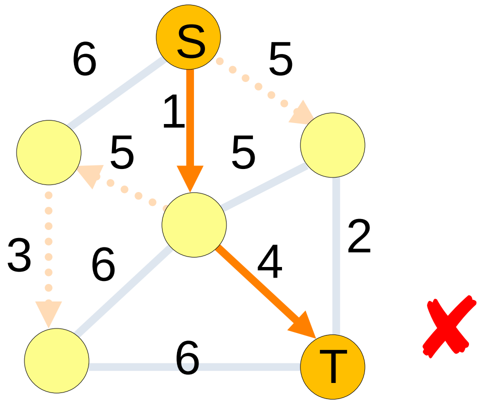
No-Instance has no *certificate*, ~~proof of yes~~

Has Spanning Tree w/ Cost <=**14** ?



1+5+3+4+2=15 > 14

Has S-T path w/ Cost <=**4** ?

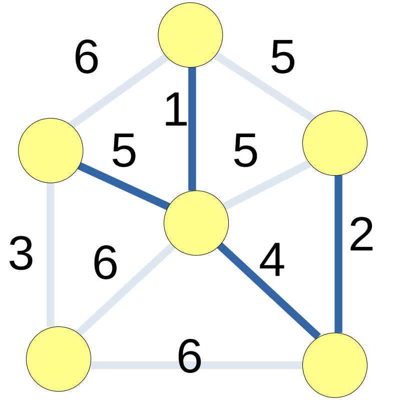


1+4=5 > 4

……

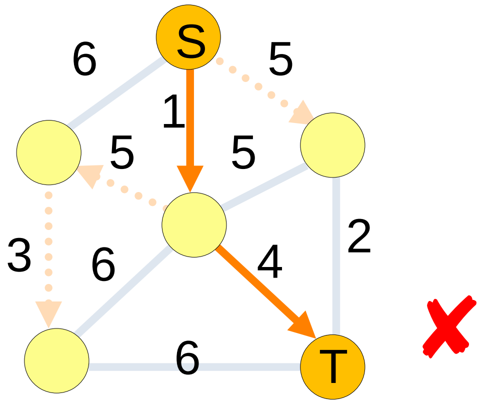
No-Instance has no *certificate*, ~~proof of yes~~

Has Spanning Tree w/ Cost <=**14** ?



1+5+4+2=12 <= 14

Has S-T path w/ Cost <=**4** ?

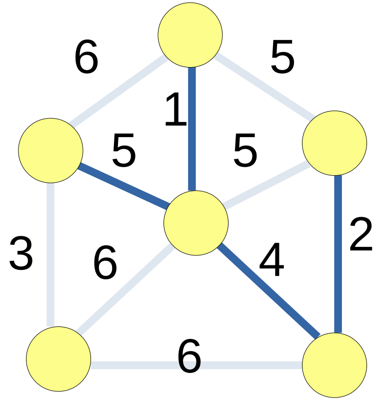


1+4=5 > 4

……

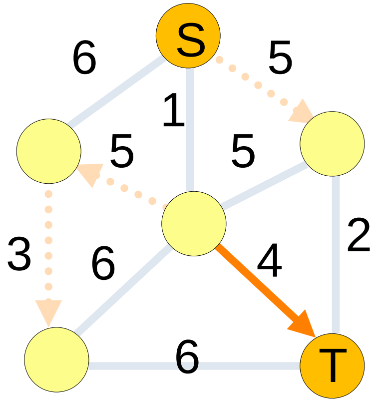
No-Instance has no *certificate*, ~~proof of yes~~

Has Spanning Tree w/ Cost <=**14** ?



1+5+4+2=12 <= 14

Has S-T path w/ Cost <=**4** ?

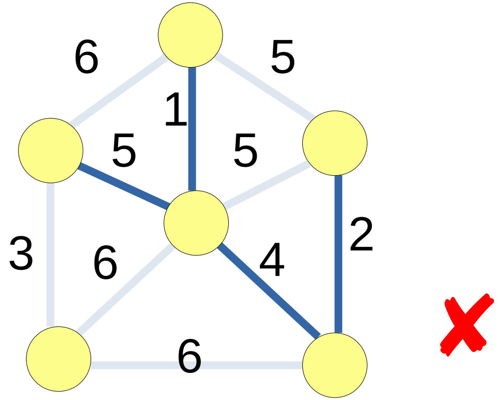


4=4 <= 4

……

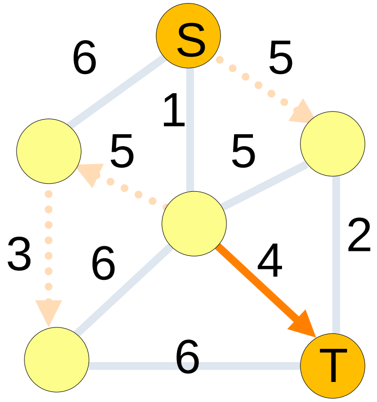
No-Instance has no *certificate*, ~~proof of yes~~

Has Spanning Tree w/ Cost <=**14** ?



1+5+4+2=12 <= 14

Has S-T path w/ Cost <=**4** ?

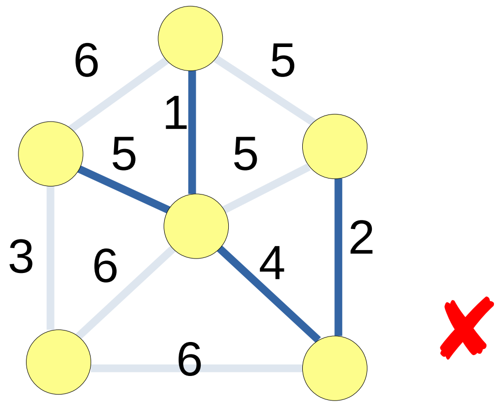


4=4 <= 4

……

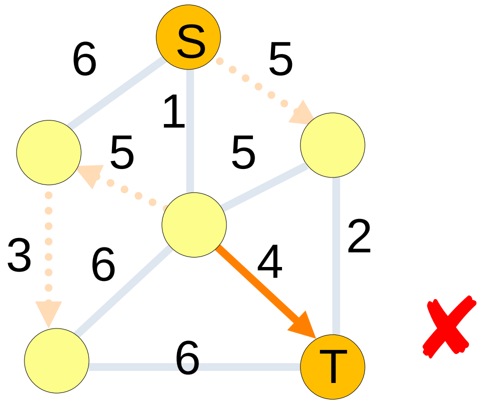
No-Instance has no *certificate*, ~~proof of yes~~

Has Spanning Tree w/ Cost <=**14** ?



1+5+4+2=12 <= 14

Has S-T path w/ Cost <=**4** ?

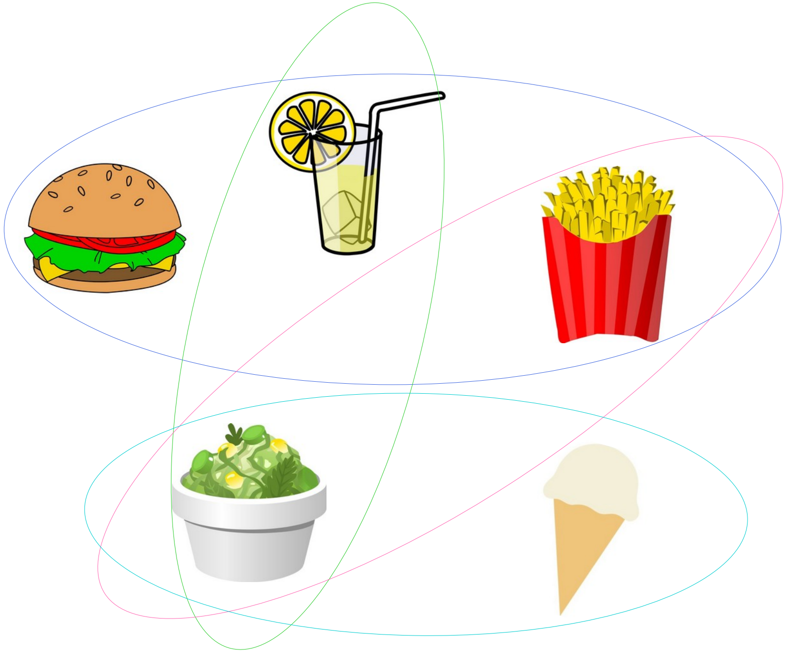


4=4 <= 4

……

There are also many other problems...

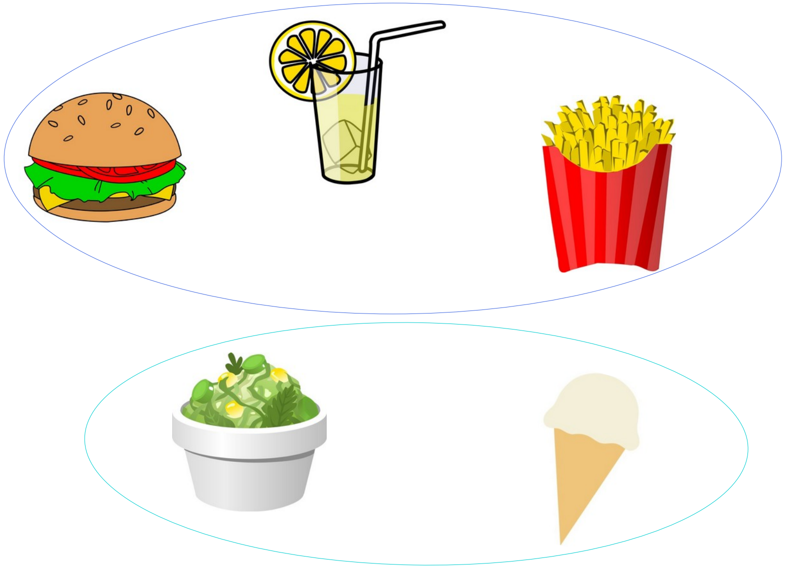
Can we get all by buying only **2** bundles?



SET-COVER

There are also many other problems...

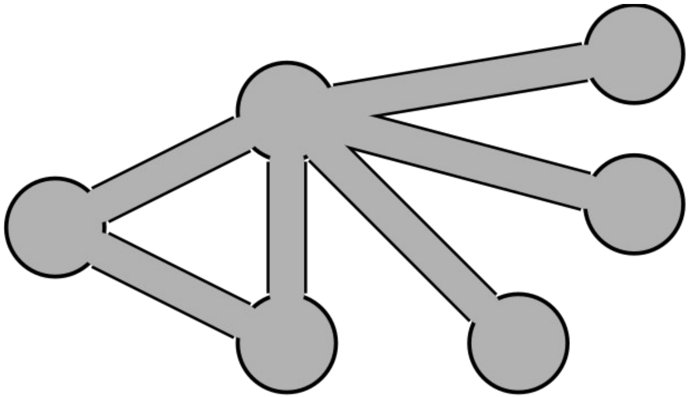
Can we get all by buying only **2** bundles?



SET-COVER

There are also many other problems...

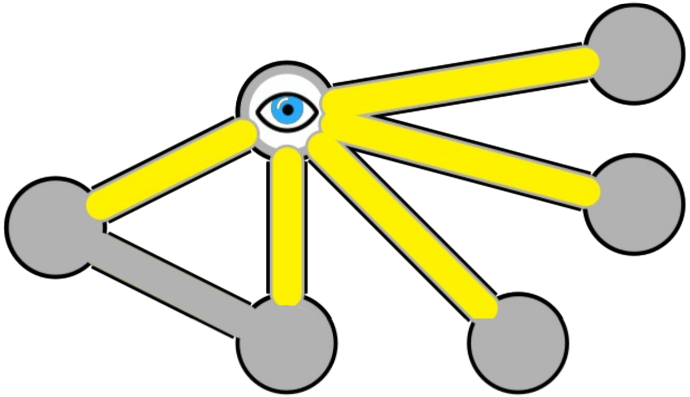
Can we watch all roads by setting only **2** sentry points?



VERTEX-COVER

There are also many other problems...

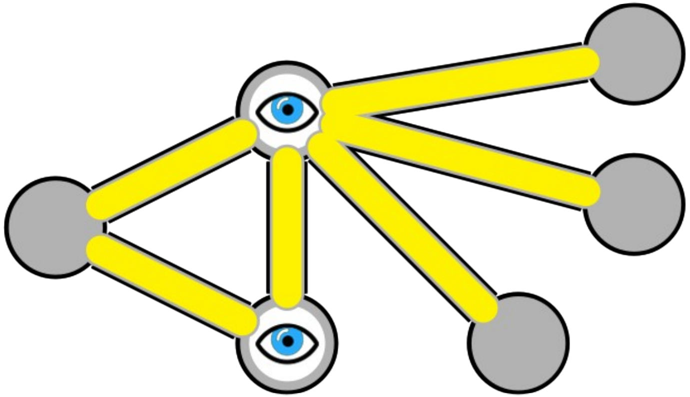
Can we watch all roads by setting only **2** sentry points?



VERTEX-COVER

There are also many other problems...

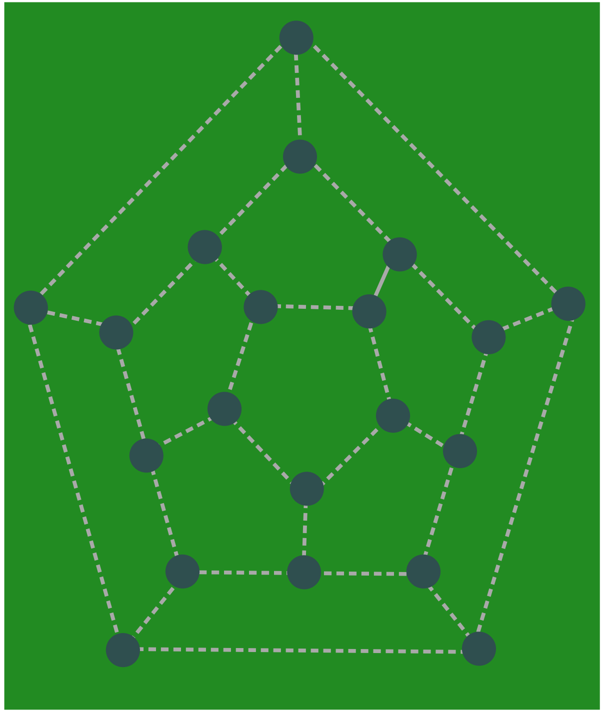
Can we watch all roads by setting only **2** sentry points?



VERTEX-COVER

There are also many other problems...

