Enumeration algorithms

M2 ORCO – Grenoble

Aurélie Lagoutte

November 9, 2022

Enumeration : principle

Some problems need as an answer a **list** of solutions, instead of a **single** solution. For example :

Enumeration : principle

Some problems need as an answer a **list** of solutions, instead of a **single** solution. For example :

- Answer to a database query

```
$ select appellation, vignoble, type from AOC
Côte-Rôtie                  |   Vallée du Rhône   |   Rouge
Saint-Emilion               |   Bordeaux          |   Rouge
Saint-Nicolas-de-Bourgueil  |   Val de Loire      |   Rouge
```

Enumeration : principle

Some problems need as an answer a **list** of solutions, instead of a **single** solution. For example :

- Answer to a database query

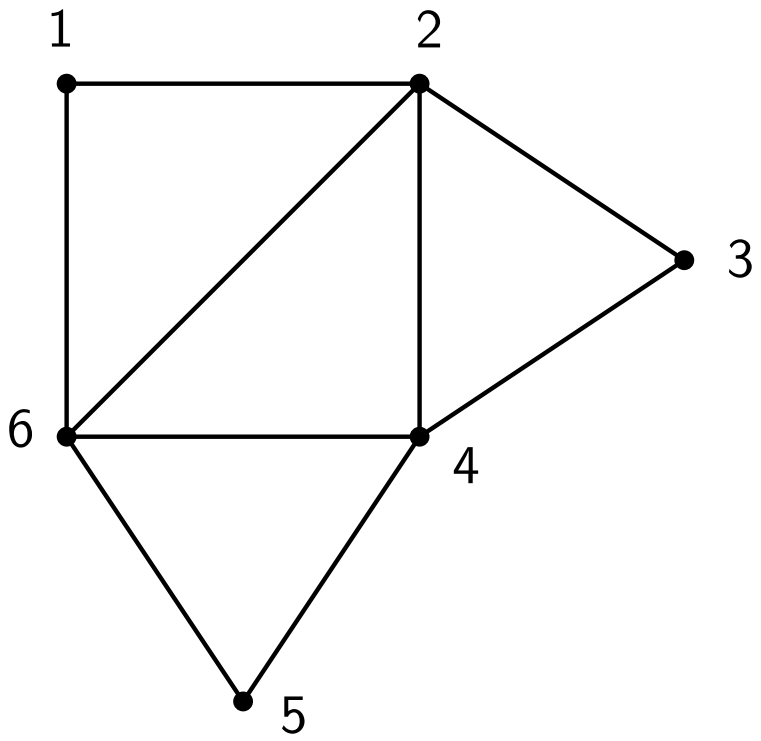
```
$ select appellation, vignoble, type from AOC
Côte-Rôtie                    |   Vallée du Rhône   |   Rouge
Saint-Emilion                 |   Bordeaux          |   Rouge
Saint-Nicolas-de-Bourgueil    |   Val de Loire      |   Rouge
```

- Truth table : list all input combinations

e_1	e_2	e_3	S
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Enumeration : a typical example

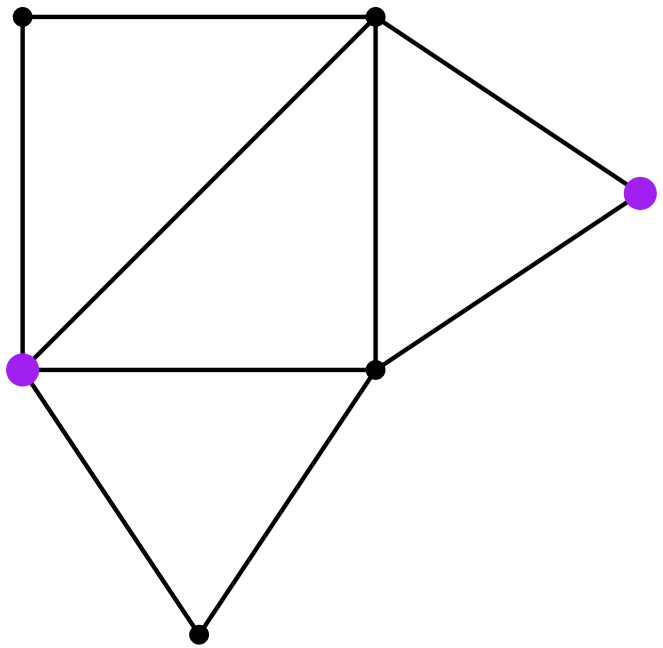
Input: Graph G
Output : The list of all **inclusion-wise maximal** stable sets of G



$\{1, 3, 5\}, \{1, 4\}, \{2, 5\}, \{3, 6\}$

Focus on easy problems

Input: Graph G
Output : one **inclusion-wise maximal** stable set. $\in$ P (greedy)



Not to be confused with :
Input: Graph G NP-complete
Output : a stable set of maximum size

Enumerating in graphs : useful cases

- Graph databases : answer to a query
- Graph model is not exact : some solutions are *best* based on qualitative criteria, we have to examine them one by one
- Identify all problematic (or interesting !) patterns in a network

Application fields: bioinformatics (phylogenetic trees), chemistry (molecule structure), complex system modeling, databases...

