Advanced Algorithmic Techniques (COMP523)

NP-Completeness 2

Recap and plan

- **Previous lecture:**
  - Polynomial time reductions
  - Computational classes: P and NP
  - NP-hardness and NP-completeness
  - NP-Complete problems: 3SAT and Vertex Cover
- **This lecture:**
  - Decision vs Optimisation.
  - NP-hardness for Subset Sum and Knapsack.
  - A catalogue of NP-complete problems.
  - A taxonomy of NP-complete problems.

Recall: Vertex Cover

- Definition: A vertex cover C of a graph G=(V, E) is a subset of the nodes such that every edge e in the graph has at least one endpoint in C.

- Definition: A minimum vertex cover is a vertex cover of the smallest possible size.

- Vertex Cover
  Input: A graph G=(V, E)
  Output: A minimum vertex cover.

Vertex Cover decision version

- Definition: A vertex cover C of a graph G=(V, E) is a subset of the nodes such that every edge e in the graph has at least one endpoint in C.

- Definition: A minimum vertex cover is a vertex cover of the smallest possible size.

- Vertex Cover
  Input: A graph G=(V, E) and a number k
  Output: Is there a vertex cover of size ≤ k?.

From optimisation to decision

- We are given an optimisation problem P (assume minimisation).
  - E.g., find the minimum vertex cover.
- We introduce a threshold k.
- The decision version P_d becomes: Given an instance of P and the threshold k as input, is there a solution to P of value at most k?
  - E.g., is there a vertex cover of size at most k?

Optimisation vs decision

Optimisation vs decision

- If we can solve P in polynomial time, we can solve P_d in polynomial time. (why?)

Optimisation vs decision

- If we can solve P in polynomial time, we can solve P_d in polynomial time. (why?)
  - This implies that if the decision version is NP-hard, so is the optimisation version.

Optimisation vs decision

- If we can solve P in polynomial time, we can solve P_d in polynomial time. (why?)
    - This implies that if the decision version is NP-hard, so is the optimisation version.
- Often the opposite is also true.

Optimisation vs decision

- If we can solve P in polynomial time, we can solve P_d in polynomial time. (why?)
  - This implies that if the decision version is NP-hard, so is the optimisation version.
- Often the opposite is also true.
  - If we can solve P_d in polynomial time, we can solve P in polynomial time.

Optimisation vs decision

- Vertex Cover (Optimisation)
  Input: A graph $G=(V, E)$
  Output: A minimum vertex cover.

- Vertex Cover (Decision)
  Input: A graph $G=(V, E)$ and a number k
  Output: Is there a vertex cover of size $\leq k$?.

Optimisation vs decision

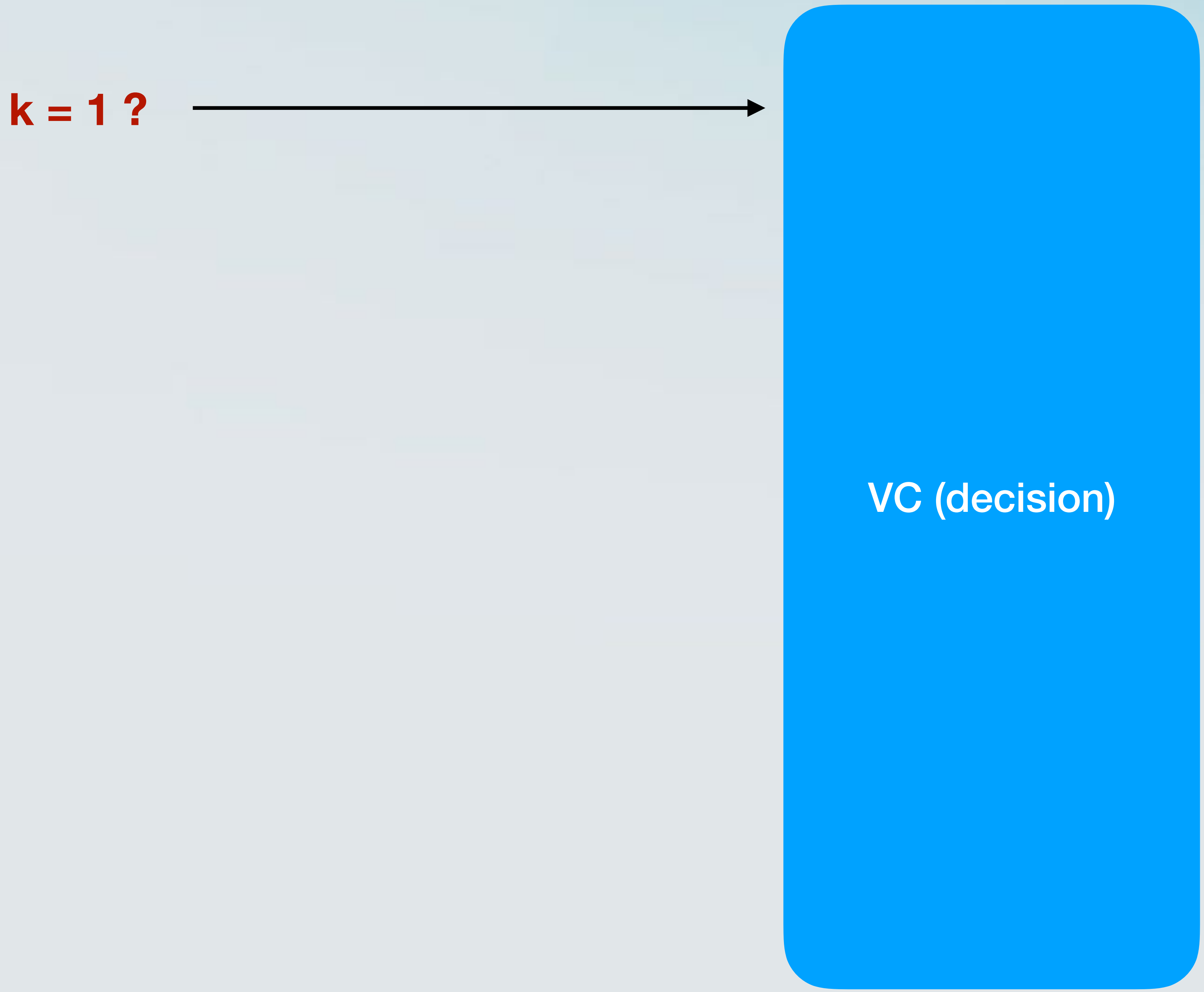
- Vertex Cover Size (Optimisation)
  Input: A graph $G=(V, E)$
  Output: **The size of** a minimum vertex cover.

- Vertex Cover (Decision)
  Input: A graph $G=(V, E)$ and a number k
  Output: Is there a vertex cover of size $\leq k$?.

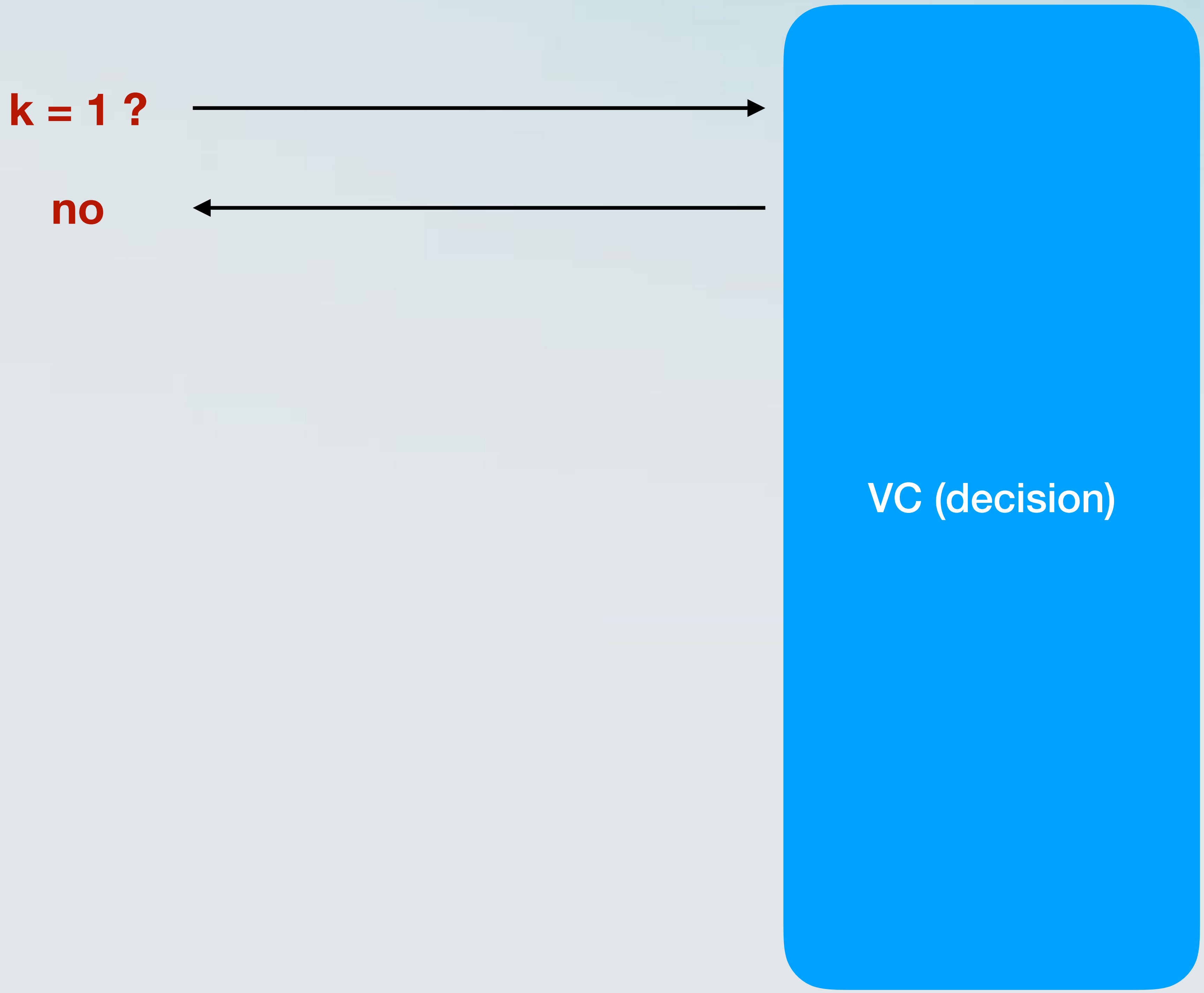
Vertex Cover Size

VC (decision)