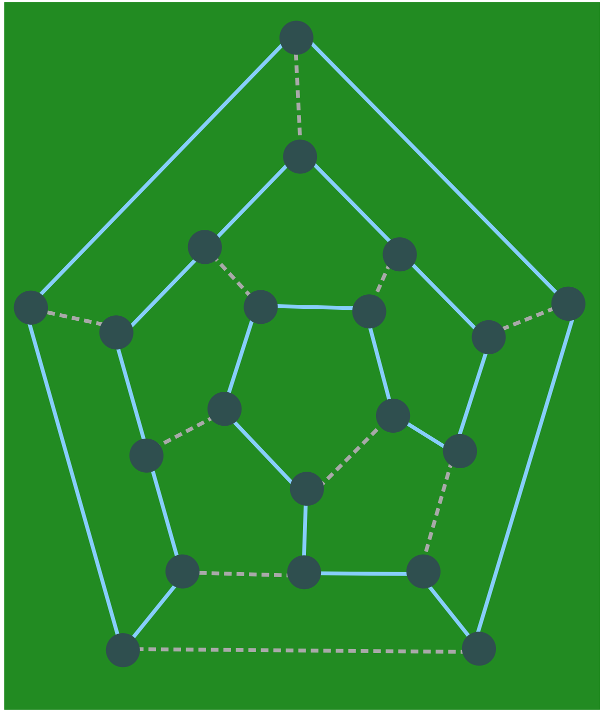
Is there a cycle that visits all vertices?



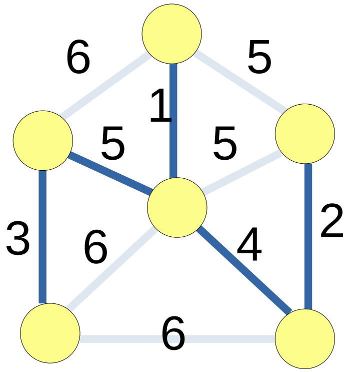
HAMILTONIAN-CYCLE

There are also many other problems...

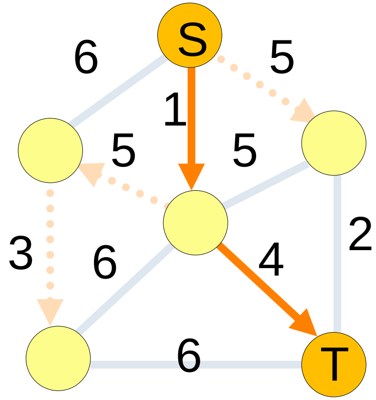
Is there a cycle that visits all vertices?



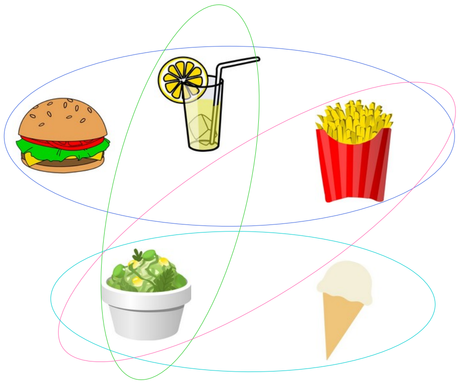
HAMILTONIAN-CYCLE



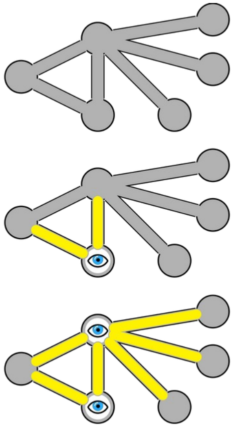
Minimum-Spanning-Tree



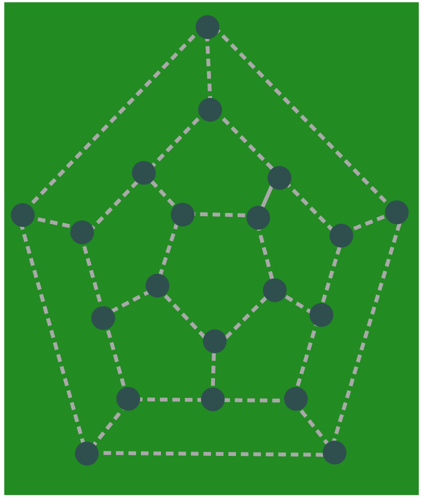
Shortest-Path



Set-Cover



Vertex-Cover



Hamiltonian-Cycle

Q: What do they have in common?

SET-COVER

VERTEX-COVER

HAMILTONIAN-CYCLE

Q: What do they have in common?

A: Validity of certificate EASY to check!
(can be done in polynomial-time)

HAMILTONIAN-CYCLE

Q: What do they have in common?

A: Validity of certificate EASY to check!
(can be done in polynomial-time)

$O(n)$ $O(n^2)$

Set-Cover Vertex-Cover Hamiltonian-Cycle

Q: What do they have in common?

A: Validity of certificate EASY to check!
(can be done in polynomial-time)

$O(n)$ $O(n^2)$ $O(n^{10^{10}})$

SET-COVER VERTEX-COVER HAMILTONIAN-CYCLE

Q: What do they have in common?

A: Validity of certificate EASY to check!
(can be done in polynomial-time)

$O(n)$ $O(n^2)$ $O(n^{10^{10}})$ ~~$O(1.01^n)$~~

SET-COVER VERTEX-COVER HAMILTONIAN-CYCLE

Q: What do they have in common?

A: Validity of certificate EASY to check!
(can be done in polynomial-time)

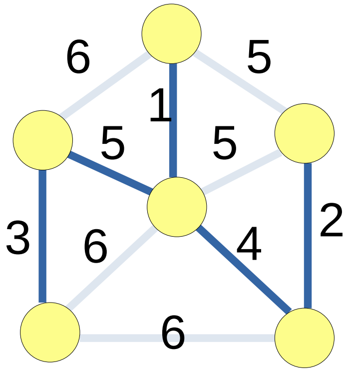
NP-Problems

(**N**on-deterministic **P**olynomial-time)

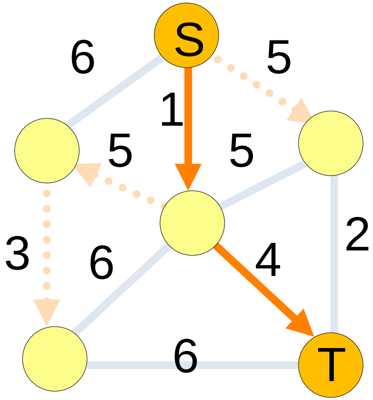
SET-COVER

VERTEX-COVER

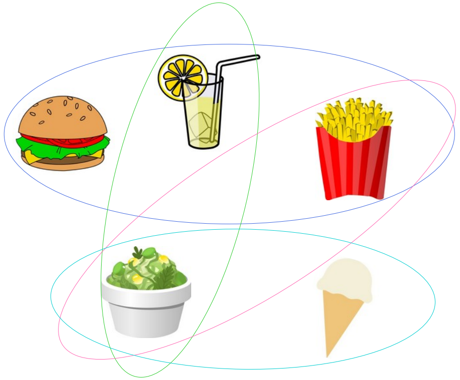
HAMILTONIAN-CYCLE



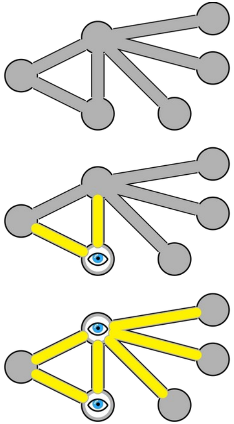
MINIMUM-SPANNING-TREE



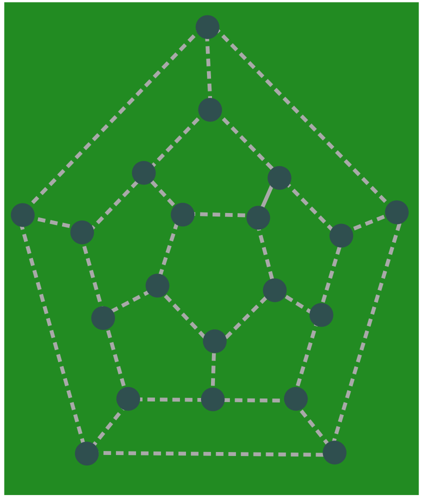
SHORTEST-PATH



SET-COVER



VERTEX-COVER



HAMILTONIAN-CYCLE

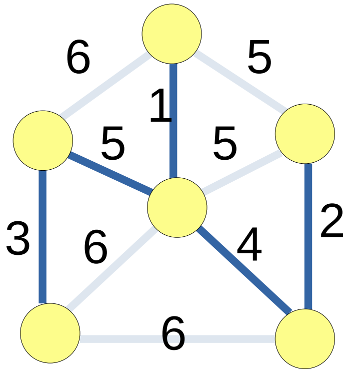
Q: Any difference?

SET-COVER

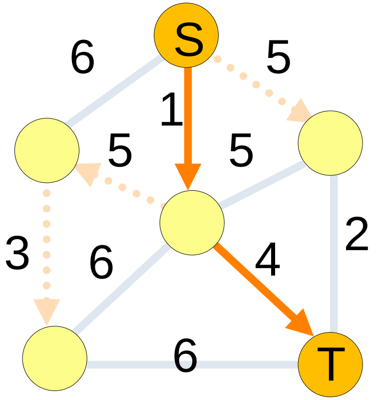
VERTEX-COVER

HAMILTONIAN-CYCLE

“Easy”

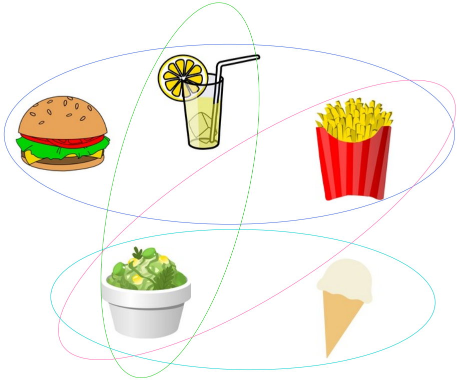


MINIMUM-SPANNING-TREE

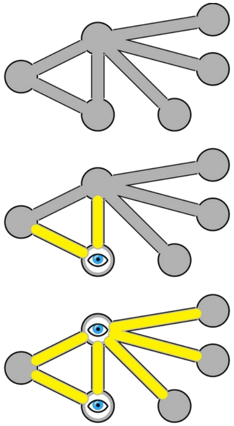


SHORTEST-PATH

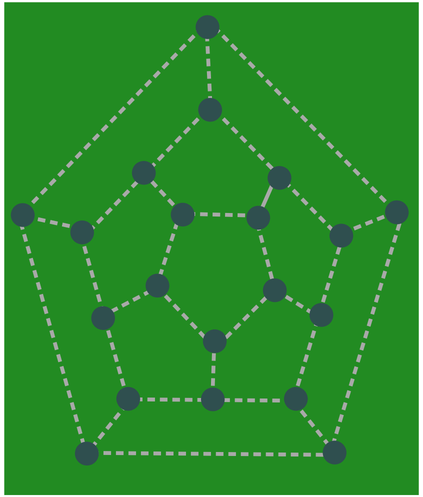
“Hard”



SET-COVER



VERTEX-COVER



HAMILTONIAN-CYCLE

Q: Any difference?

A: It is generally believed that:

“Hard” problems have NO efficient algorithms

HAMILTONIAN-CYCLE

Q: Any difference?

A: It is generally believed that:

“Hard” problems have NO efficient algorithms

But there’s no proof for it yet...

Set-Cover Vertex-Cover Hamiltonian-Cycle

Q: Any difference?

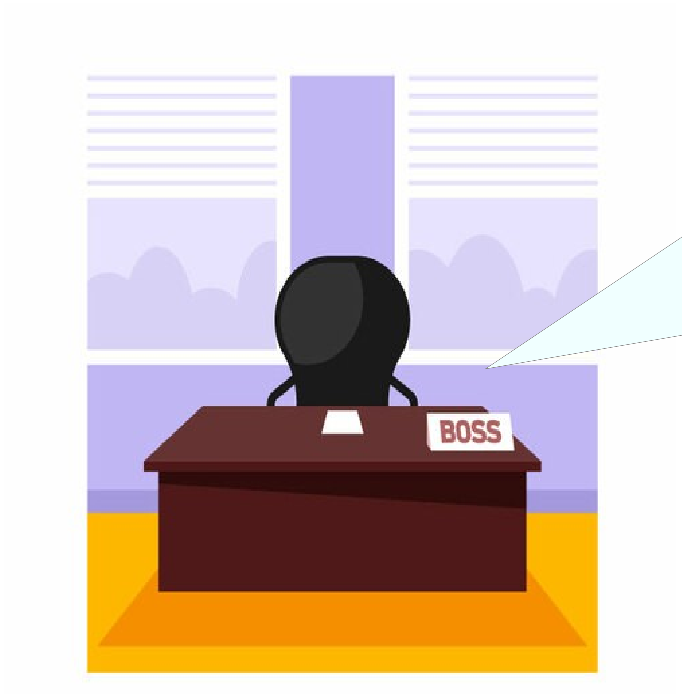
A: It is generally believed that:

"Hard" problems have NO efficient algorithms

But there's no proof for it yet...

SET-COVER VERTEX-COVER HAMILTONIAN-CYCLE

How do you prove that an NP-problem is "Hard"?



Design an efficient algorithm for problem N!

How do you prove that an NP-problem is “Hard”?

Design an efficient algorithm for problem N!

But… problem N is “Hard”

How do you prove that an NP-problem is “Hard”?

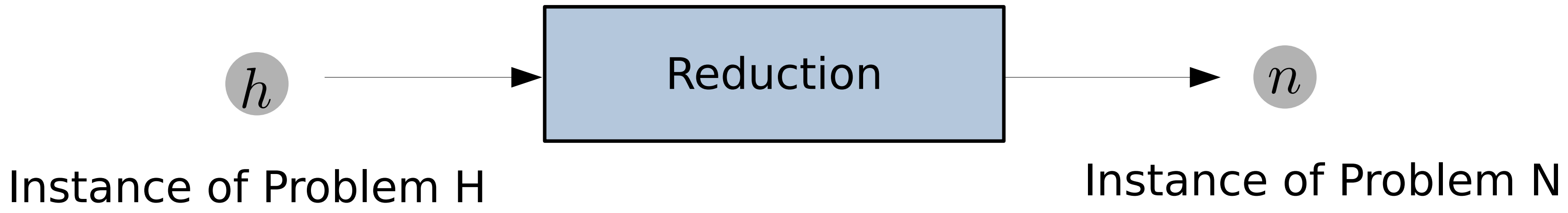
If N could be solved,
a known hard problem H
could be also solved.

How do you prove that an NP-problem is "Hard"?

"reduction"

If N could be solved,
a known hard problem H
could be also solved.