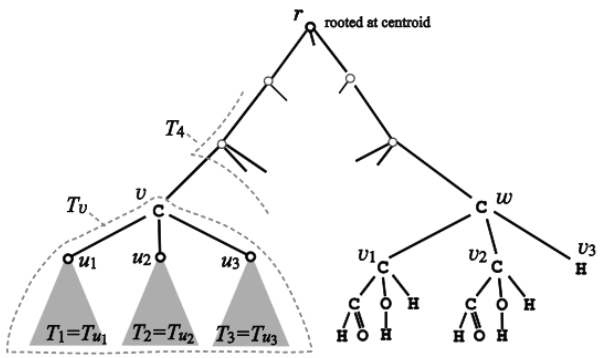


Diagram of a stereoisomer[1]

[1] Comparison and Enumeration of Chemical Graphs, T. Akutsu, H. Nagamochi, *Comp. and Struct. Biotechnology Journal*

Complexity for enumeration problems

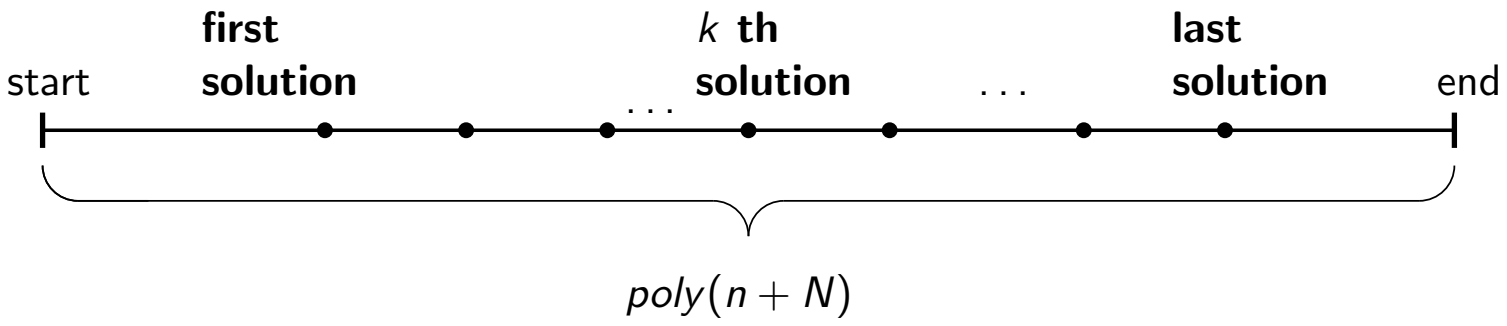
In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

1. Output-polynomial

Input of size n, N solutions to output.

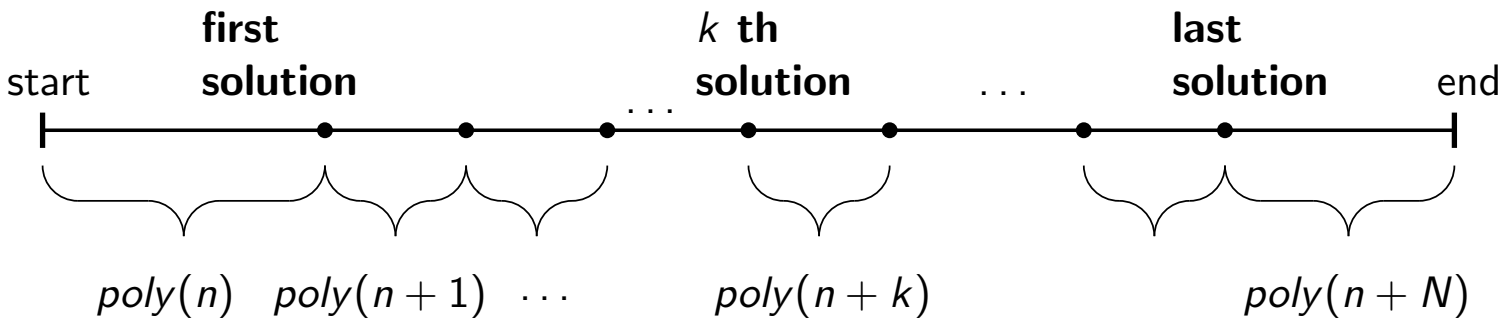


Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

1. Output-polynomial
2. Incremental polynomial

Input of size n, N solutions to output.

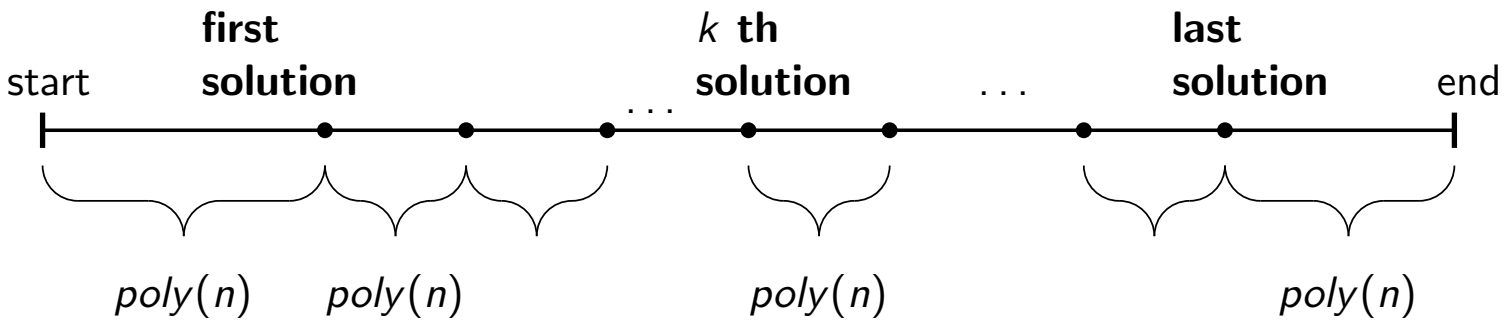


Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

1. Output-polynomial
2. Incremental polynomial
3. Polynomial delay

Input of size n, N solutions to output.