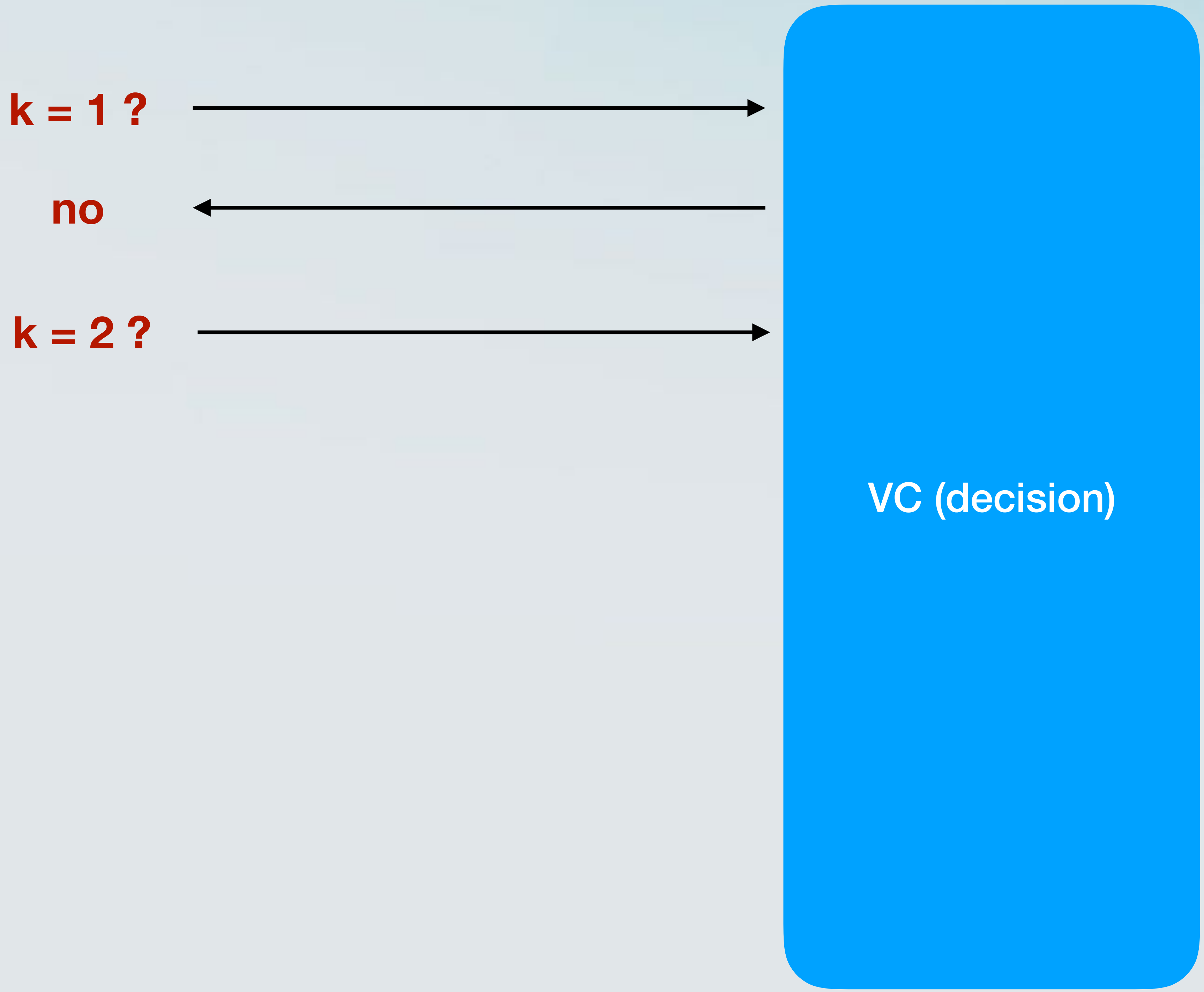
Vertex Cover Size



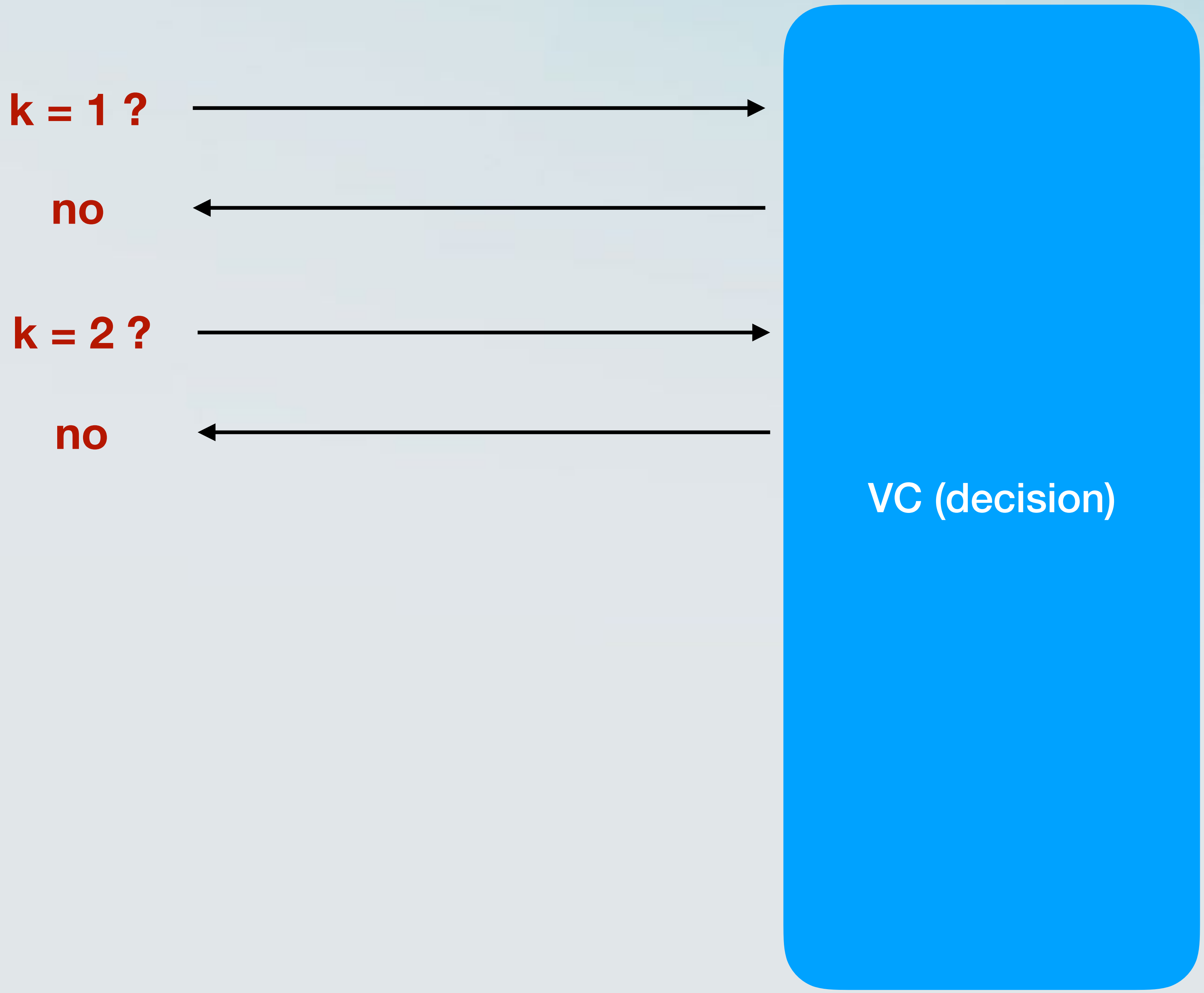
Vertex Cover Size



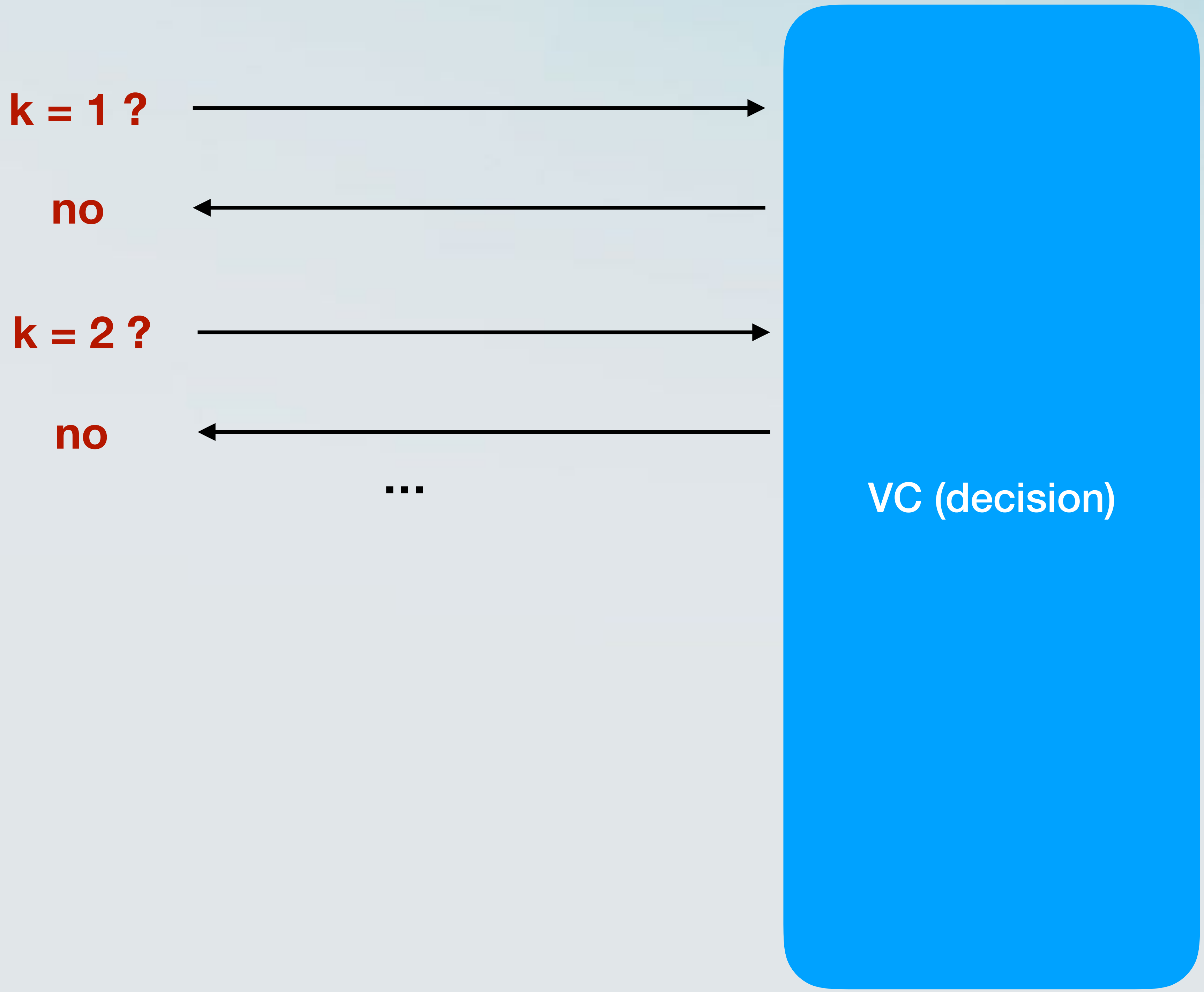