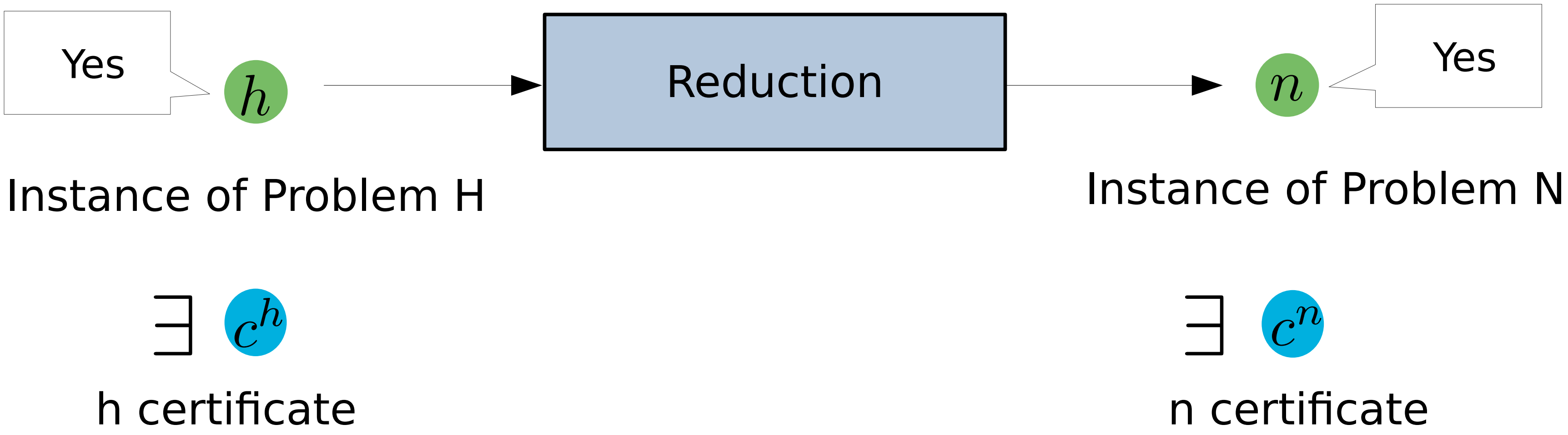
One-Call Reduction



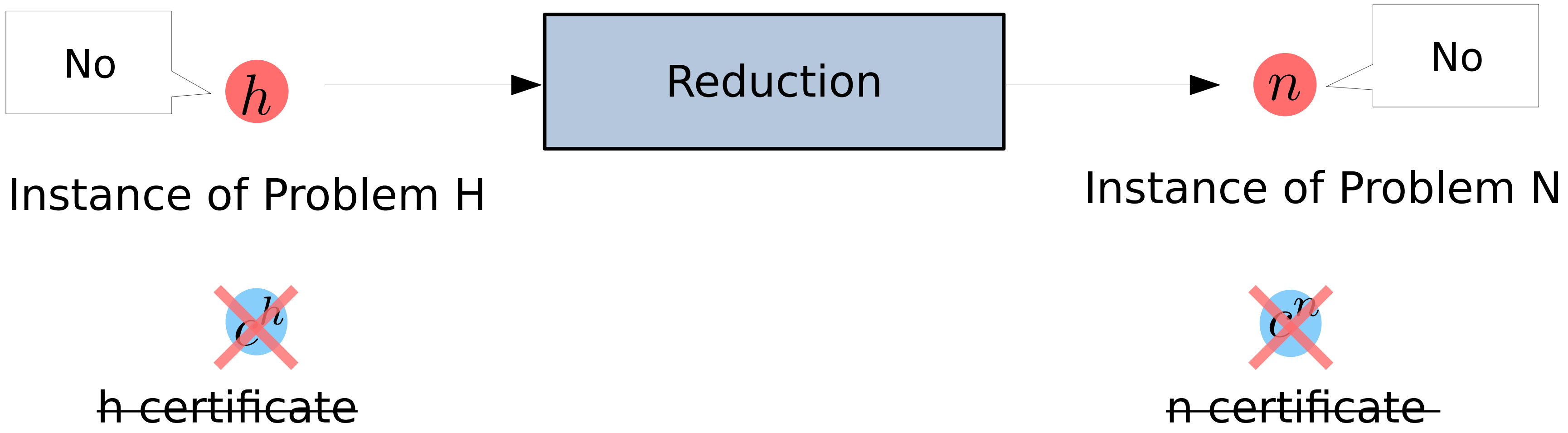
One-Call Reduction – Correctness Property