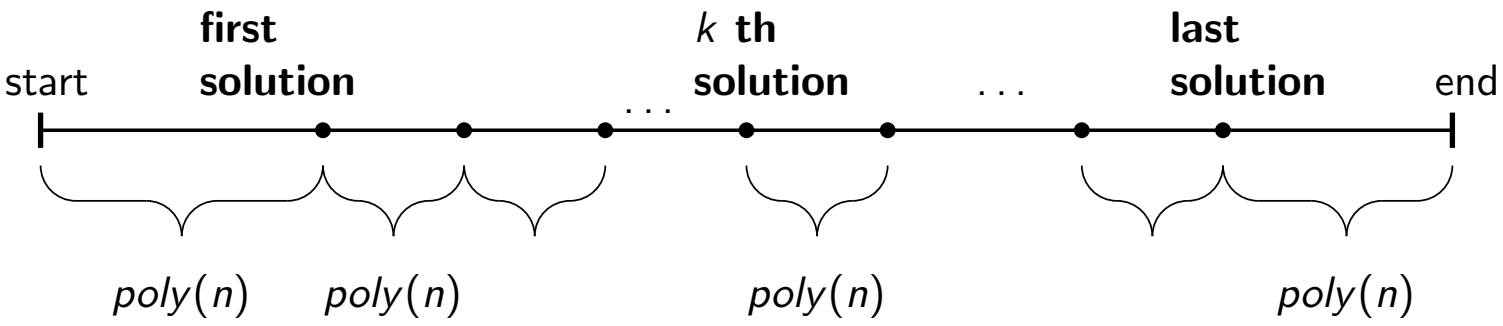
Complexity for enumeration problems

In most cases : exponential number of solutions to output
(ex: $3^{n/3}$ max. stable sets)
$\Rightarrow$ *Good complexity measure?*

1. Output-polynomial
2. Incremental polynomial
3. Polynomial delay

poly space vs.
exponential space

Input of size n, N solutions to output.



Interesting objects to enumerate

1. Inclusion-wise minimal transversal of a hypergraph

Interesting objects to enumerate

1. Inclusion-wise minimal transversal of a hypergraph
2. Inclusion-wise minimal dominating sets

Interesting objects to enumerate

1. Inclusion-wise minimal transversal of a hypergraph
2. Inclusion-wise minimal dominating sets
3. Spanning trees

Interesting objects to enumerate

1. Inclusion-wise minimal transversal of a hypergraph
2. Inclusion-wise minimal dominating sets
3. Spanning trees
4. "Structured patterns" : inclusion-wise max. stable sets or cliques ...

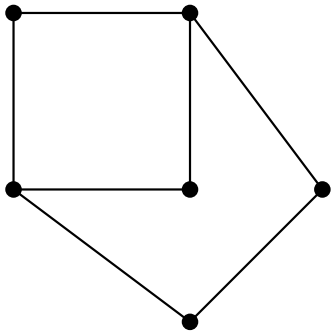
Interesting objects to enumerate

1. Inclusion-wise minimal transversal of a hypergraph
2. Inclusion-wise minimal dominating sets
3. Spanning trees
4. "Structured patterns" : inclusion-wise max. stable sets or cliques ...
5. Inclusion-wise minimal "Π-fixings" of a graph
   $\rightarrow$ Completions, deletion, induced subgraphs of a graph ...
   ... satisfying a given property Π

Minimal fixings

3 variants

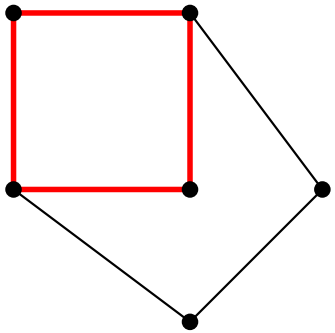
We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



Minimal fixings

3 variants

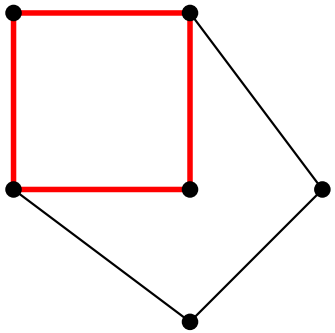
We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