Vertex Cover Size



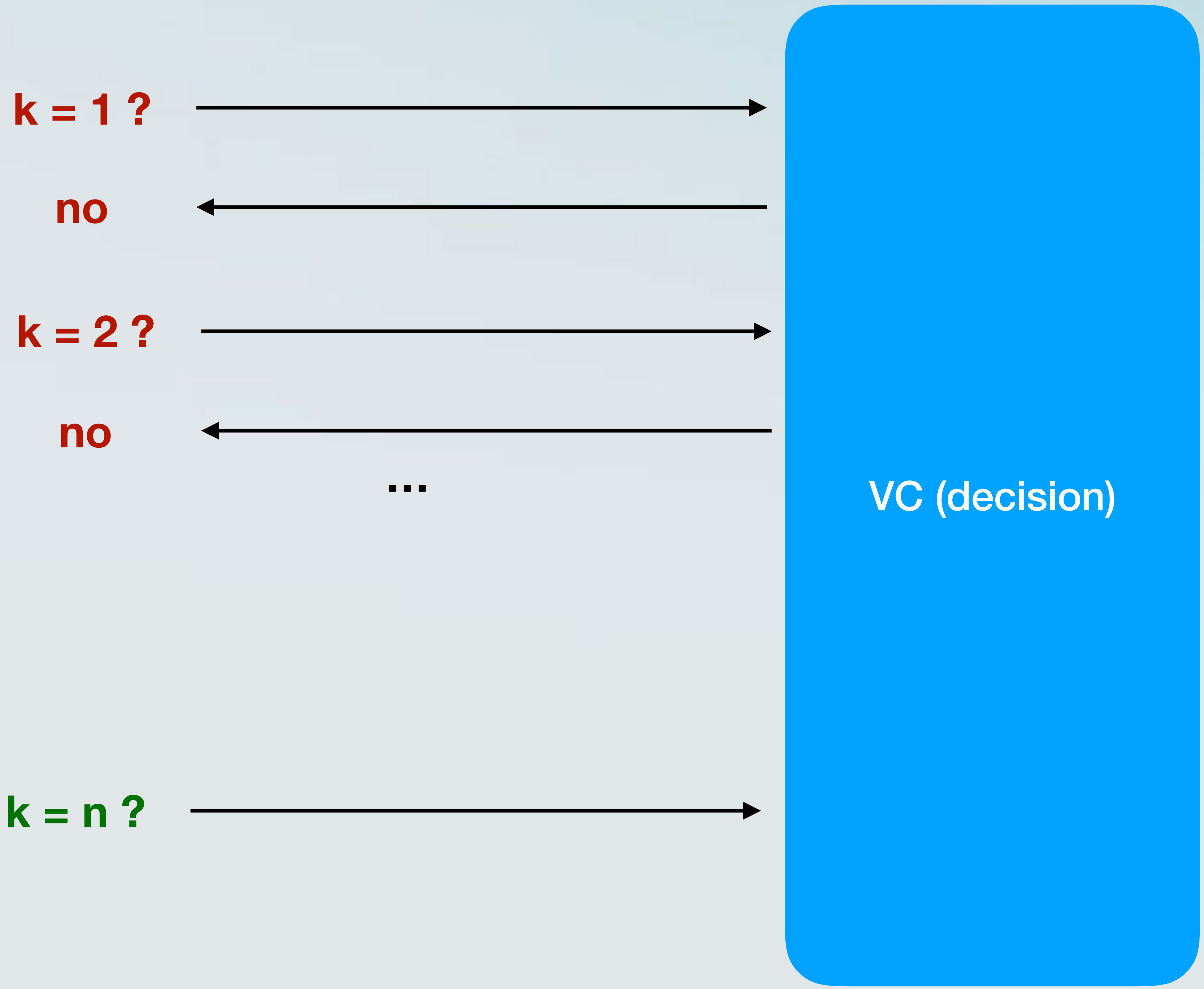
Vertex Cover Size



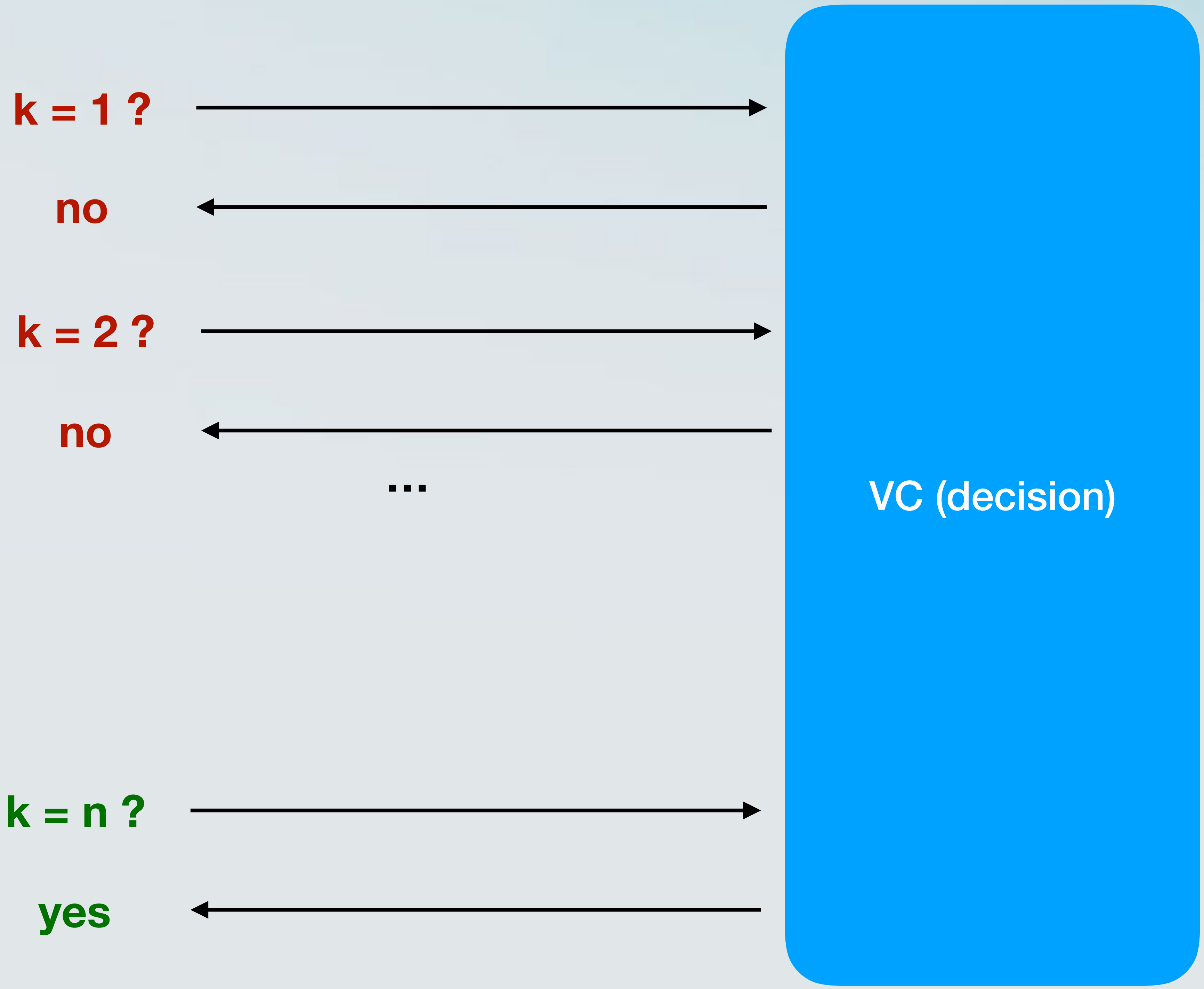