H is the problem known to be hard

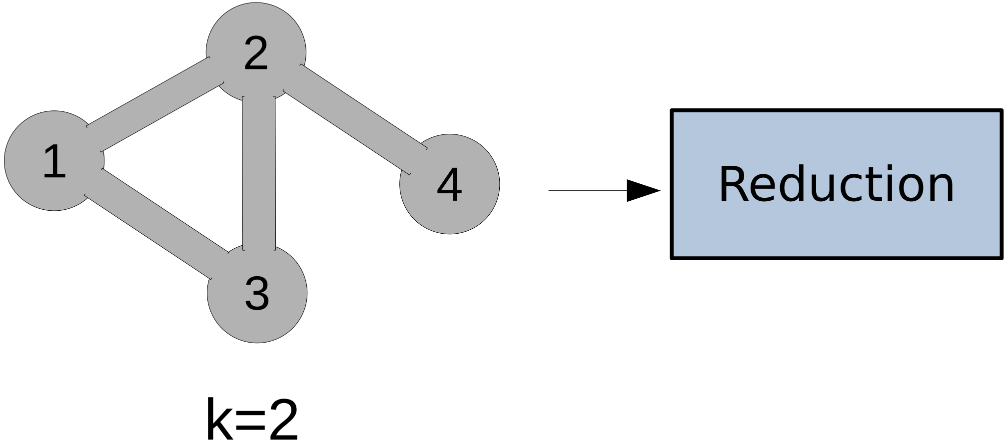
n is the new problem



One-Call Reduction – Correctness Property



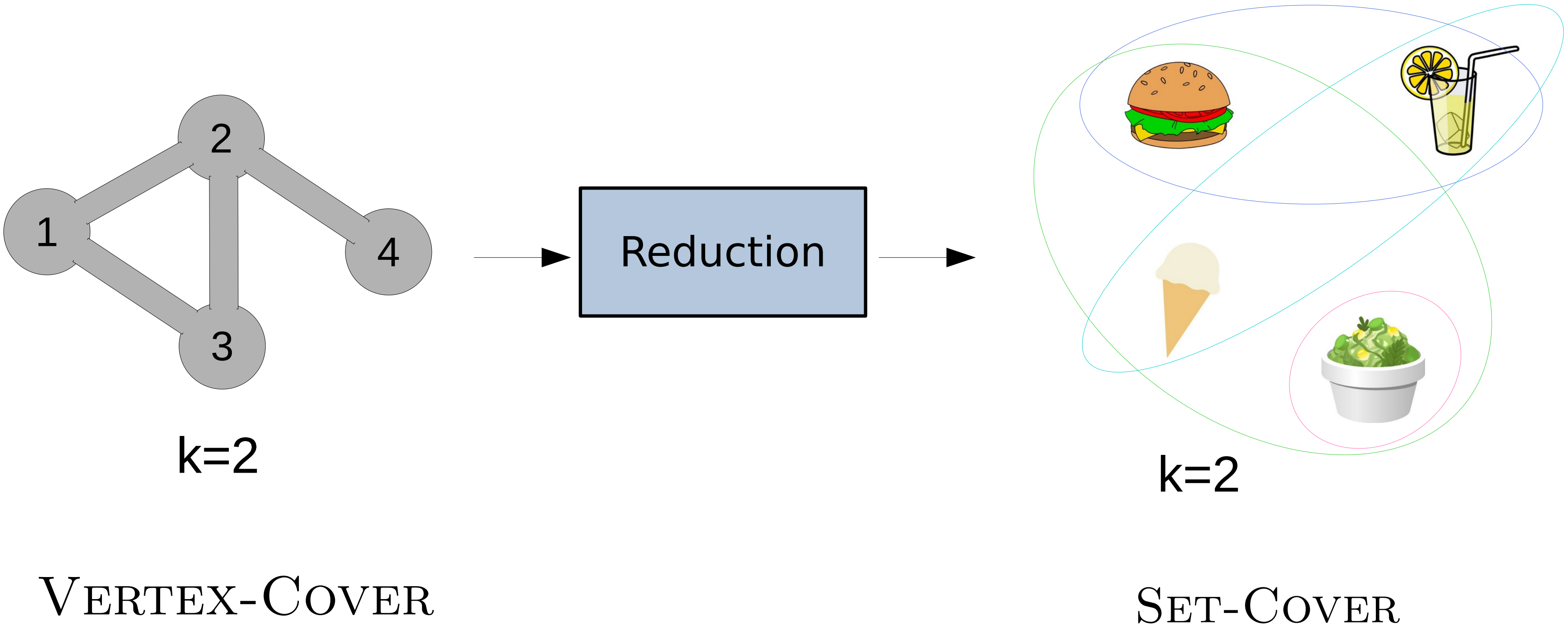
One-Call Reduction



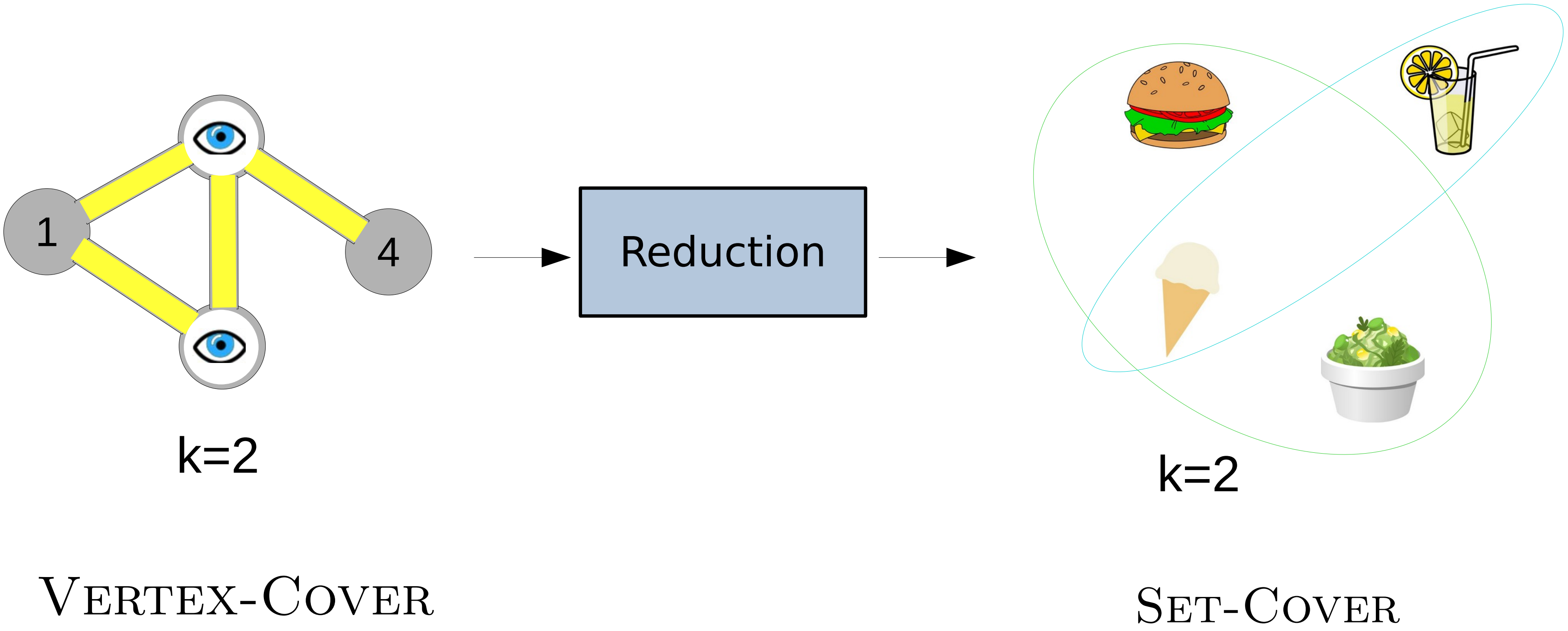
VERTEX-COVER

SET-COVER

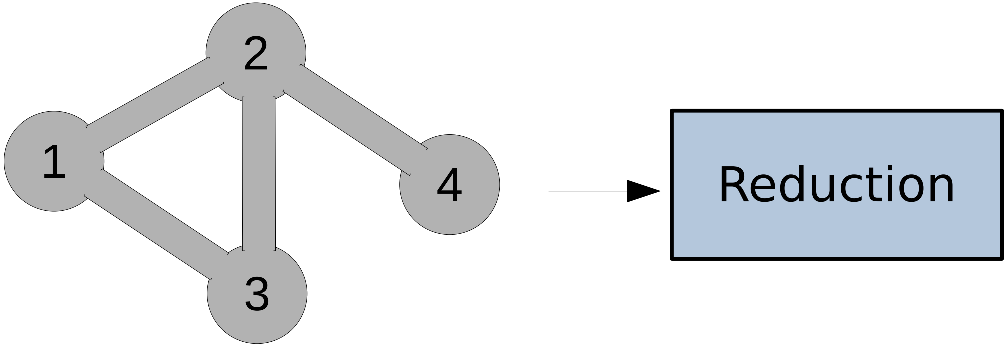
One-Call Reduction



One-Call Reduction



One-Call Reduction

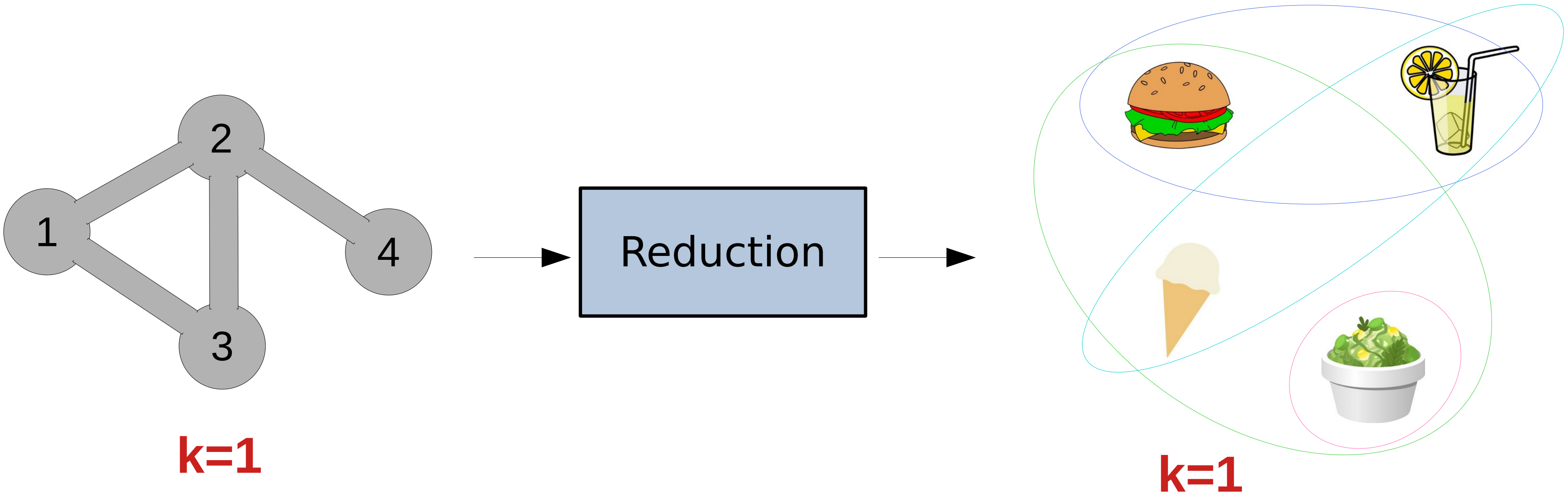


k=1

Vertex-Cover

Set-Cover

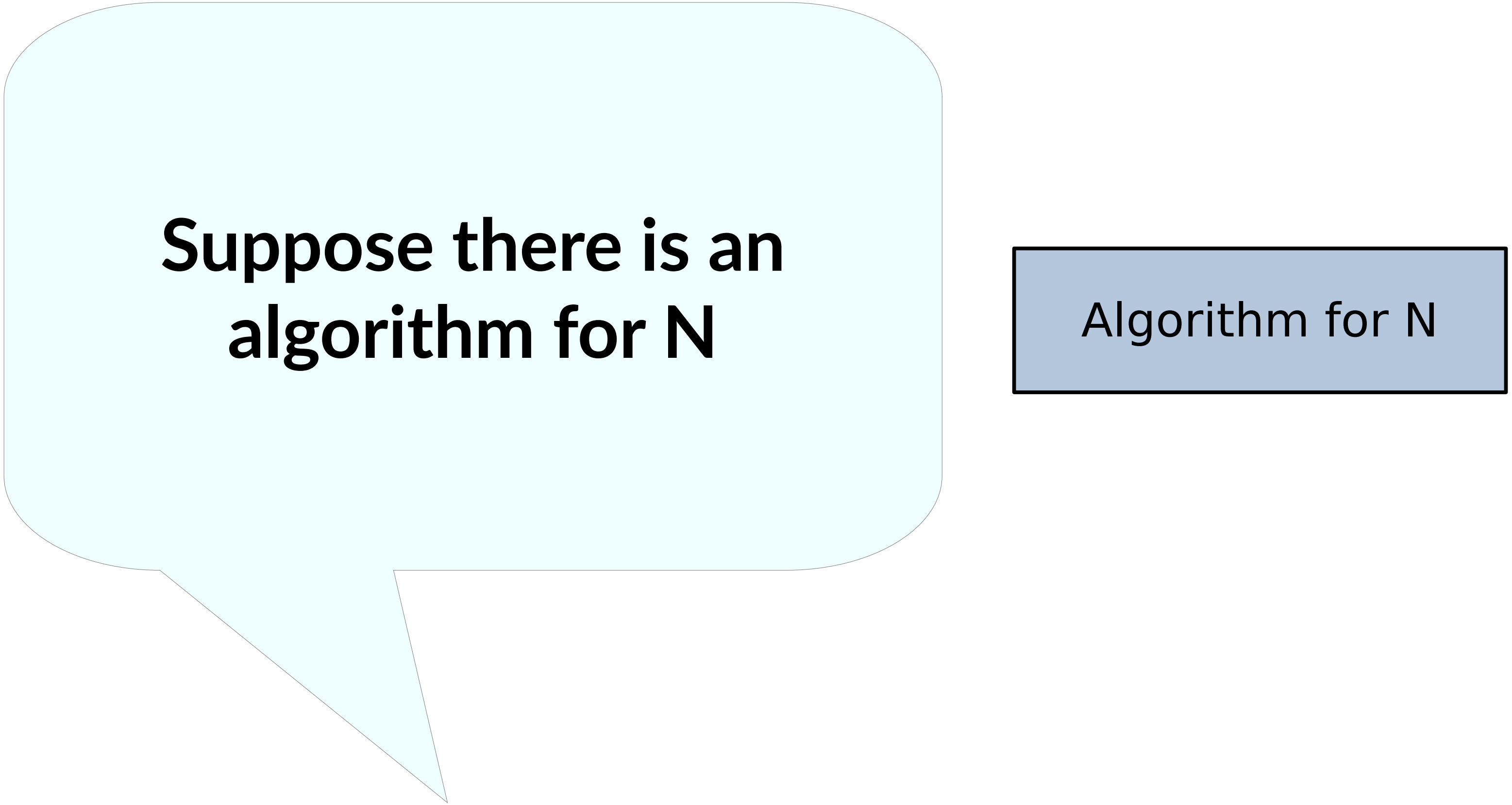
One-Call Reduction



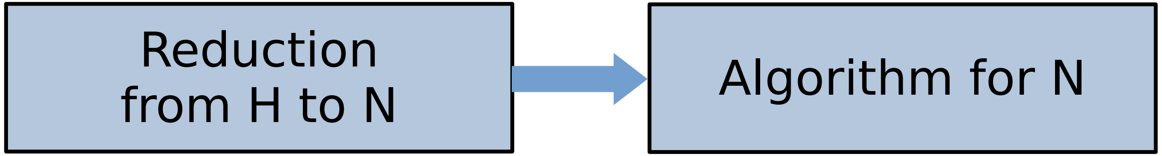