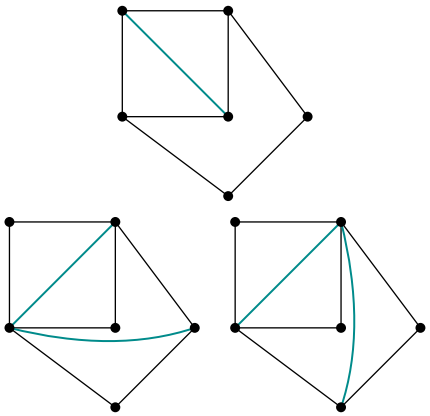
Minimal fixings

3 variants

We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



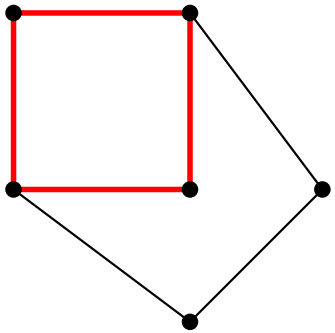
Fixing by
adding edges
Min. Π-completion



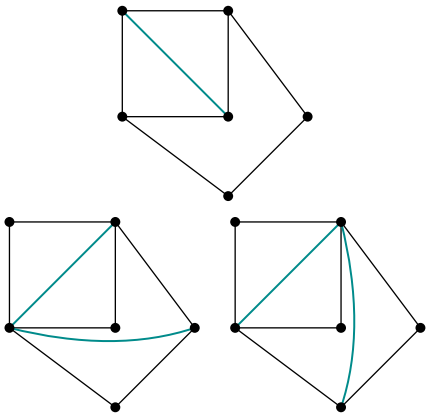
Minimal fixings

3 variants

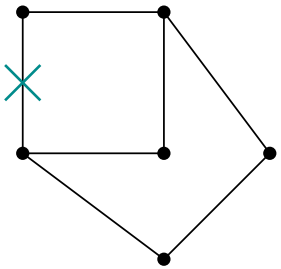
We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



Fixing by
adding edges
Min. Π-completion



Fixing by
deleting edges
Min. Π-deletion

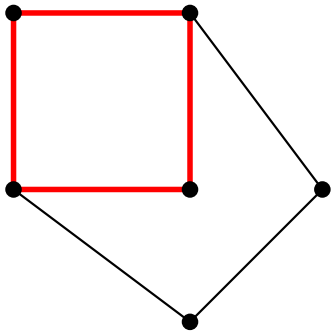


+ 3 others

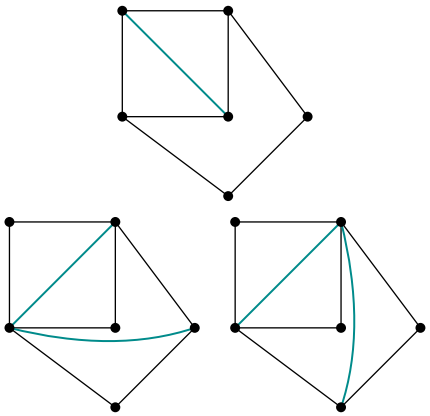
Minimal fixings

3 variants

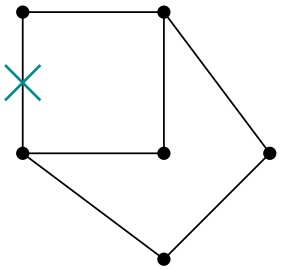
We want to satisfy a given property Π
Example : $\Pi = C_4$-free
(contains no induced C_4)



Fixing by adding edges
Min. Π-completion

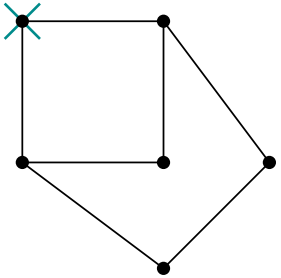


Fixing by deleting edges
Min. Π-deletion



+ 3 others

Fixing by deleting vertices
Max. Π-induced subgraph



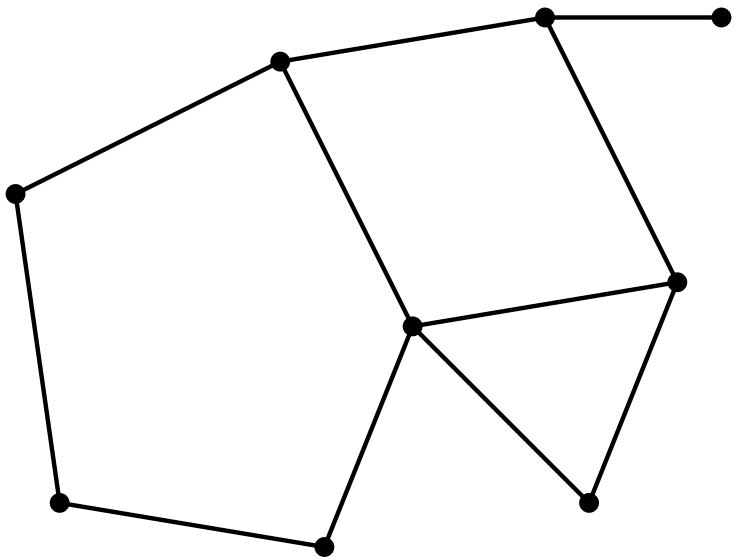
+ 3 others

Chordal completion

Given a graph $G = (V, E)$, a *chordal completion* of G is a completion of G that is chordal.

Sometimes we refer to the chordal completion as a *fill-in* F where F denotes the set of non-edges of G that must be turned into edges so that $(V, E \uplus F)$ is chordal.

A chordal completion of G is also called a *triangulation* of G.