Vertex Cover Size



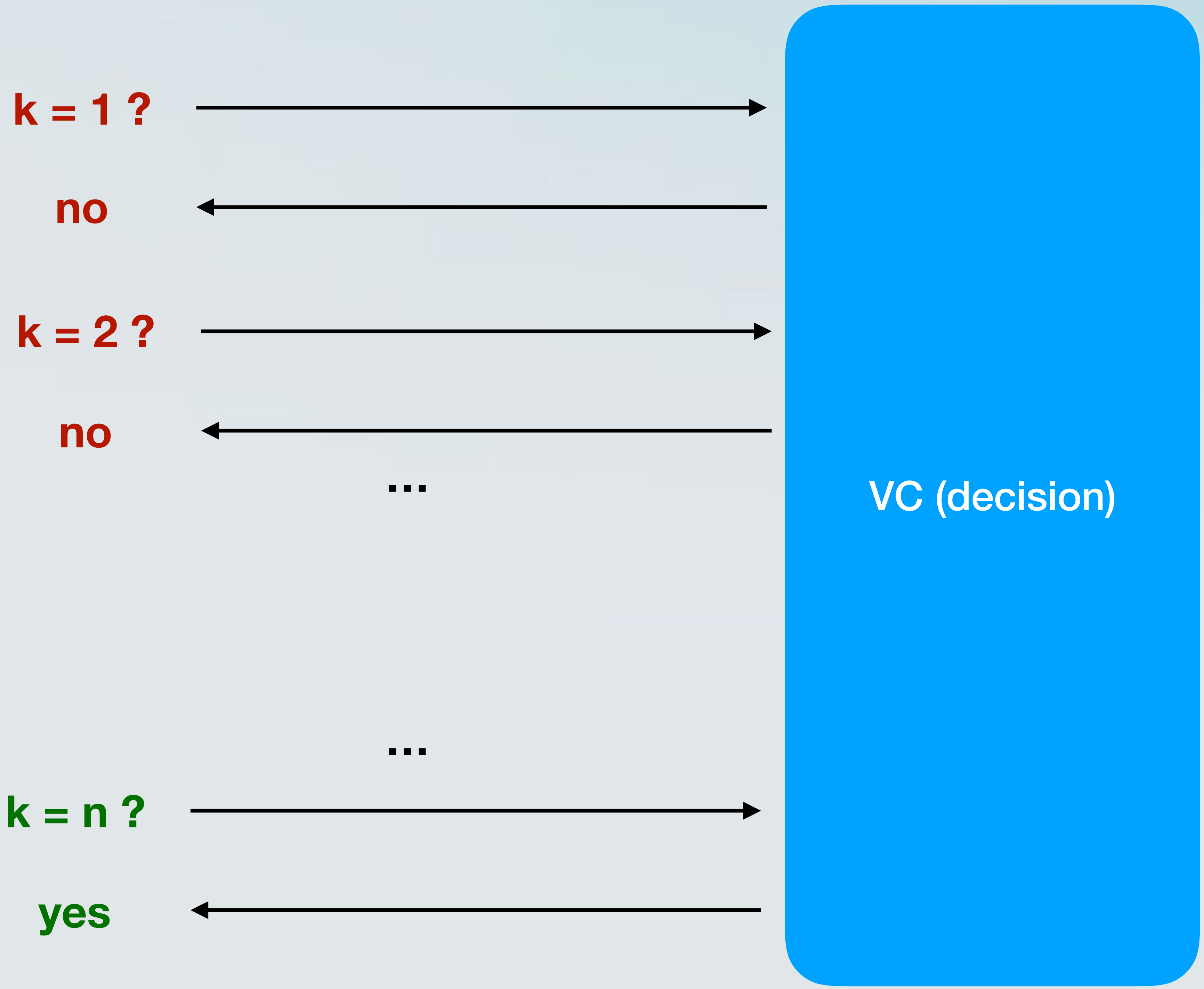
Vertex Cover Size



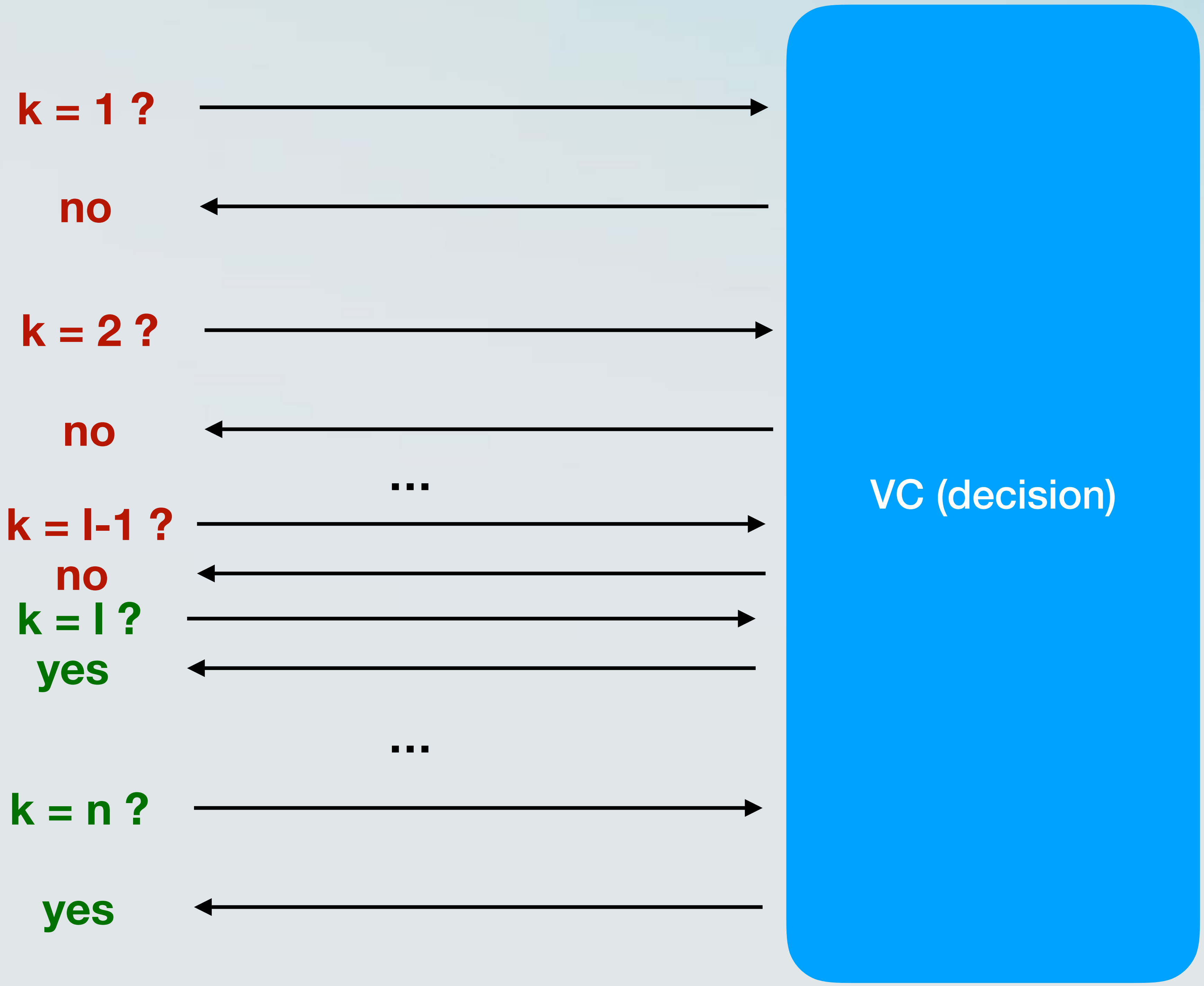
Vertex Cover Size