VERTEX-COVER

SET-COVER

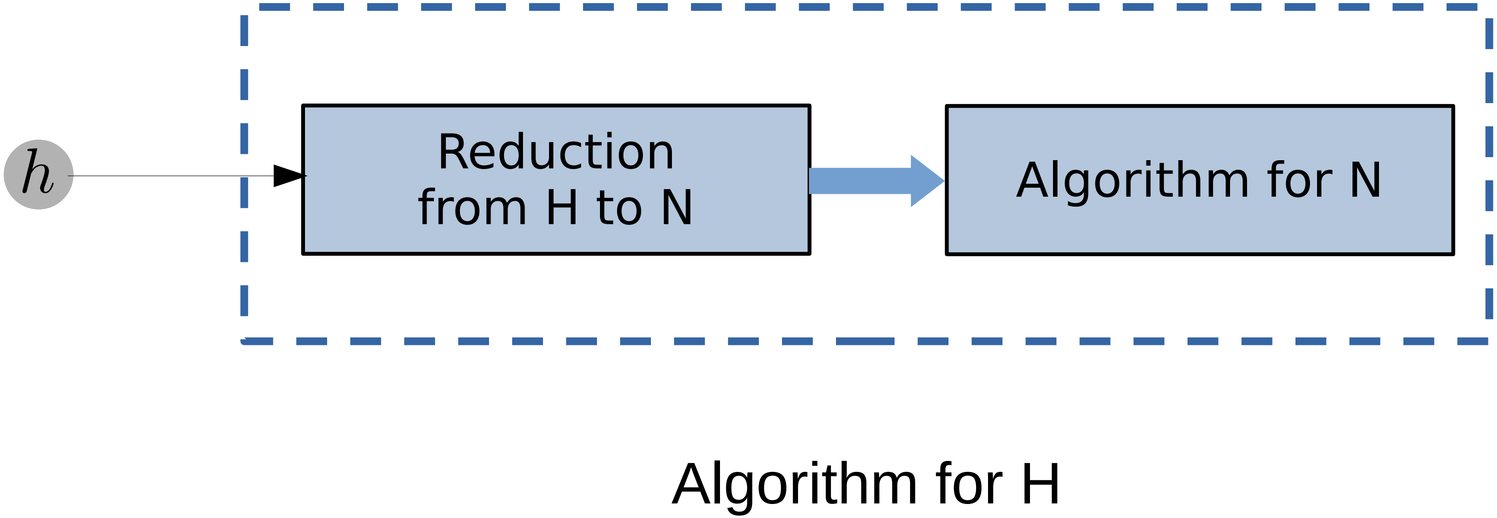
One-Call Reduction



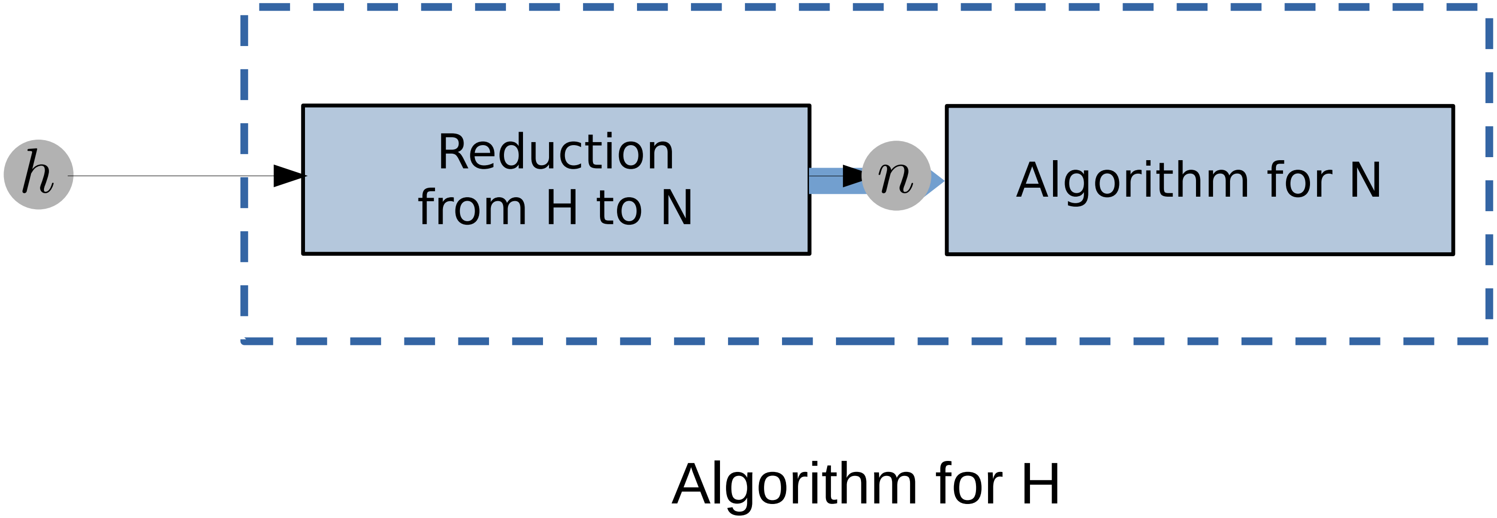
One-Call Reduction



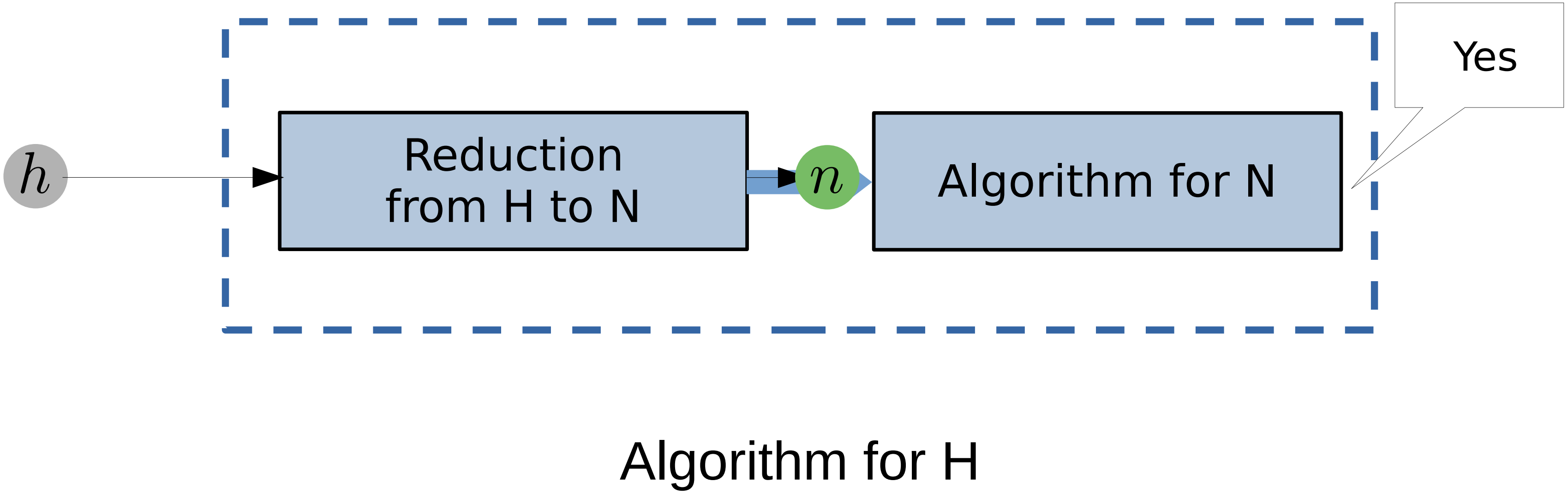
One-Call Reduction



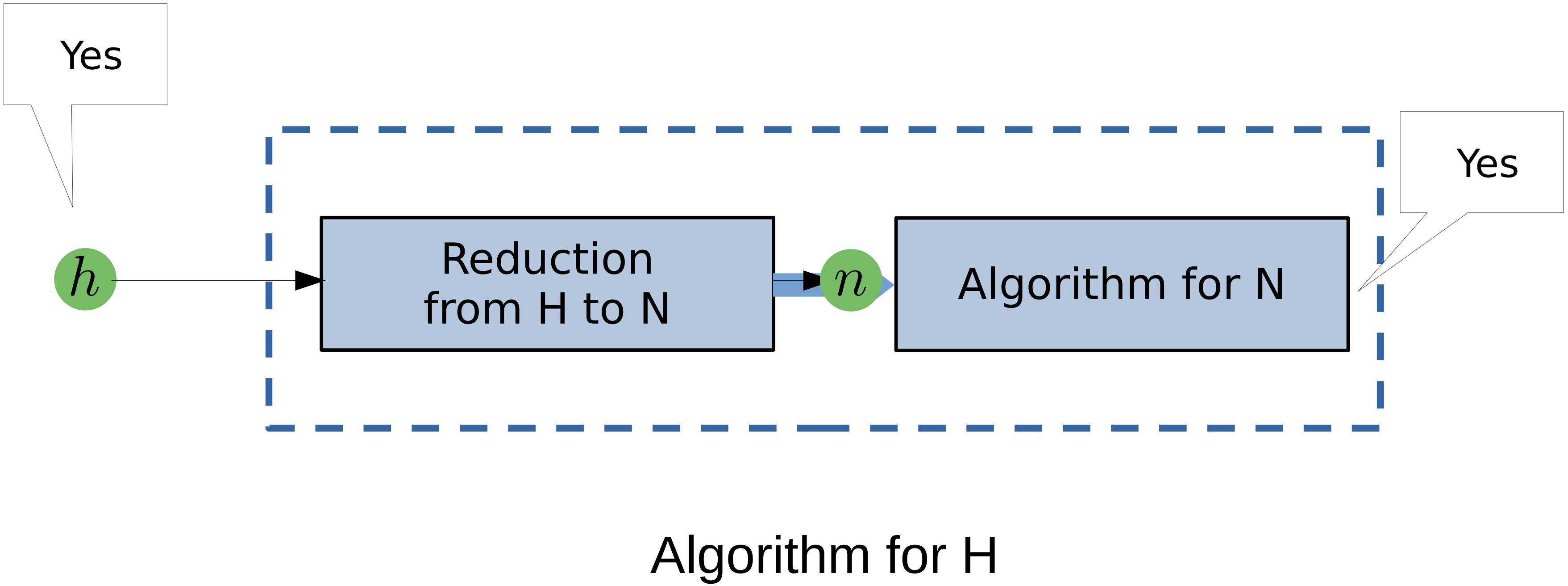
One-Call Reduction



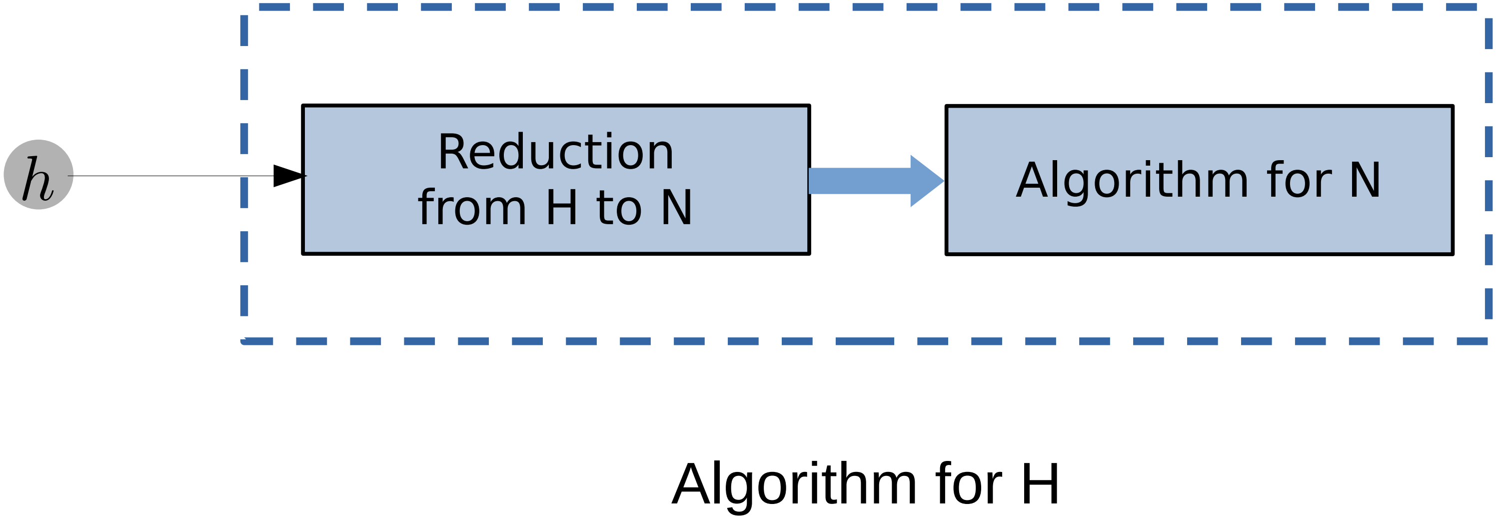
One-Call Reduction



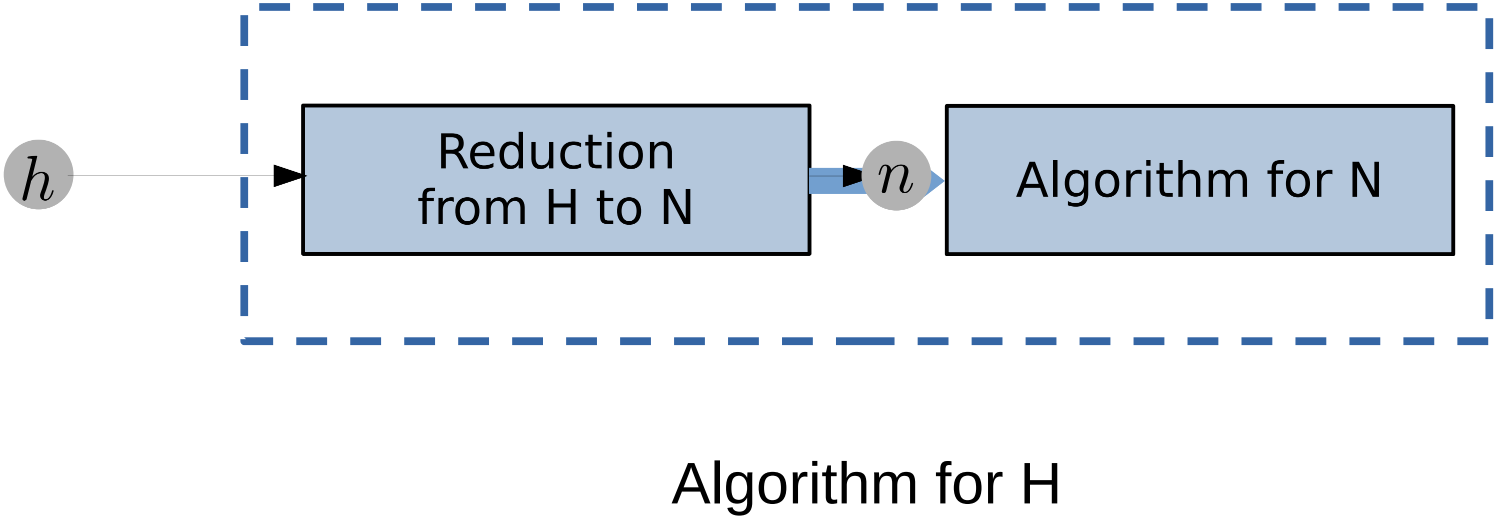
One-Call Reduction



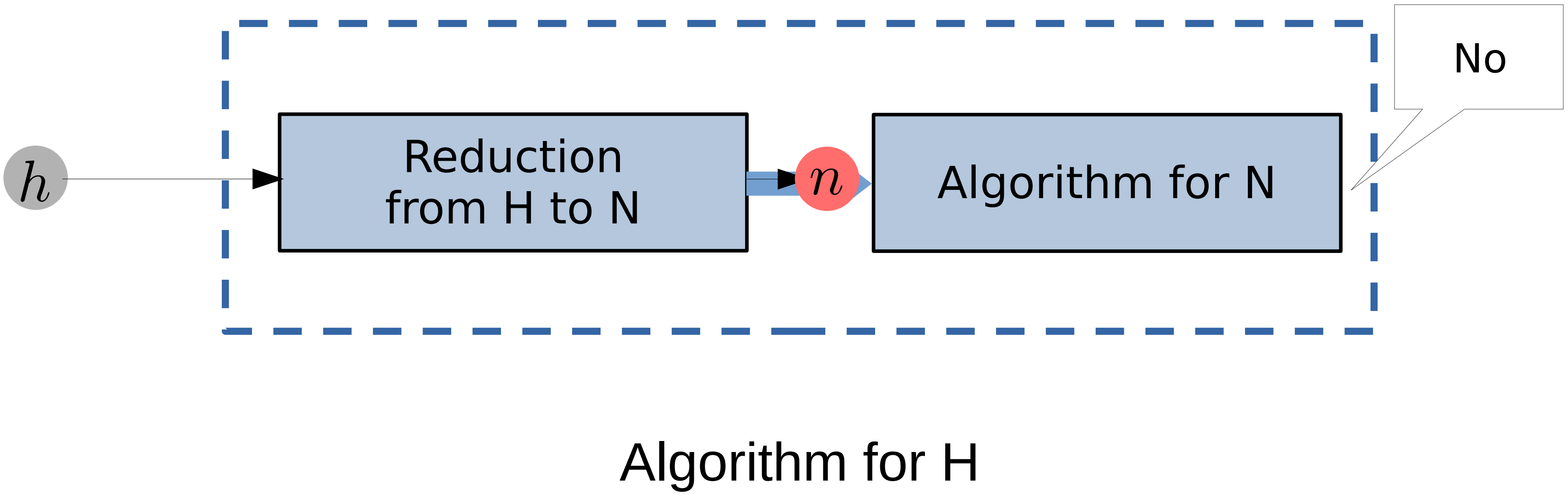
One-Call Reduction



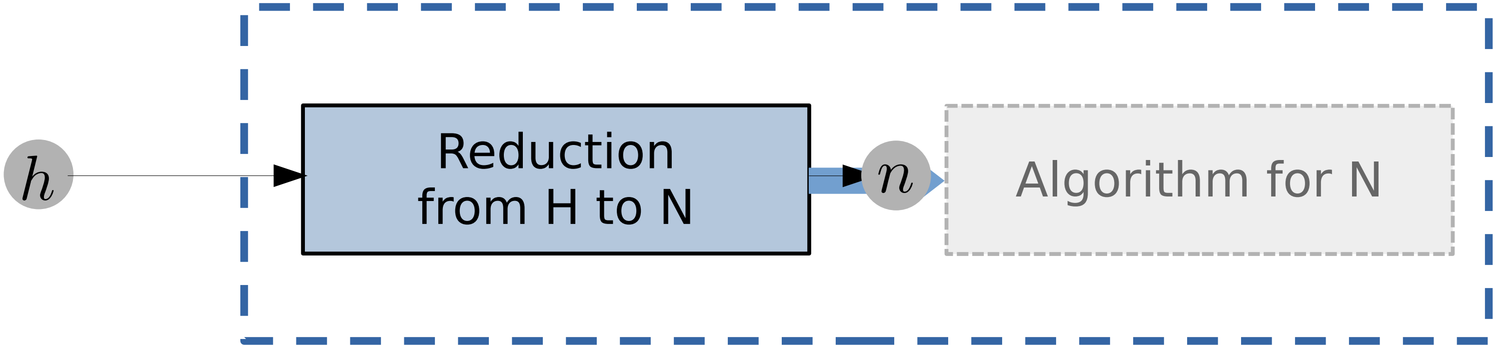
One-Call Reduction