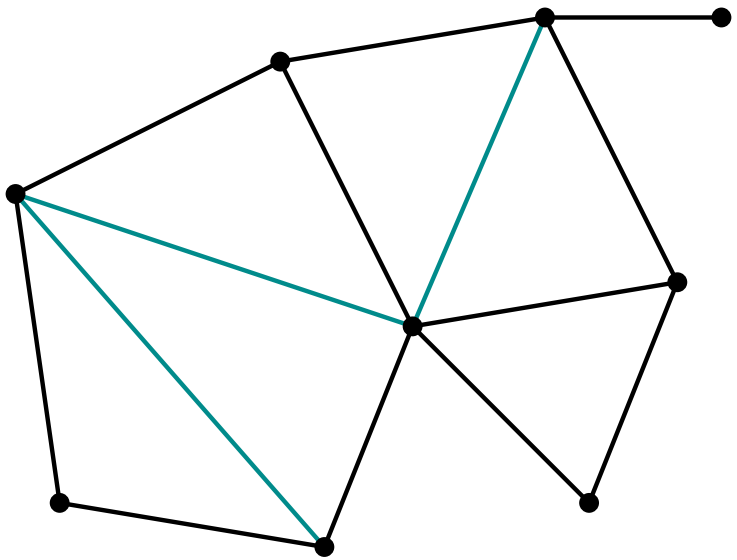


Chordal completion

Given a graph $G = (V, E)$, a *chordal completion* of G is a completion of G that is chordal.

Sometimes we refer to the chordal completion as a *fill-in* F where F denotes the set of non-edges of G that must be turned into edges so that $(V, E \uplus F)$ is chordal.

A chordal completion of G is also called a *triangulation* of G.



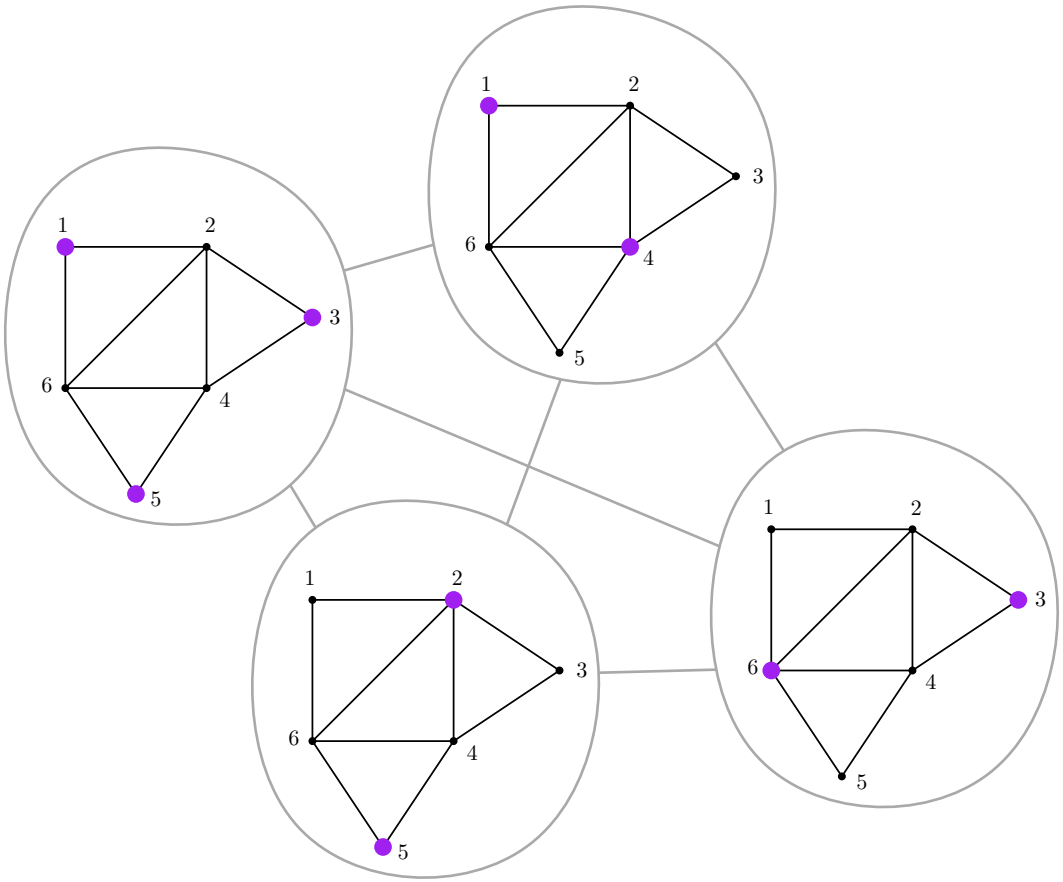
Algorithmic methods to enumerate

(Un petit quiz avant d'enchaîner ?
`http://wooclap.com/M2ORCRO`)