Vertex Cover Size



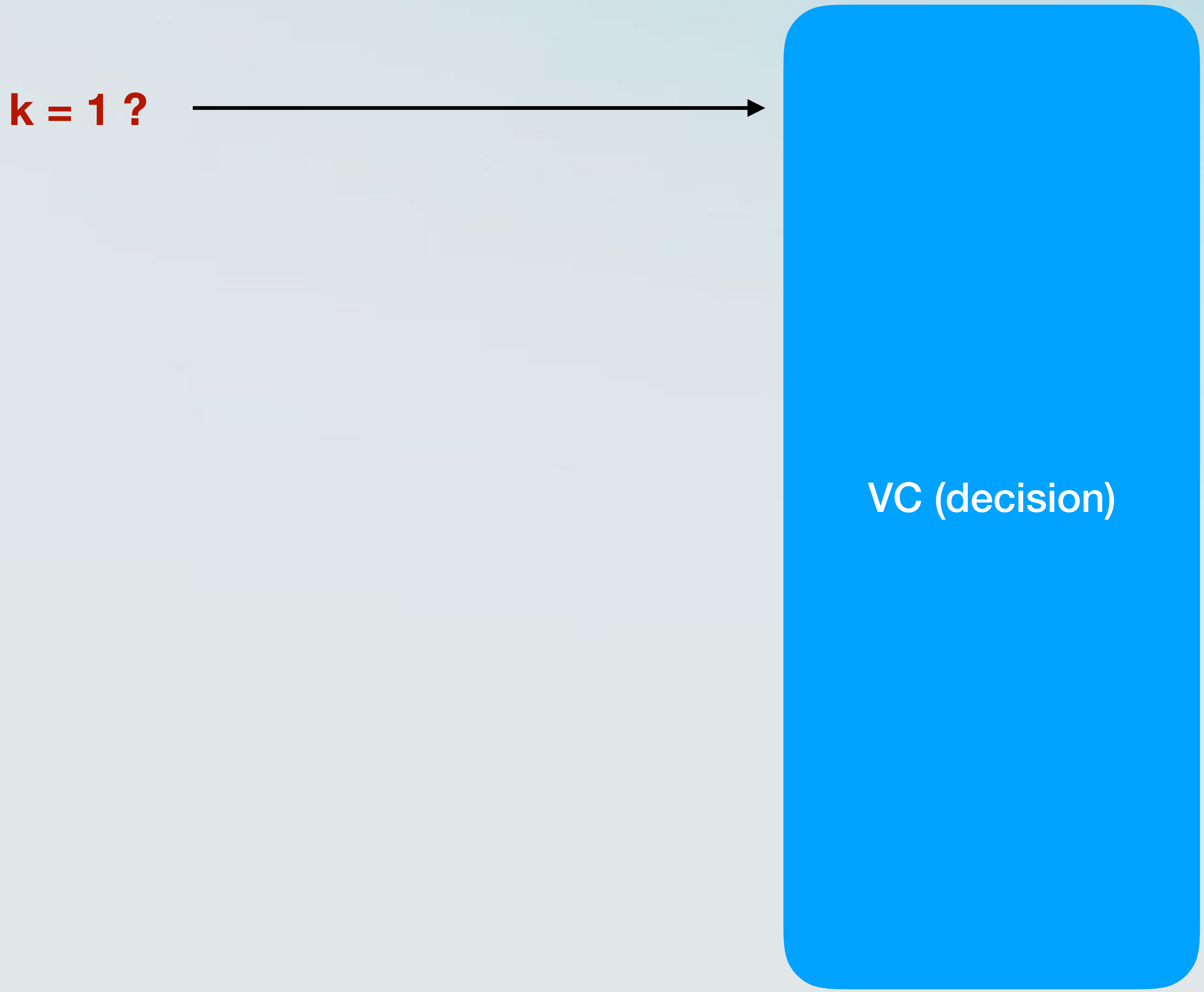
Vertex Cover Size



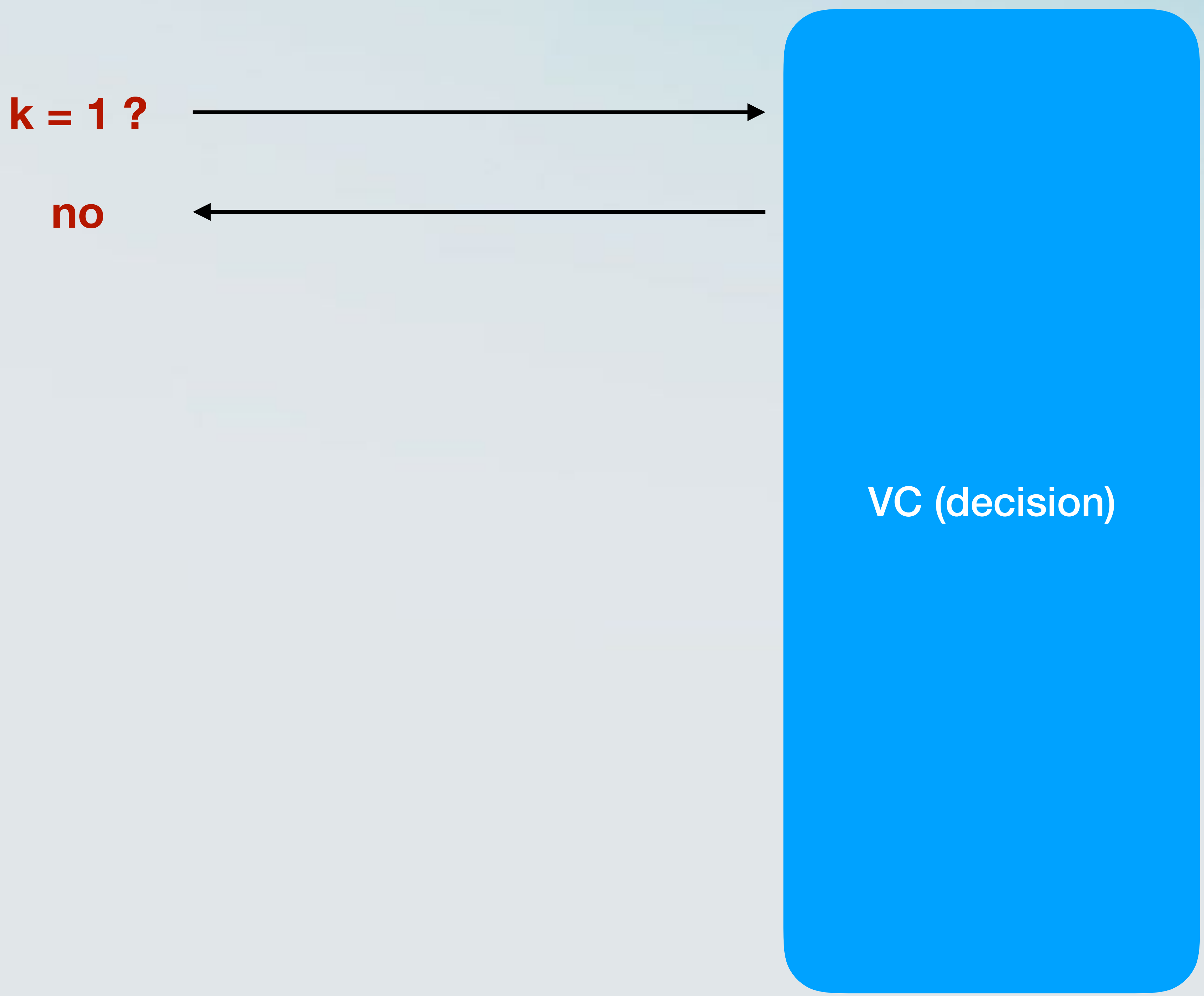
Vertex Cover Size

VC (decision)