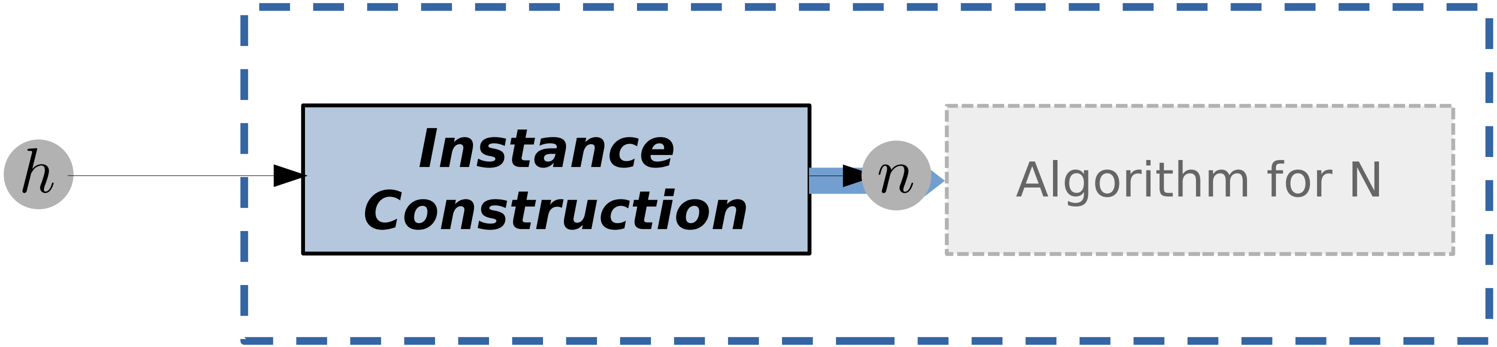
One-Call Reduction



Reduction and Justifications of Correctness



Call this part "*instance construction*" from now on

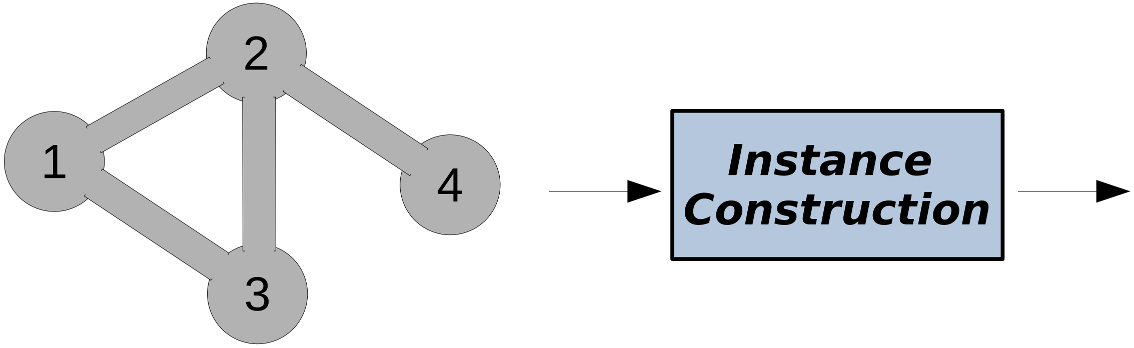


Instance Construction

VERTEX-COVER

SET-COVER

Instance Construction

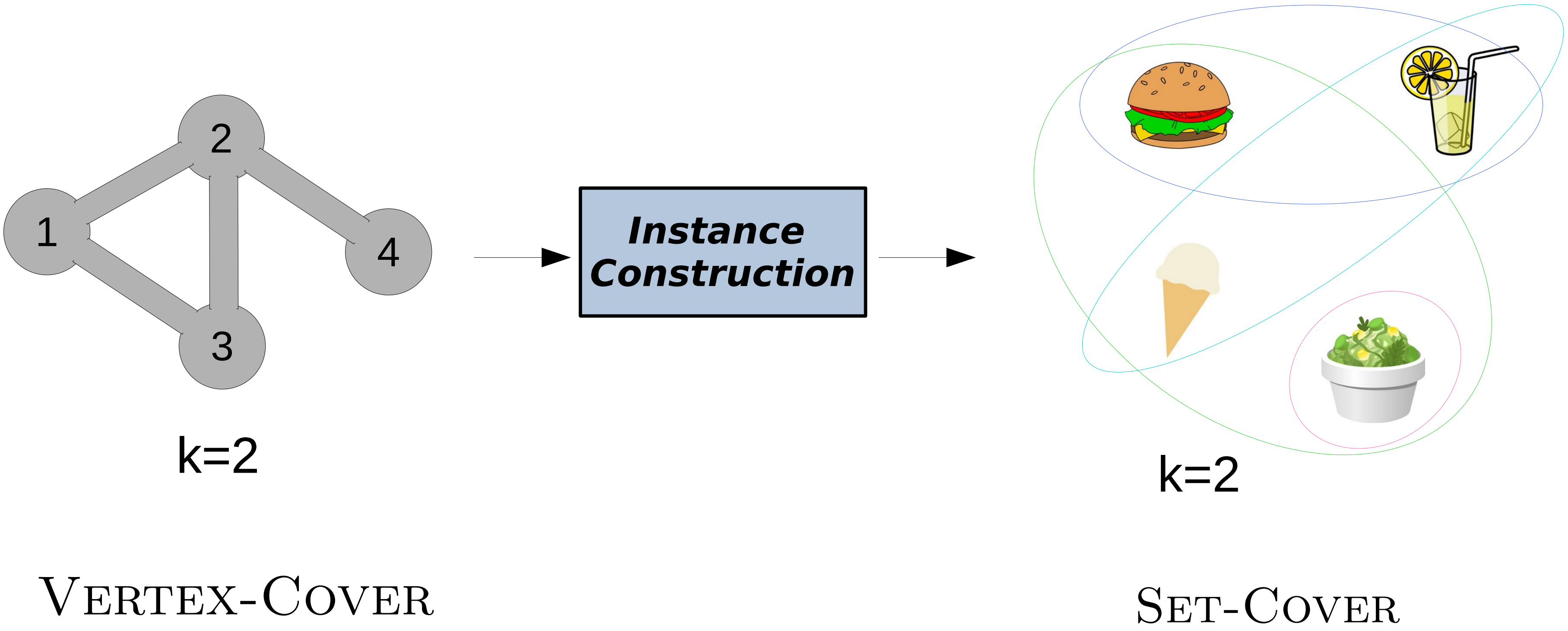


k=2

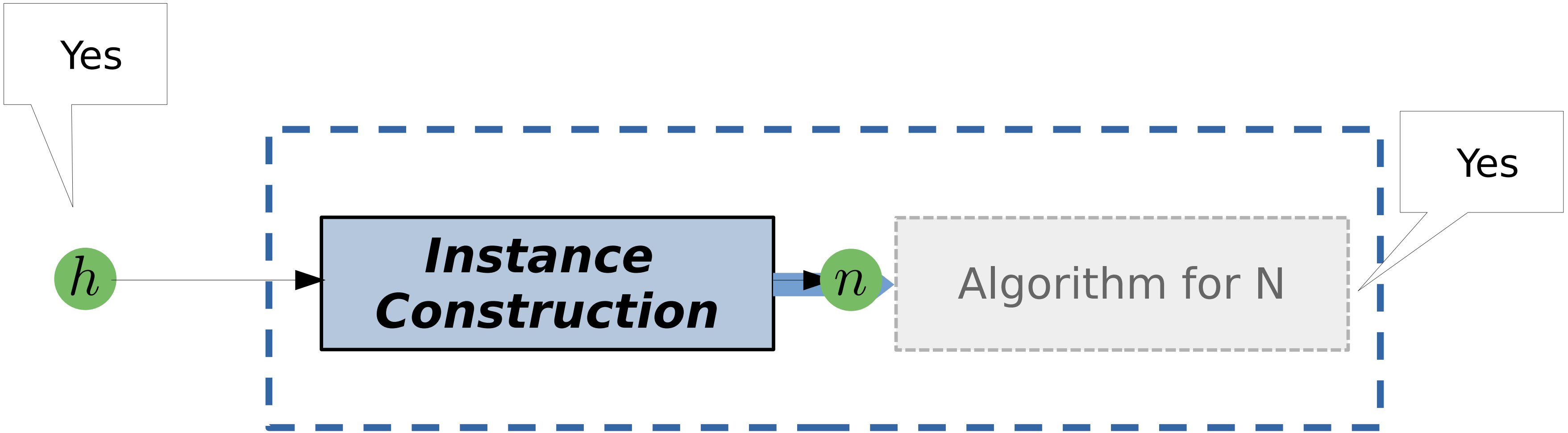
Vertex-Cover

Set-Cover

Instance Construction



Justifying N Yes => H Yes



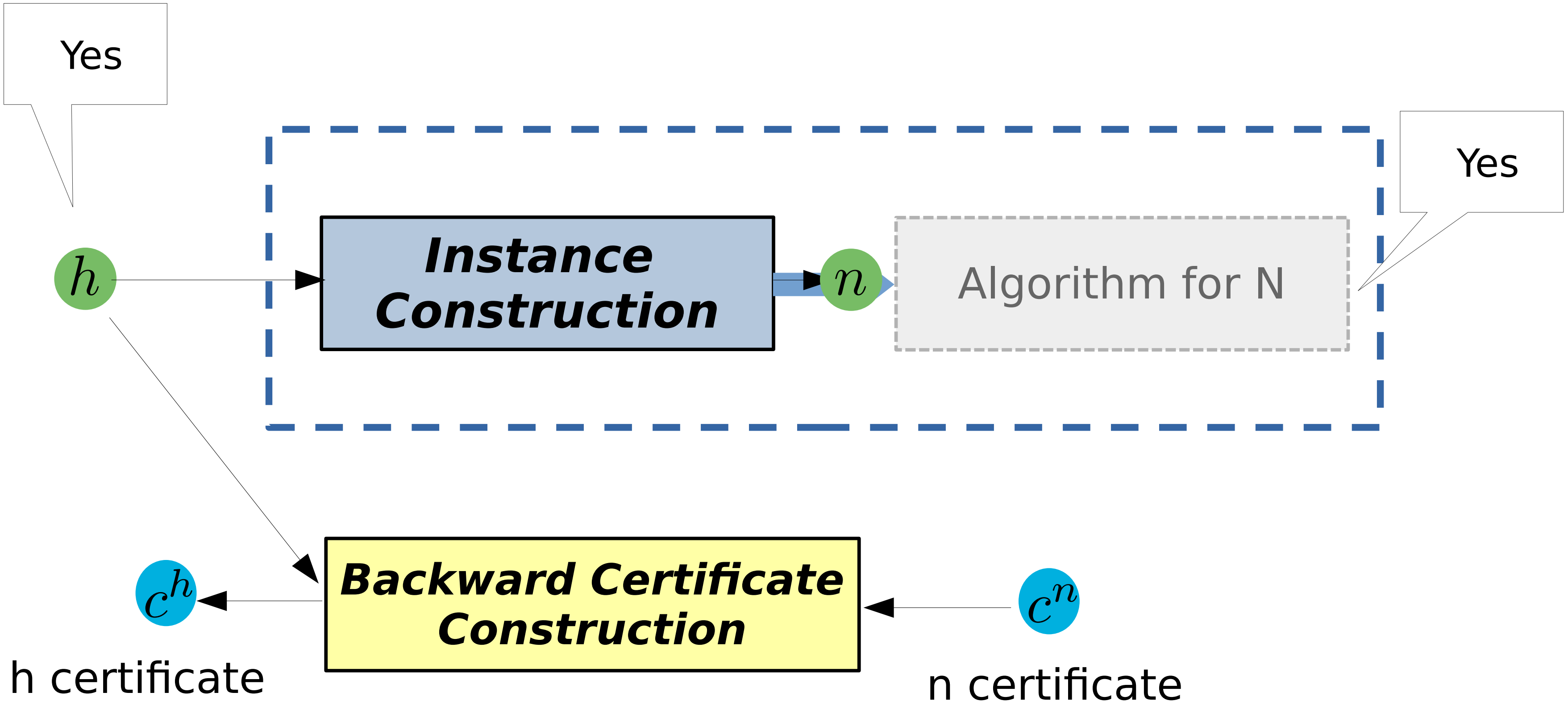
$\exists$ c^h

h certificate

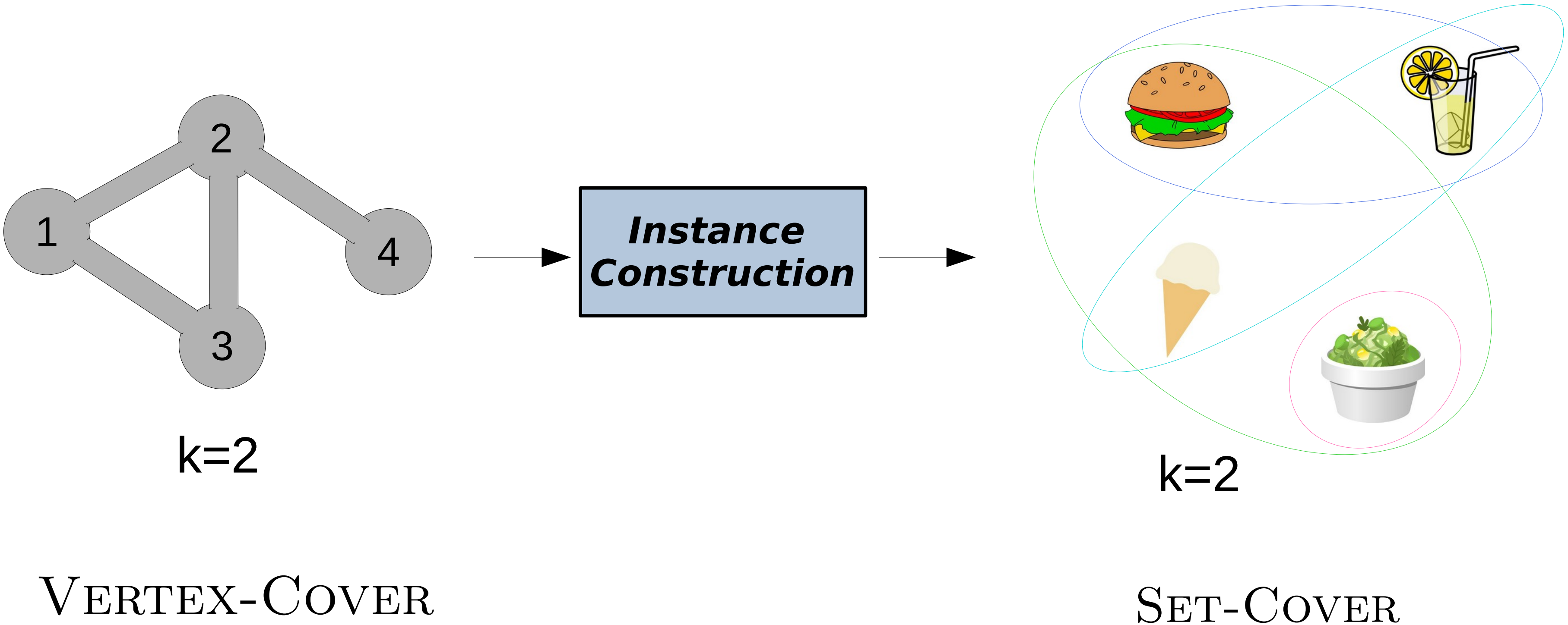