General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph

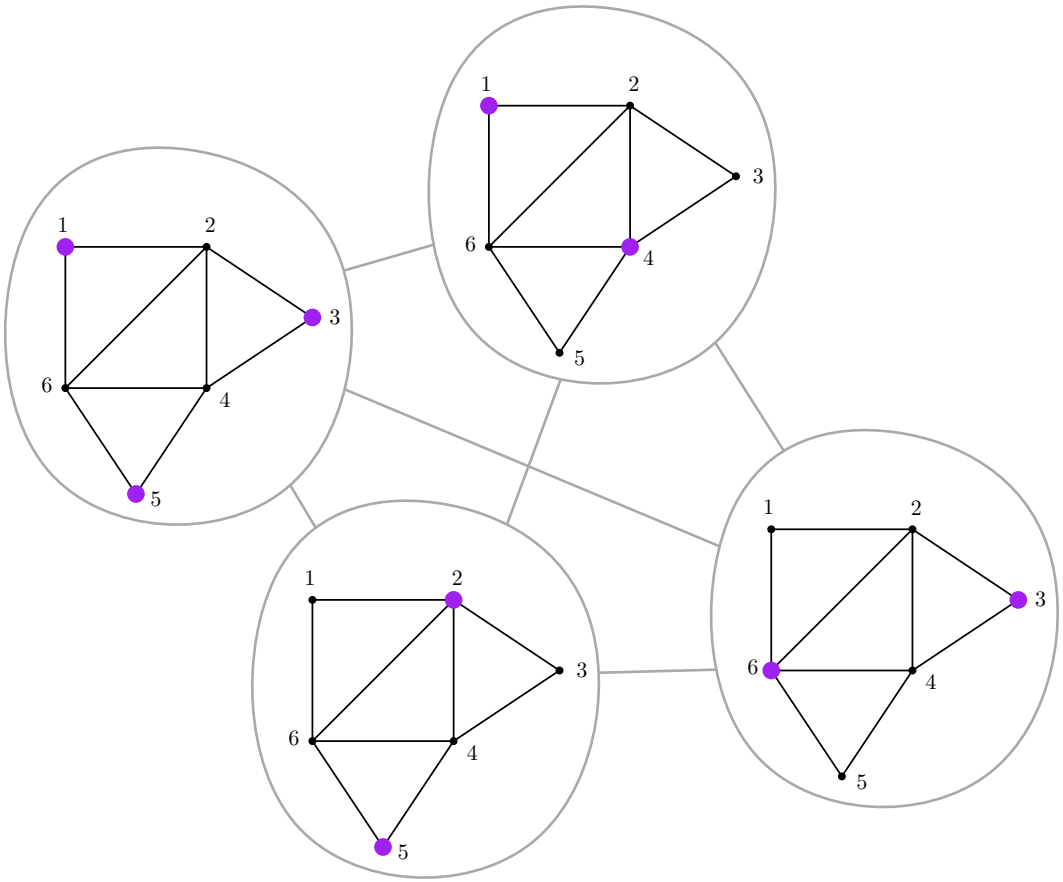
Example with
maximal stable sets
Solution metagraph



General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph
- outputting each solution once

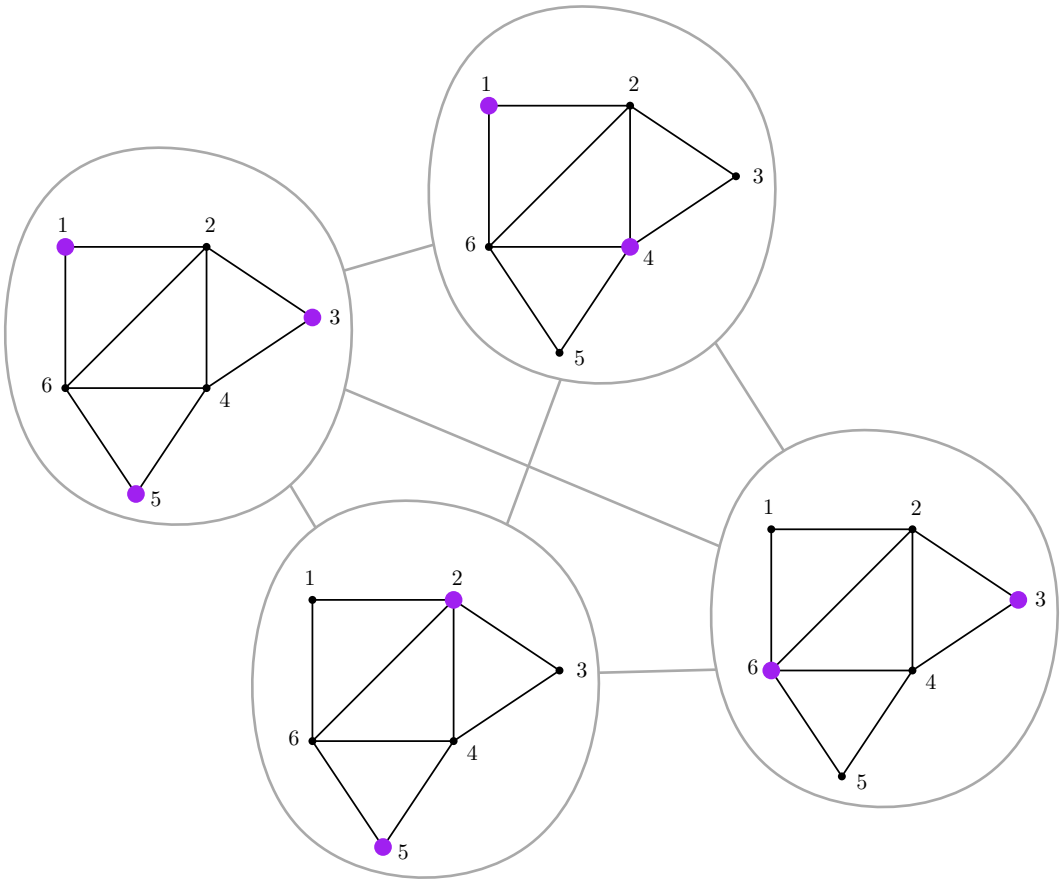
Example with
maximal stable sets
Solution metagraph