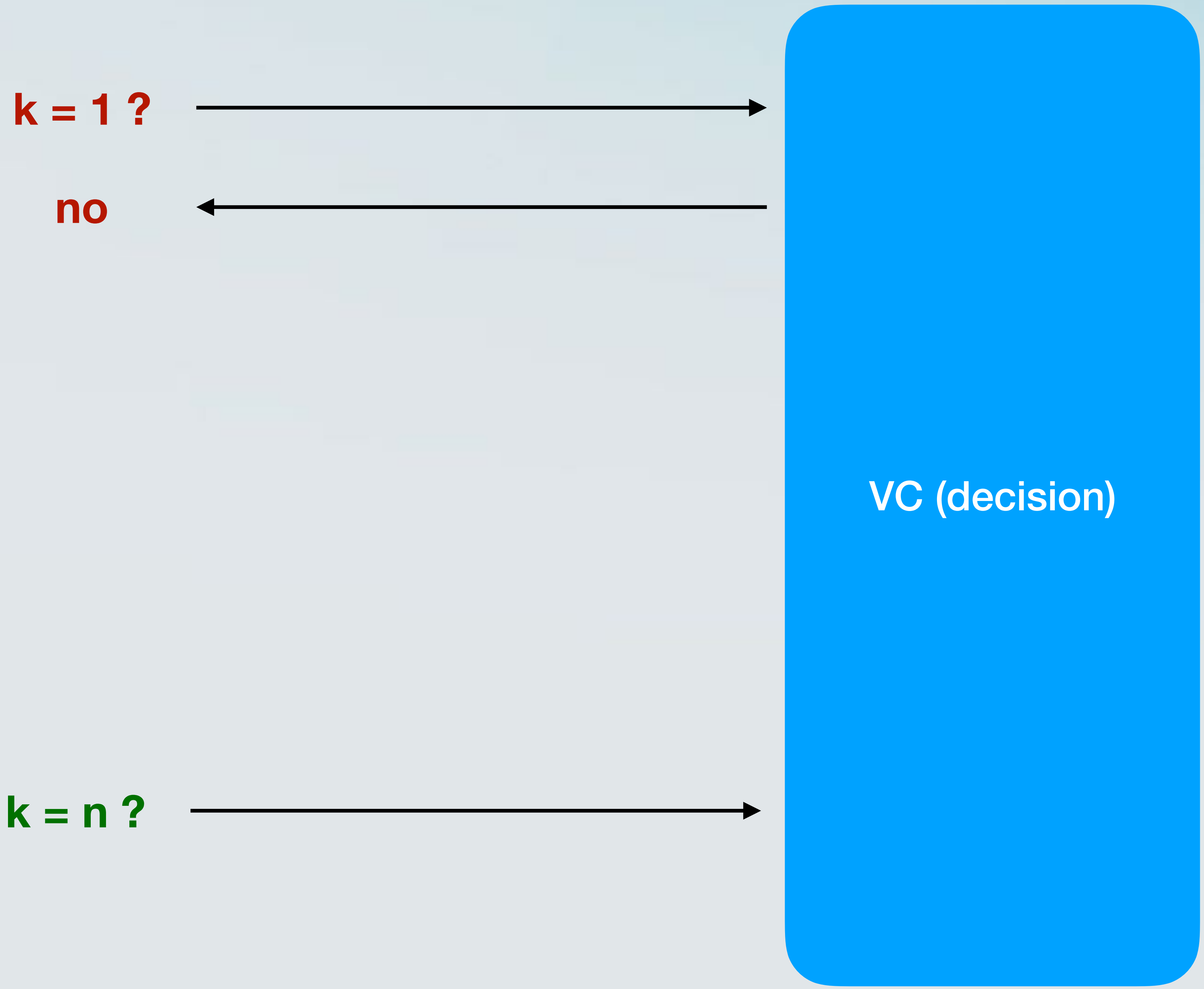
Vertex Cover Size



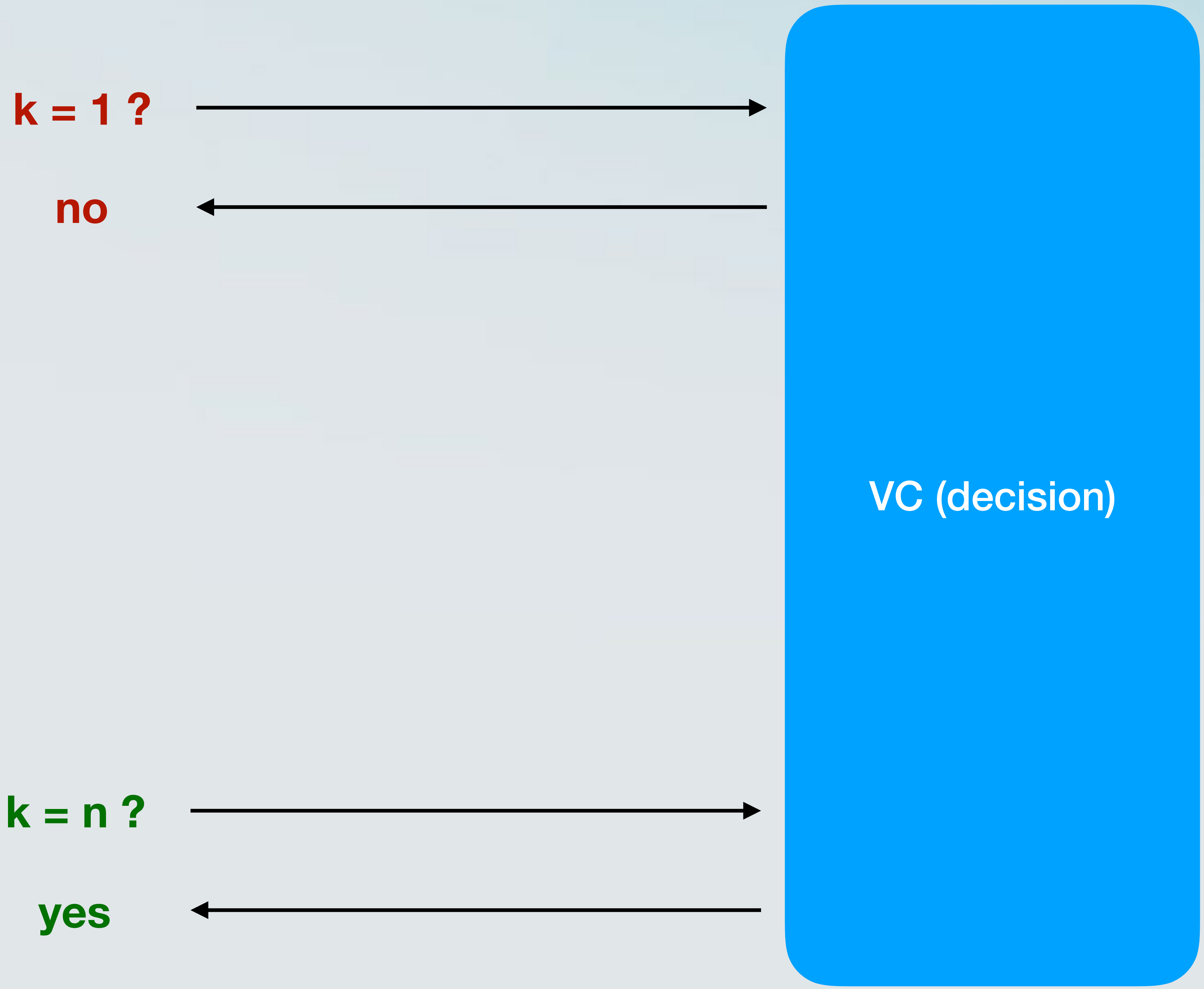
Vertex Cover Size



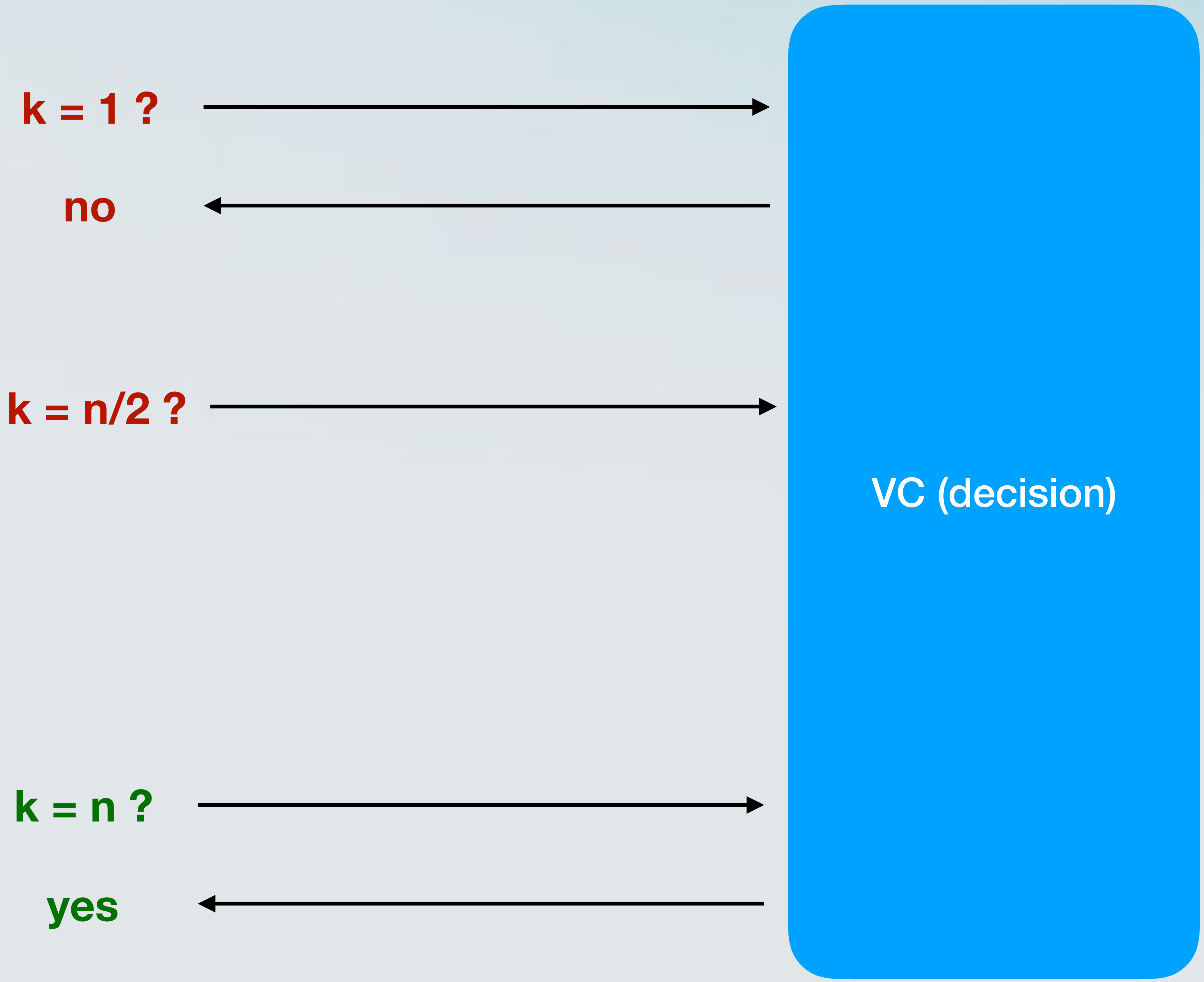
Vertex Cover Size



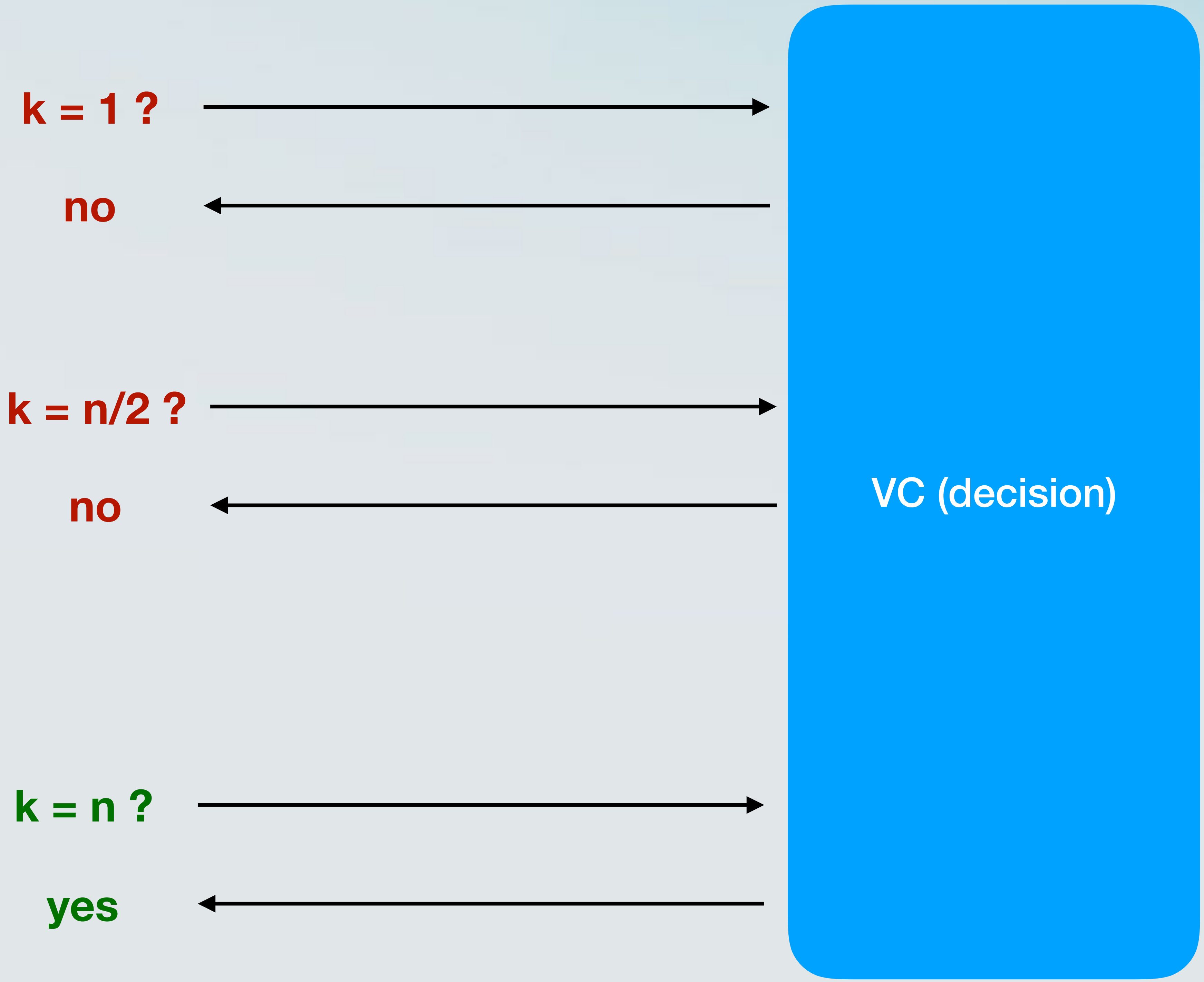
Vertex Cover Size



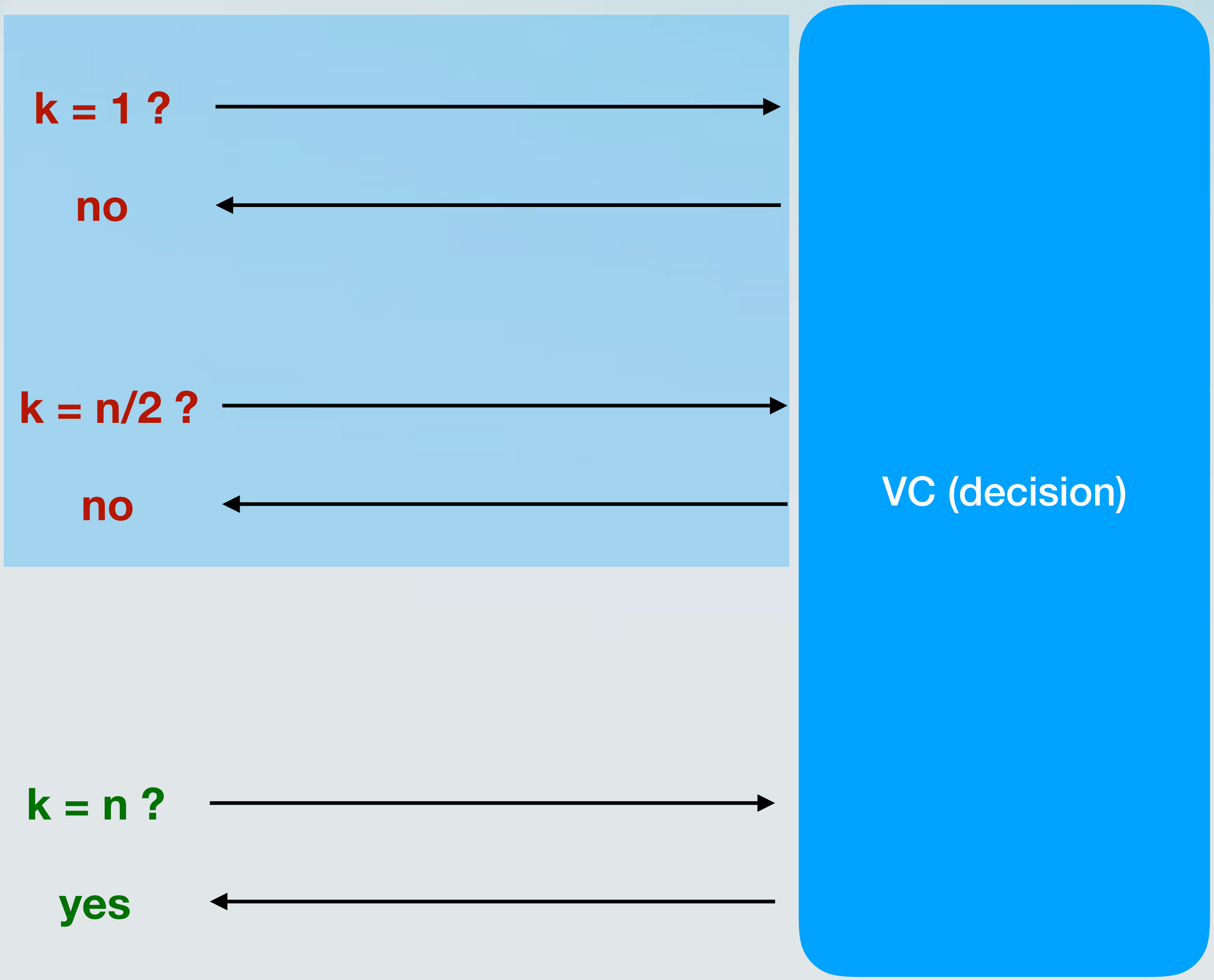
Vertex Cover Size



Vertex Cover Size



Vertex Cover Size



Optimisation vs decision

- Vertex Cover Size (Optimisation)
  Input: A graph $G=(V, E)$
  Output: **The size of** a minimum vertex cover.

- Vertex Cover (Decision)
  Input: A graph $G=(V, E)$ and a number k
  Output: Is there a vertex cover of size $\leq k$?.

Optimisation vs decision

- Vertex Cover (Optimisation)
  Input: A graph $G=(V, E)$
  Output: A minimum vertex cover.

- Vertex Cover (Decision)
  Input: A graph $G=(V, E)$ and a number k
  Output: Is there a vertex cover of size $\leq k$?.

Vertex Cover

- First, find the value k^* of the minimum vertex cover using the algorithm for VC_d.
- Pick a vertex v in the graph.
    - Remove it (and the incident edges) to get graph $G - \{v\}$.
    - Property: If v was in any minimum vertex cover, $G - \{v\}$ has a minimum vertex cover of size k^*-1.
    - Check if the graph $G - \{v\}$ has a vertex cover of size at most k^*-1.
        - Yes: Include v in the vertex cover.
        - No: Do not include v in the vertex cover.
        - Then move to the next vertex.

The subset sum problem

- We are given a set of n items {*1*, *2*, … , *n*}.
- Each item *i* has a non-negative integer weight w_i.
- We are given an integer bound W.
- Goal: Select a subset S of the items such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} w_i$ is maximised.

Equivalent formulation decision version

- We are given a set T of n items {*1*, *2*, … , *n*}.
- Each item *i* has a non-negative integer weight w_i.
- We are given an integer bound W.
- Goal: Decide if there exists a subset S of the items such that

$$\sum_{i \in S} w_i = W$$

Subset Sum is in NP

- If we are given a candidate solution S, we can easily check whether the following holds or not:

$$\sum_{i \in S} w_i = W$$

Subset Sum is in NP-hard

- We will reduce from 3SAT.
- Given a 3CNF formula ϕ (with m clauses and n variables) we will construct an instance <T, W> of the subset sum problem such that:
  - ϕ is satisfiable if any only if there exists a subset S of T whose sum is exactly W.
- Assumptions wlog:
  - Every variable appears in some clause.
  - A clause does not include both a variable and its negation.

The reduction

- We will create integers with $m+n$ digits that look like this:

$x_1\ x_2\ x_3 \ldots x_n \ldots C_1\ C_2 \ldots C_m$

The reduction

- We will create integers with m+n digits that look like this:

x_1 x_2 x_3 ... x_n ... C_1 C_2 ... C_m

Variables

The reduction

- We will create integers with m+n digits that look like this:

$x_1\ x_2\ x_3 \ldots x_n \ldots C_1\ C_2 \ldots C_m$

Variables Clauses

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

- We will create integers with $m+n$ digits that look like this:

  $x_1\ x_2\ x_3 \ldots x_n \ldots C_1\ C_2 \ldots C_m$

- We will set W to have *1* in all “variable digits” and *0* in all “clause digits”.

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
W		1	1	1	4	4	4	4

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

- We will create integers with $m+n$ digits that look like this:

  $x_1\ x_2\ x_3 \ldots x_n \ldots C_1\ C_2 \ldots C_m$

- We will set W to have *1* in all "variable digits" and *0* in all "clause digits".

- For each variable x_i, we create two integers z_i and y_i.

  - Each of them has *1* in the digit x_i and 0 in the other "variable digits".

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0				
y_1		1	0	0				
z_2		0	1	0				
y_2		0	1	0				
z_3		0	0	1				
y_3		0	0	1				
W		1	1	1	4	4	4	4

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

- We will create integers with $m+n$ digits that look like this:

  $x_1\ x_2\ x_3 \ldots x_n \ldots C_1\ C_2 \ldots C_m$

- We will set W to have *1* in all "variable digits" and *0* in all "clause digits".
- For each variable x_i, we create two integers z_i and y_i.
  - Each of them has *1* in the digit x_i and 0 in the other "variable digits".
  - If literal x_i appears in clause C_j, z_i contains a *1* in the corresponding "clause digit".
  - If literal $\neg x_i$ appears in clause C_j, y_i contains a *1* in the corresponding "clause digit".
  - All other "clause digits" are set to 0.

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
W		1	1	1	4	4	4	4

$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

The reduction

- We will create integers with $m+n$ digits that look like this:

  $x_1\ x_2\ x_3 \ldots x_n \ldots C_1\ C_2 \ldots C_m$

- We will set W to have *1* in all "variable digits" and *0* in all "clause digits".
- For each variable x_i, we create two integers z_i and y_i.
  - Each of them has *1* in the digit x_i and 0 in the other "variable digits".
  - If literal x_i appears in clause C_j, z_i contains a *1* in the corresponding "clause digit".
  - If literal $\neg x_i$ appears in clause C_j, y_i contains a *1* in the corresponding "clause digit".
  - All other "clause digits" are set to 0.