$\exists$ c^n

n certificate

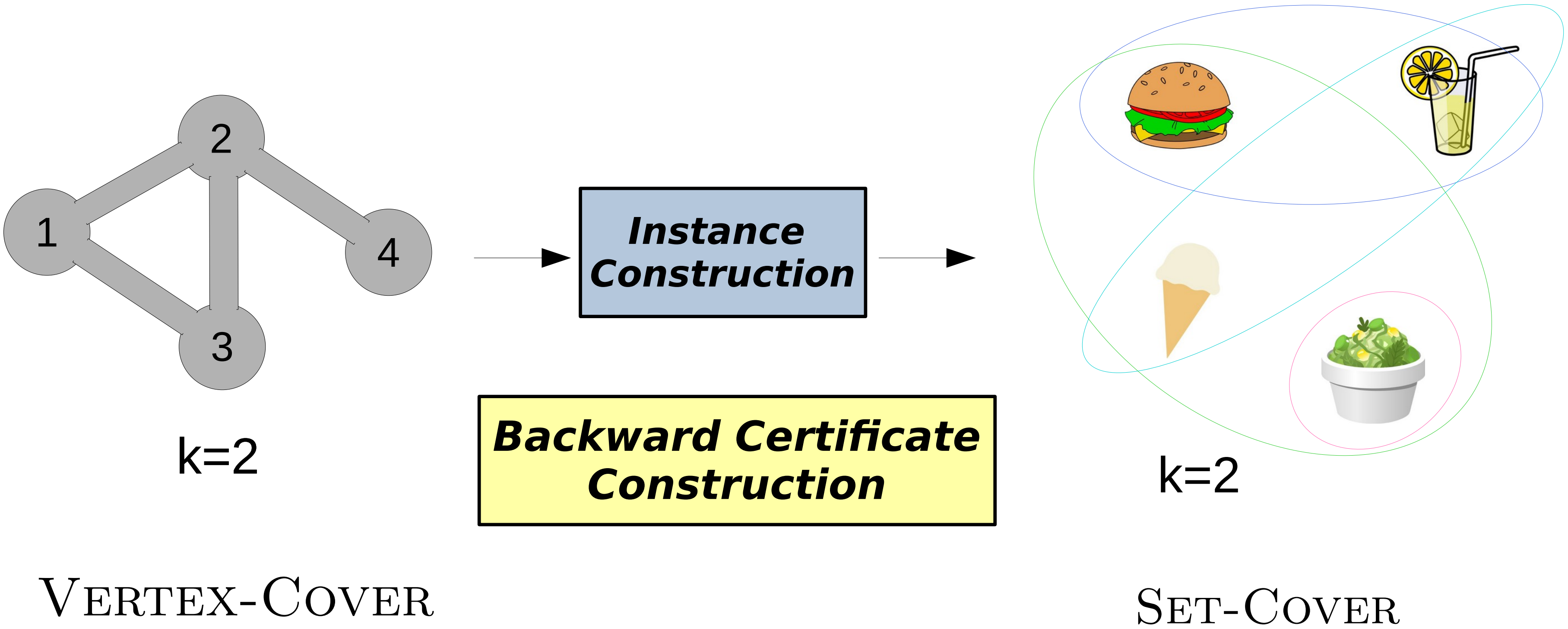
Justifying N Yes => H Yes



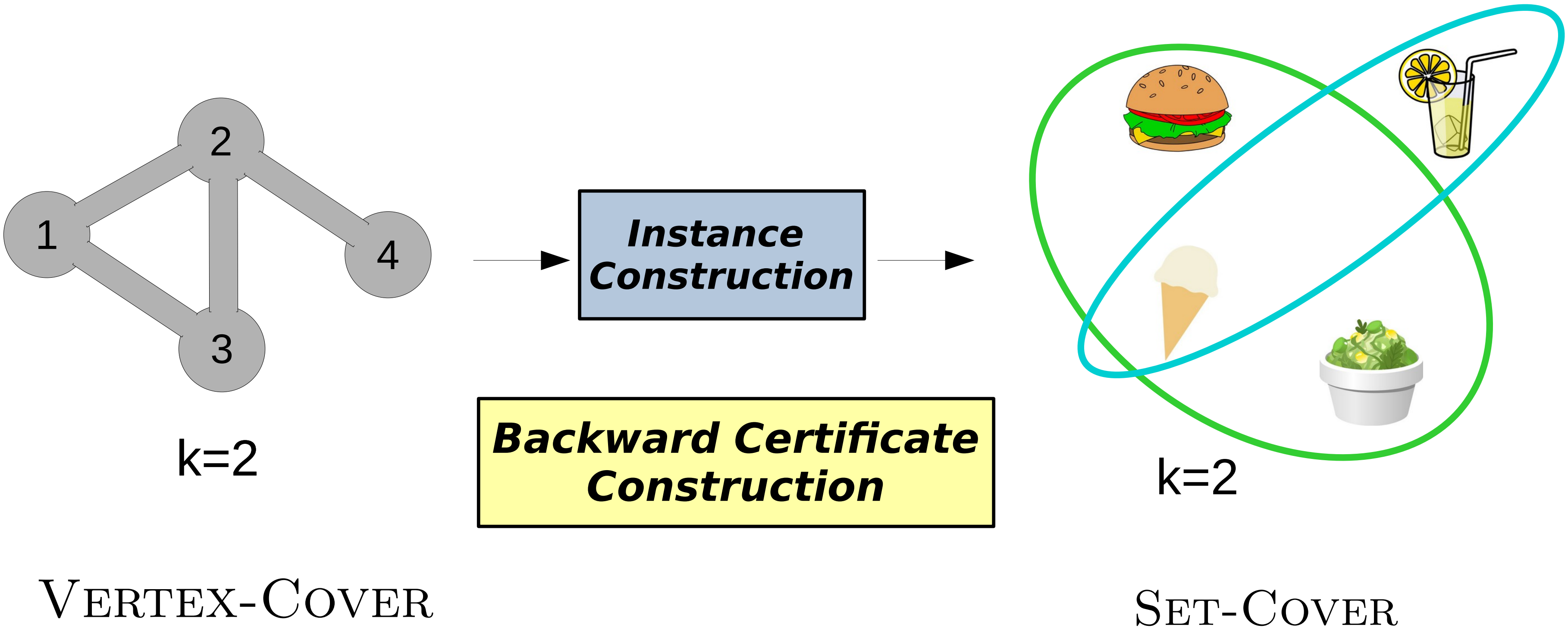
Backward Certificate Construction



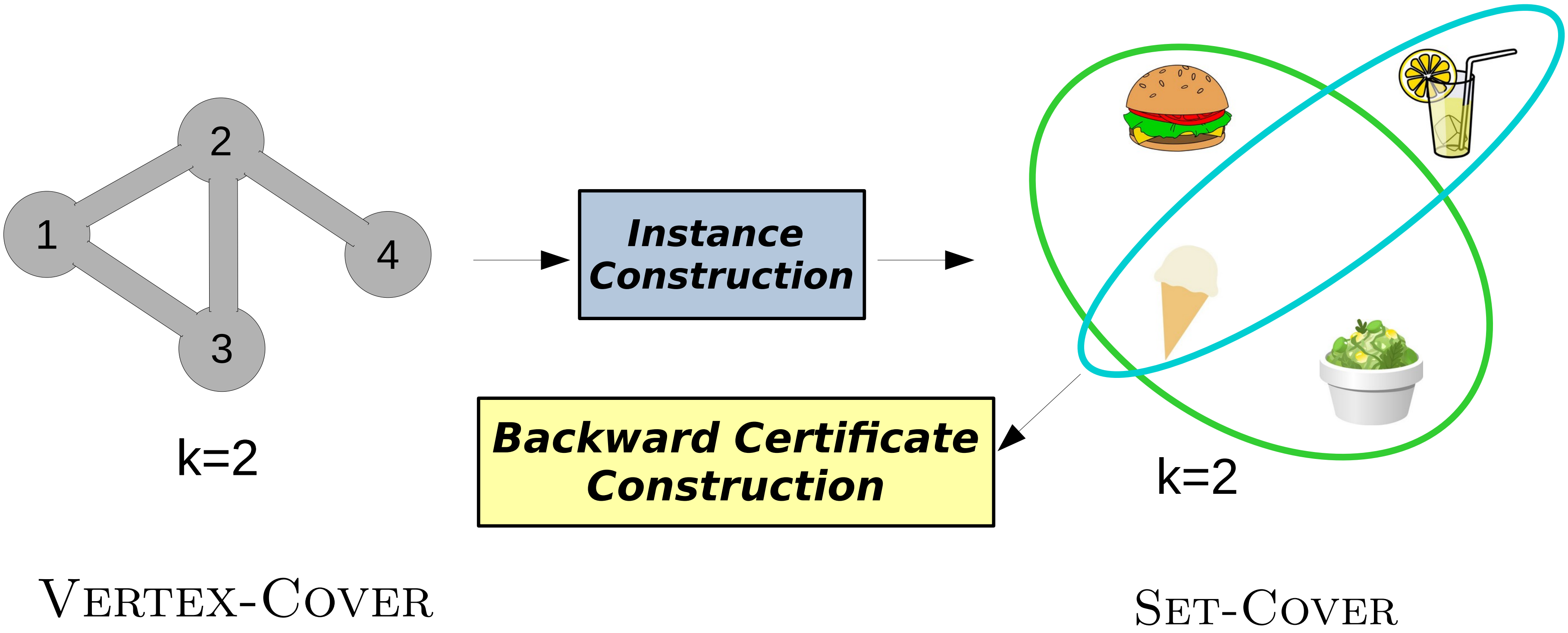
Backward Certificate Construction



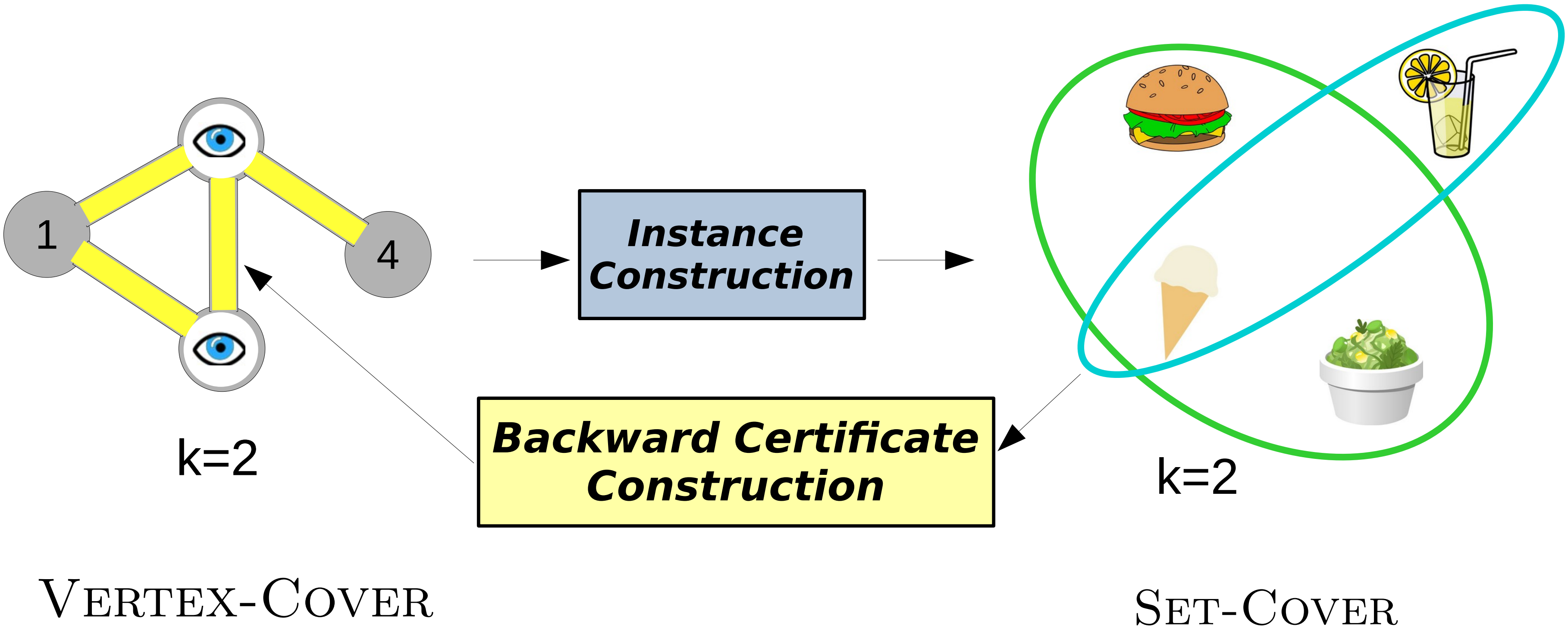
Backward Certificate Construction



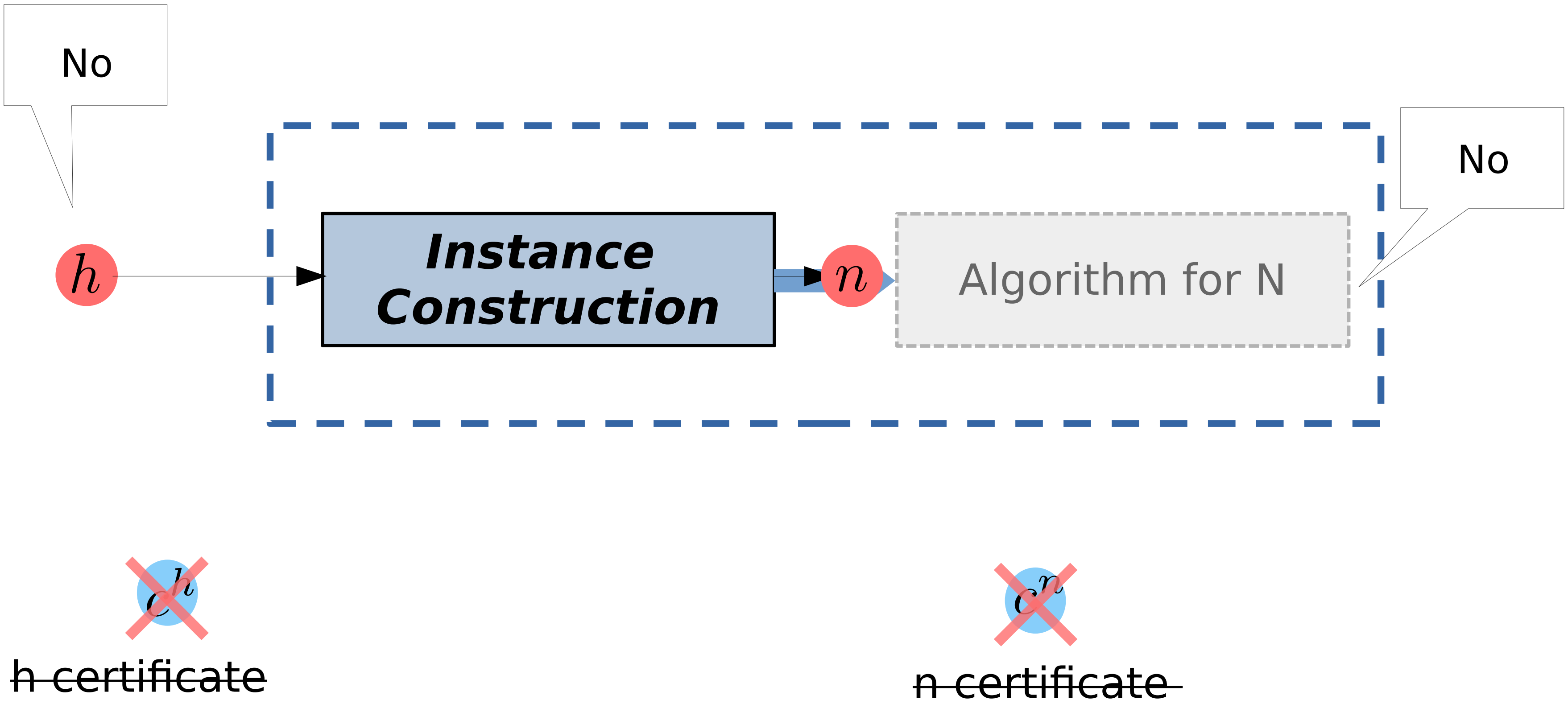
Backward Certificate Construction



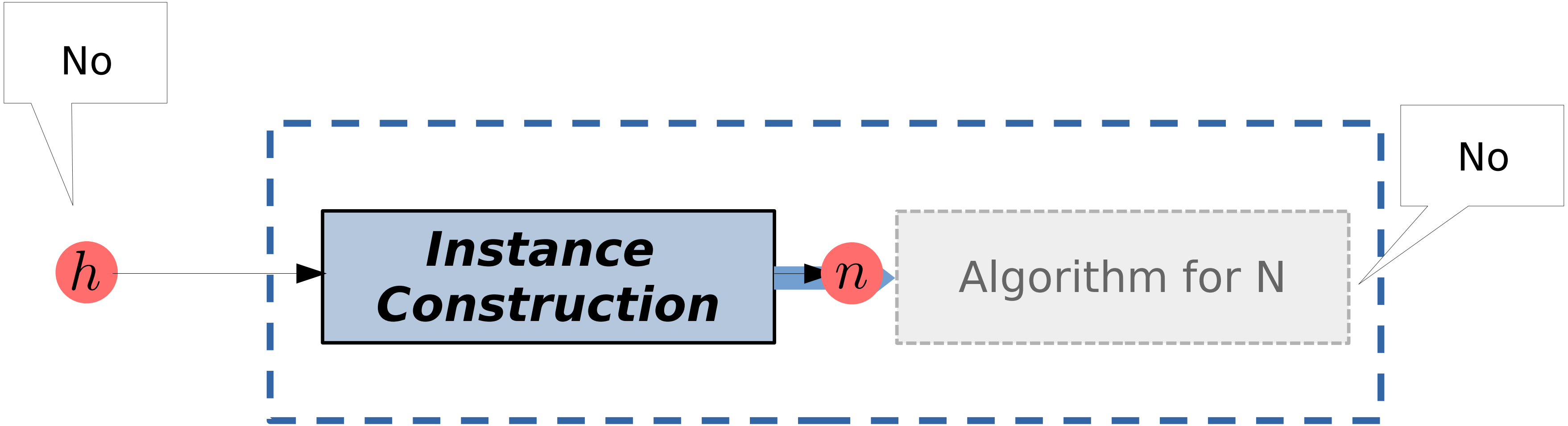
Backward Certificate Construction



Justifying N No => H No