General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph
- outputting each solution once
- and **only** once

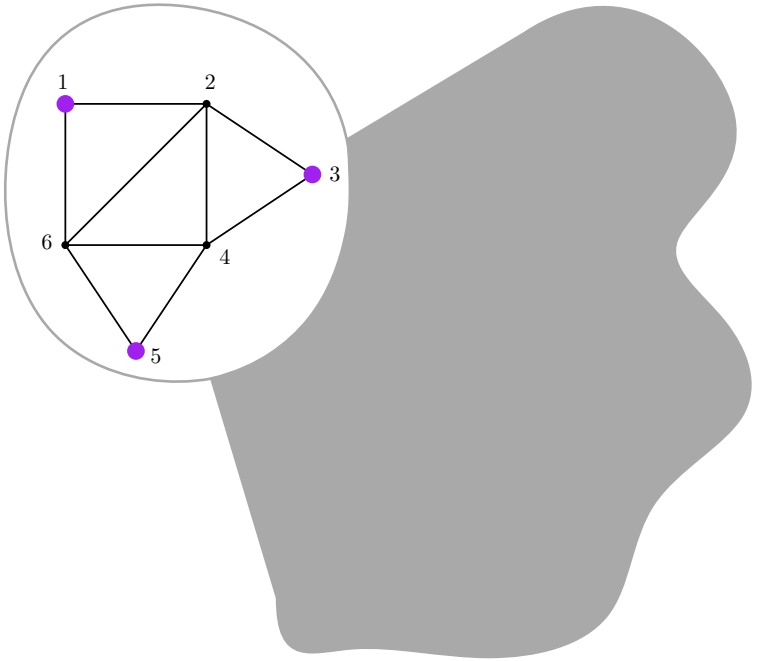
Example with
maximal stable sets
Solution metagraph



General principle of an enumeration algorithm

- Metagraph of solutions : traversal of this metagraph
- outputting each solution once
- and **only** once

Example with
maximal stable sets
Solution metagraph



Three classical methods :

Goal : get polynomial delay
+ poly space for 1 et 2 (and sometimes 3)

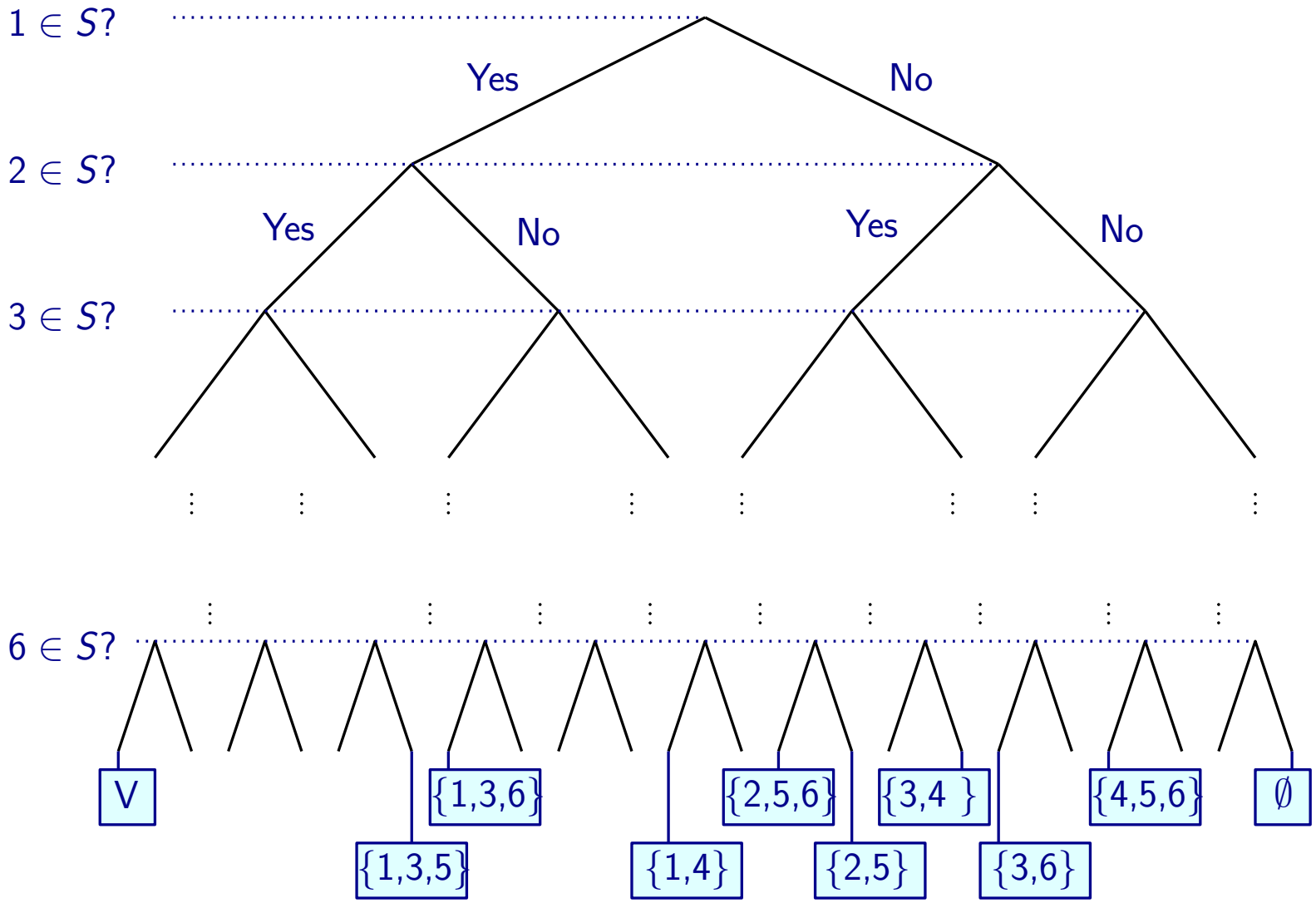
1. *Flashlight search* or *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