The reduction

- For each clause C_j we create two integers s_i and t_i.
  - These have 0 everywhere, except in the corresponding "clause digit".

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0		0	0	0
t_1		0	0	0		0	0	0
s_2		0	0	0	0		0	0
t_2		0	0	0	0		0	0
s_3		0	0	0	0	0		0
t_3		0	0	0	0	0		0
s_4		0	0	0	0	0	0	
t_4		0	0	0	0	0	0	
W		1	1	1	4	4	4	4

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

- For each clause C_j we create two integers s_i and t_i.
  - These have 0 everywhere, except in the corresponding "clause digit".
  - s_i has a 1 in the corresponding "clause digit".
  - t_i has a 2 in the corresponding "clause digit".

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Subset Sum is in NP-hard

- We will reduce from 3SAT.
- Given a 3CNF formula ϕ (with m clauses and n variables) we will construct an instance $\langle T, W \rangle$ of the subset sum problem such that:
  - ϕ is satisfiable if any only if there exists a subset S of T whose sum is exactly W.

Subset Sum is in NP-hard

- We will reduce from 3SAT.
- Given a 3CNF formula ϕ (with m clauses and n variables) we will construct an instance $<T, W>$ of the subset sum problem such that:
  - ϕ is satisfiable if any only if there exists a subset S of T whose sum is exactly W.
- One direction:

  ϕ is satisfiable => there exists a subset S of T whose sum is exactly W.

The reduction

- Let x be a satisfying assignment.
  - If $x_1 = 1$, include z_1.
  - Otherwise, include y_1.
  - Include appropriate choices of s_i and t_i.

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to *0* in *x*.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in *x*.

$$\boldsymbol{\phi} = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in *x*.

If we need either
1 more "*1*" or
2 more "*1*"s
to get to "*4*".

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in *x*.

If we need either 1 more "*1*" or 2 more "*1*"s to get to "*4*".

We can pick the appropriate s_i or t_i to make up the difference.

$$\phi = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in *x*.

If we need either 1 more "*1*" or 2 more "*1*"s to get to "*4*".

We can pick the appropriate s_i or t_i to make up the difference.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in *x*.

If we need either 1 more "*1*" or 2 more "*1*"s to get to "*4*".

We can pick the appropriate s_i or t_i to make up the difference.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in *x*.

If we need either 3 more "*1*s to get to "*4*".

We pick both s_i or t_i to make up the difference.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in *x*.

If we need either 3 more "*1*s to get to "*4*".

We pick both s_i or t_i to make up the difference.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

By the construction of z and y, the "variable digits" always sum up to *1111*

Since all clauses are satisfied, we get at least one "*1*" from one of the variables that were set to "*0*" in x.

If we need either 3 more "*1*s to get to "*4*".

We pick both s_i or t_i to make up the difference.

$$\phi = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)$$

Subset Sum is in NP-hard

- We will reduce from 3SAT.
- Given a 3CNF formula ϕ (with m clauses and n variables) we will construct an instance $<T, W>$ of the subset sum problem such that:
  - ϕ is satisfiable if any only if there exists a subset S of T whose sum is exactly W.

Subset Sum is in NP-hard

- We will reduce from 3SAT.
- Given a 3CNF formula ϕ (with m clauses and n variables) we will construct an instance $\langle T, W \rangle$ of the subset sum problem such that:
  - ϕ is satisfiable if any only if there exists a subset S of T whose sum is exactly W.
- Other direction:

  there exists a subset S of T whose sum is exactly $W \Rightarrow \phi$ is satisfiable.

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

$$\boldsymbol{\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)}$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

Set x_i = *1* if S contains z_i and x_i = *0* otherwise.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

Set $x_i = 1$ if S contains z_i and $x_i = 0$ otherwise.

Consider an arbitrary clause C_j.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

Set $x_i = 1$ if S contains z_i and $x_i = 0$ otherwise.

Consider an arbitrary clause C_j.

Consider the corresponding “clause digits”.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

Set x_i = *1* if S contains z_i and x_i = *0* otherwise.

Consider an arbitrary clause C_j.

Consider the corresponding "clause digits".

Since these add up to "*4*", it must receive at least "*1*" from the z or y numbers.

$$\boldsymbol{\phi} = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

Set $x_i = 1$ if S contains z_i and $x_i = 0$ otherwise.

Consider an arbitrary clause C_j.

Consider the corresponding "clause digits".

Since these add up to "*4*", it must receive at least "*1*" from the *chosen* z or y numbers.

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

The reduction

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	**1**	0	0	0
t_1		0	0	0	**2**	0	0	0
s_2		0	0	0	0	**1**	0	0
t_2		0	0	0	0	**2**	0	0
s_3		0	0	0	0	0	**1**	0
t_3		0	0	0	0	0	**2**	0
s_4		0	0	0	0	0	0	**1**
t_4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

S must contain *exactly one* of z and y for each index *i*, otherwise the sum is not W.

Set $x_i = 1$ if S contains z_i and $x_i = 0$ otherwise.

Consider an arbitrary clause C_j.

Consider the corresponding "clause digits".

Since these add up to "*4*", it must receive at least "*1*" from the *chosen* z or y numbers.

This implies that the clause is satisfied.

$$\phi = (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)$$

Knapsack

- 0/1-Knapsack is also NP-complete.
  - Define the decision problem, containment is easy to see.
  - How do we prove hardness?
  - Which problem should we reduce from?

NP-complete problems

NP-complete problems

- Independent Set in graph G: A set of nodes in the graph, such that there is no edge between any two nodes in the set.

NP-complete problems

- Independent Set in graph G: A set of nodes in the graph, such that there is no edge between any two nodes in the set.

- Maximum Independent Set
  Given a graph G, find an independent set of maximum size.

NP-complete problems

- Independent Set in graph G: A set of nodes in the graph, such that there is no edge between any two nodes in the set.

- Maximum Independent Set
Given a graph G, find an independent set of maximum size.

- Maximum Independent Set (decision version)
Given a graph G, and an integer k, is there an independent set of size at least k?

NP-complete problems

NP-complete problems

- Set Packing
  Given a set U of elements, a collection $S_1, \dots , S_m$ of subsets of U and a number k, does there exist a collection of at least k of these sets such that no two of them intersect?

NP-complete problems

- Set Packing
  Given a set U of elements, a collection $S_1, \ldots, S_m$ of subsets of U and a number k, does there exist a collection of at least k of these sets such that no two of them intersect?

- Set Cover
  Given a set U of elements, a collection $S_1, \ldots, S_m$ of subsets of U and a number k, does there exist a collection of at most k of these sets whose union is equal to U?

NP-complete problems

NP-complete problems

- 3-Dimensional Matching
  Given disjoint sets X, Y and Z each of size n, and given a set T (which is a subset of $X \times Y \times Z$) of ordered triples, does there exist a set of n triples in T, so that each element of $X \cup Y \cup Z$ is contained in exactly in one of these triples?

NP-complete problems

NP-complete problems

- k-Colouring of a graph G: A function f: V → {*1*, …, *k*} so that for every edge (u, v) we have that f(u) ≠ f(v).

NP-complete problems

- k-Colouring of a graph G: A function f: V → {*1*, …, *k*} so that for every edge (u, v) we have that f(u) ≠ f(v).
- 3-Colouring
  Given a graph G, does it have a 3-Colouring?

Interlude

- k-Colouring of a graph G: A function f: $V \rightarrow \{1, \ldots, k\}$ so that for every edge (u, v) we have that $f(u) \neq f(v)$.
- 3-Colouring
  Given a graph G, does it have a 3-Colouring?
- What about 2-Colouring? Is it NP-complete?

NP-complete problems

NP-complete problems

- Hamiltonian cycle in a directed graph G: A cycle in a directed graph that visits each vertex *exactly once*.

NP-complete problems

- Hamiltonian cycle in a directed graph G: A cycle in a directed graph that visits each vertex *exactly once*.
- Hamiltonian path in a directed graph G: A path in a directed graph that contains each vertex *exactly once*.

NP-complete problems

- Hamiltonian cycle in a directed graph G: A cycle in a directed graph that visits each vertex *exactly once*.

- Hamiltonian path in a directed graph G: A path in a directed graph that contains each vertex *exactly once*.

- Hamiltonian Cycle
  Given a directed graph G, does it have a Hamiltonian Cycle?

NP-complete problems

- Hamiltonian cycle in a directed graph G: A cycle in a directed graph that visits each vertex *exactly once*.

- Hamiltonian path in a directed graph G: A path in a directed graph that contains each vertex *exactly once*.

- Hamiltonian Cycle
  Given a directed graph G, does it have a Hamiltonian Cycle?

- Hamiltonian Path
  Given a directed graph G, does it have a Hamiltonian Path?

NP-complete problems

- Hamiltonian cycle in a directed graph G: A cycle in a directed graph that visits each vertex *exactly once*.
- Hamiltonian path in a directed graph G: A path in a directed graph that contains each vertex *exactly once*.
- Hamiltonian Cycle
  Given a directed graph G, does it have a Hamiltonian Cycle?
- Hamiltonian Path
  Given a directed graph G, does it have a Hamiltonian Path?
- Traveling Salesman
  (def Kleinberg and Tardos, p. 474).

NP-completeness, a taxonomy

Packing problems

- Independent Set
- Set Packing

Covering problems

- Vertex Cover
- Set Cover

Partitioning problems

- 3D-Matching
- Graph Colouring

Sequencing problems

- Hamiltonian Cycle
- Hamiltonian Path
- Traveling Salesman

Numerical problems

- Subset Sum
- Knapsack

Constraint Satisfaction problems

- 3 SAT