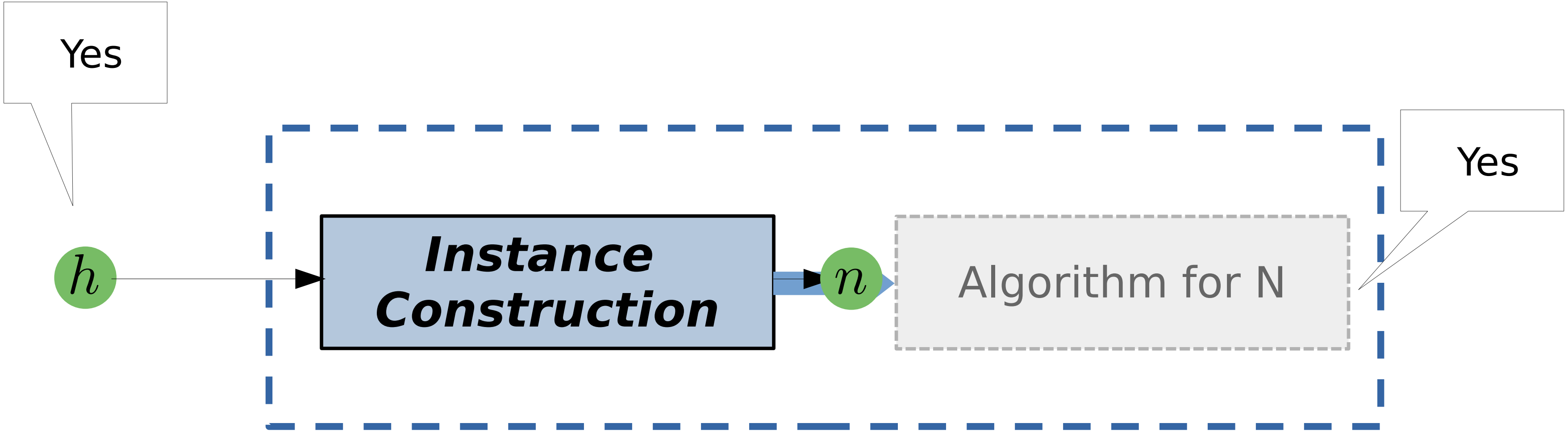


Justifying N No => H No



$$\neg\exists\ c^x \Longleftarrow \neg\exists\ c^y$$

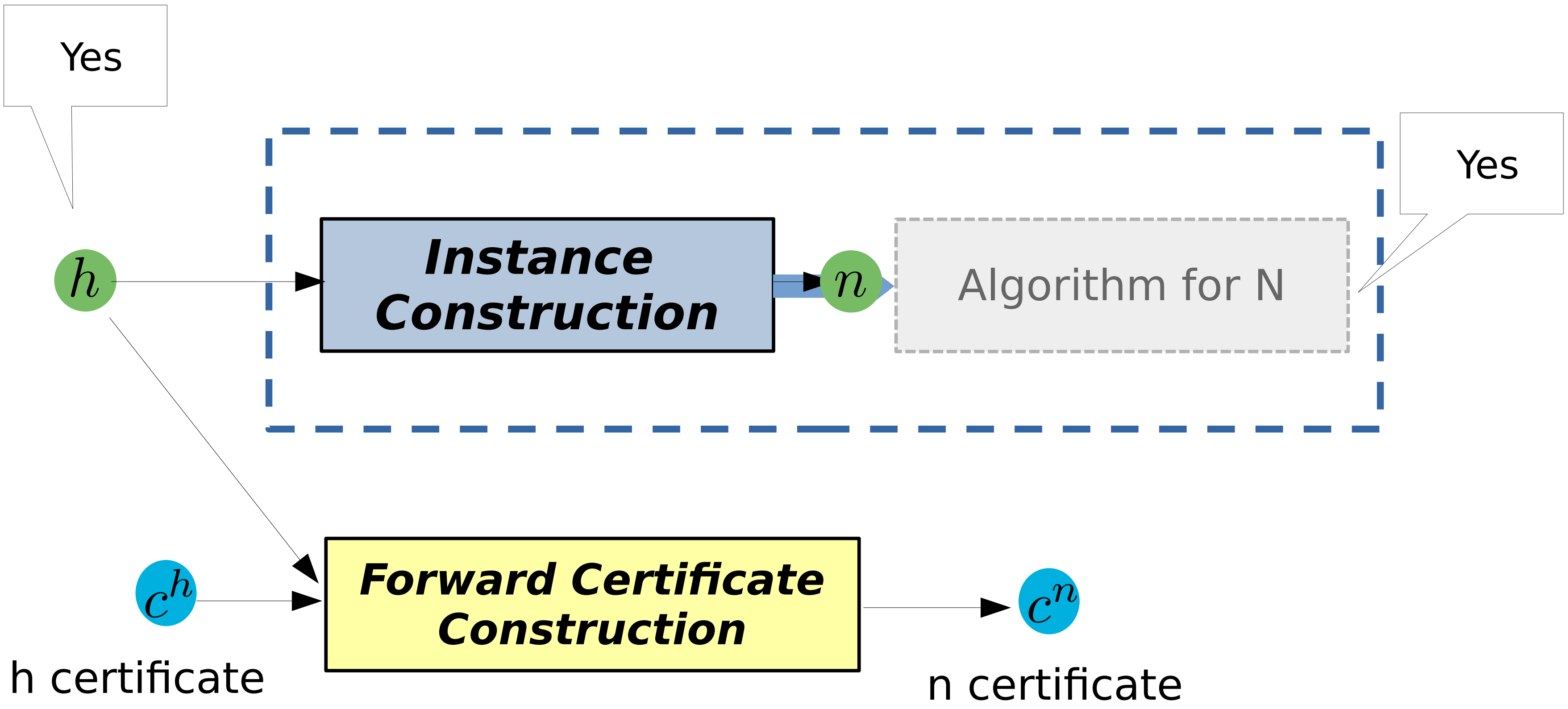
Justifying N No => H No



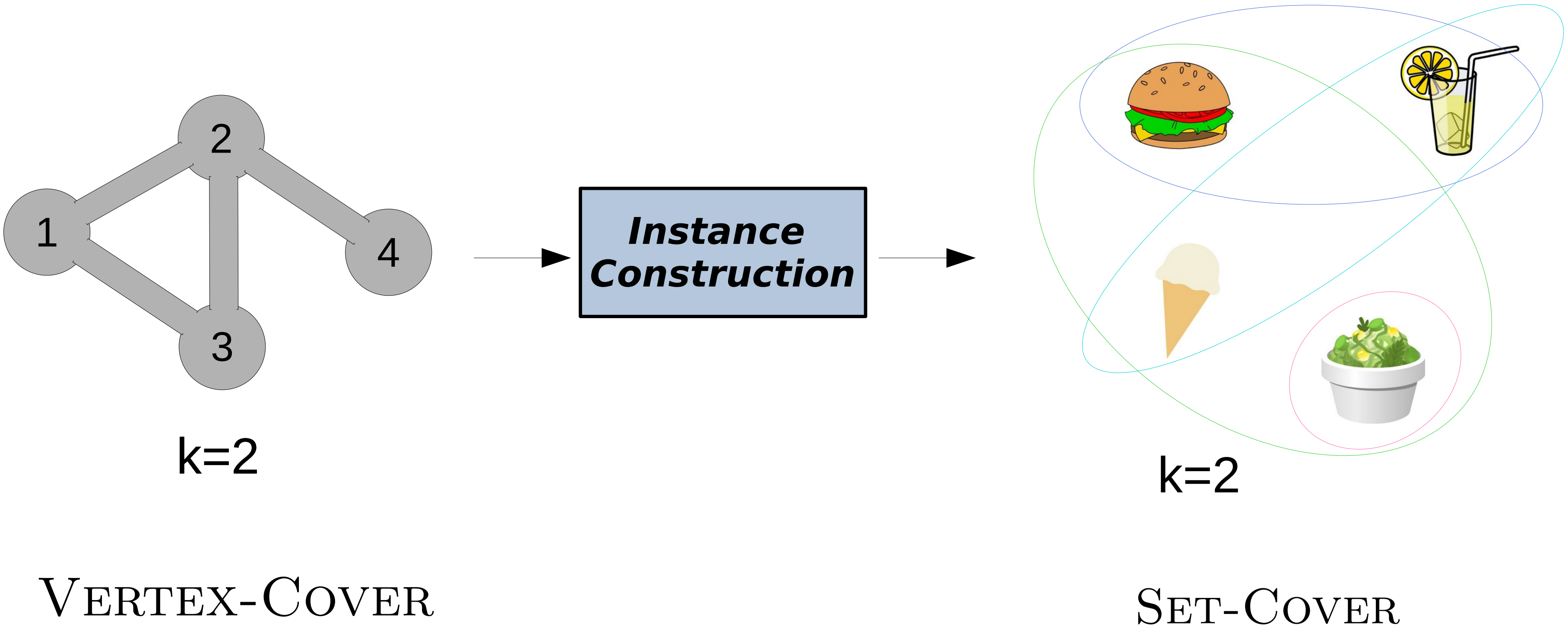
$$\exists\ c^x \implies \exists\ c^y$$

* we are in a classical world

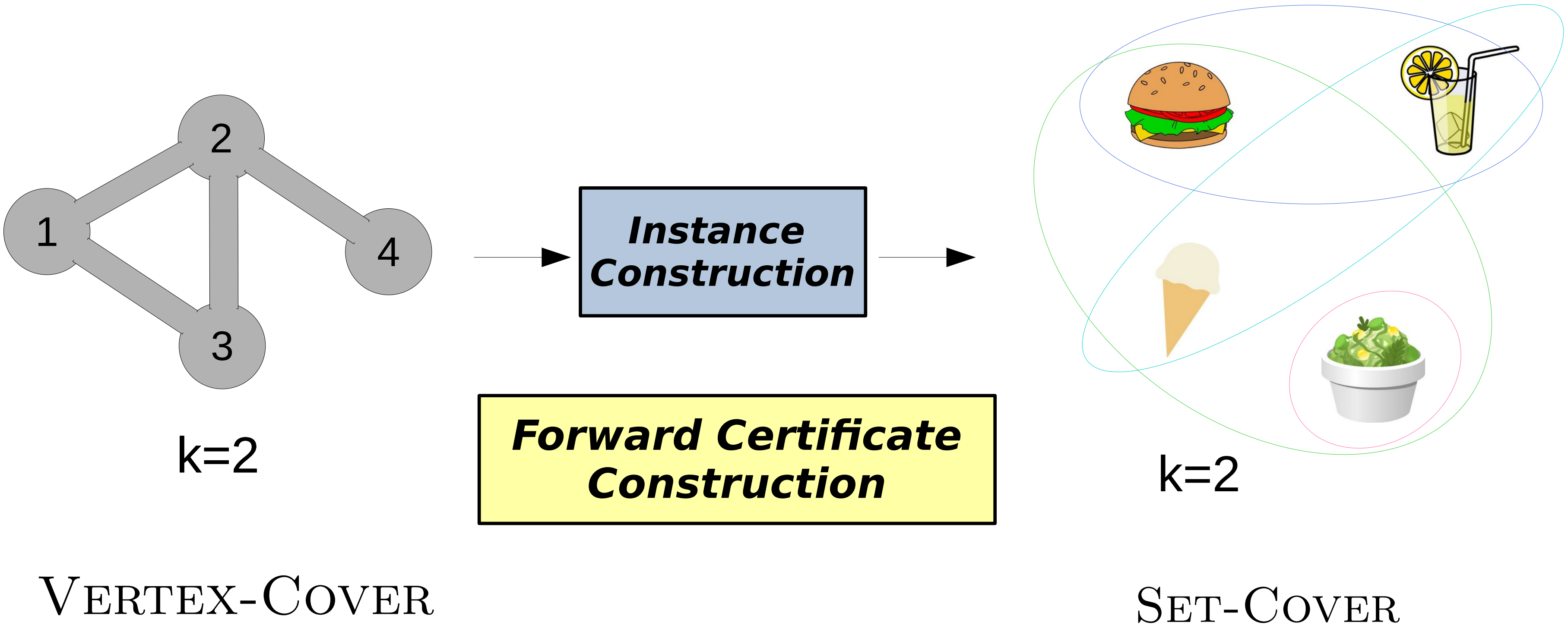
Justifying N No => H No



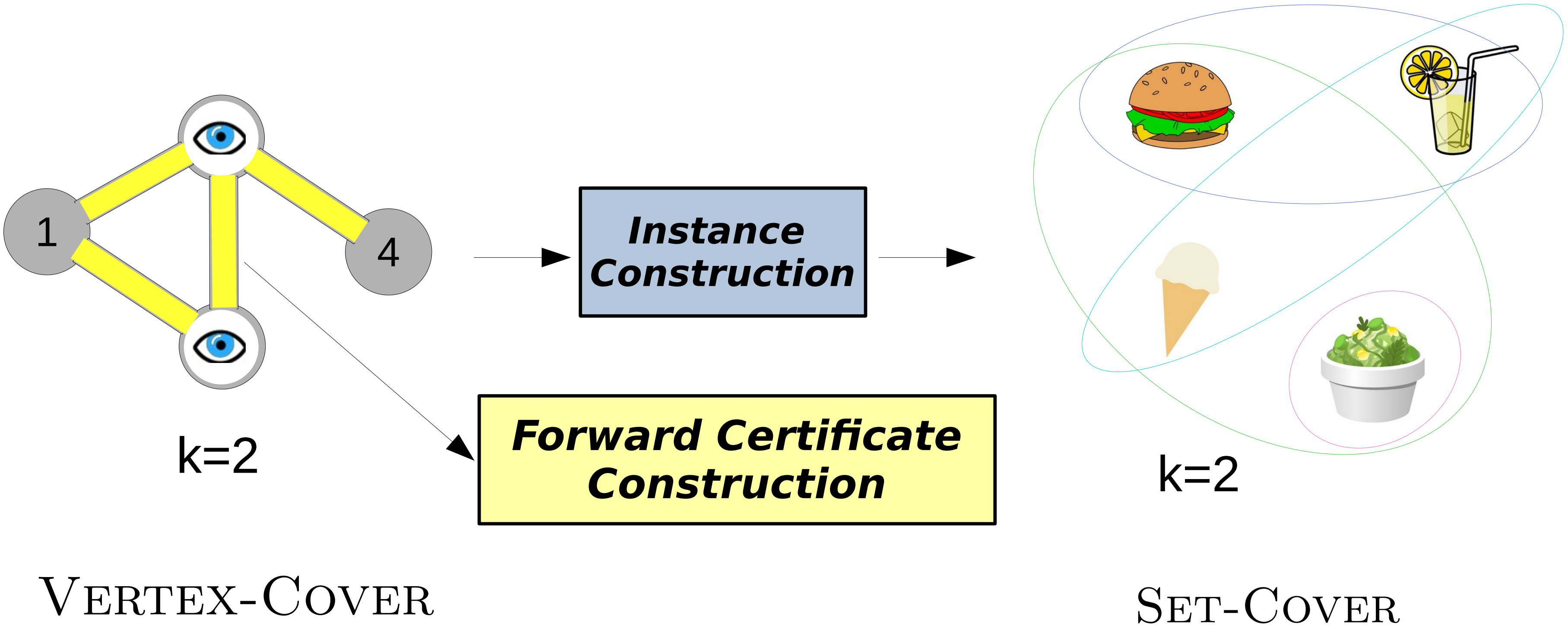
Forward Certificate Construction



Forward Certificate Construction



Forward Certificate Construction



Forward Certificate Construction