Three classical methods :

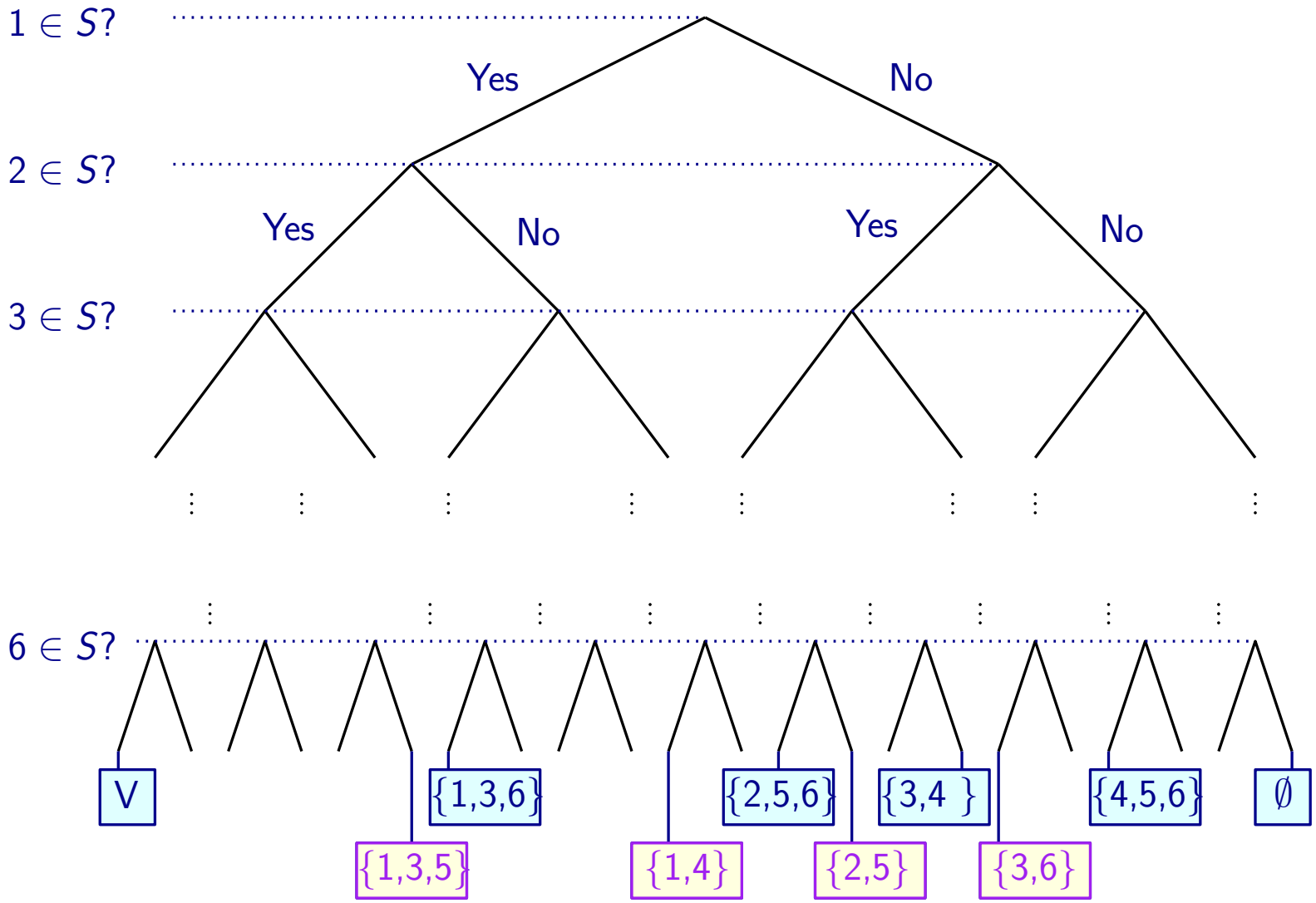
Goal : get polynomial delay
+ poly space for 1 et 2 (and sometimes 3)

1. ***Flashlight search*** or *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

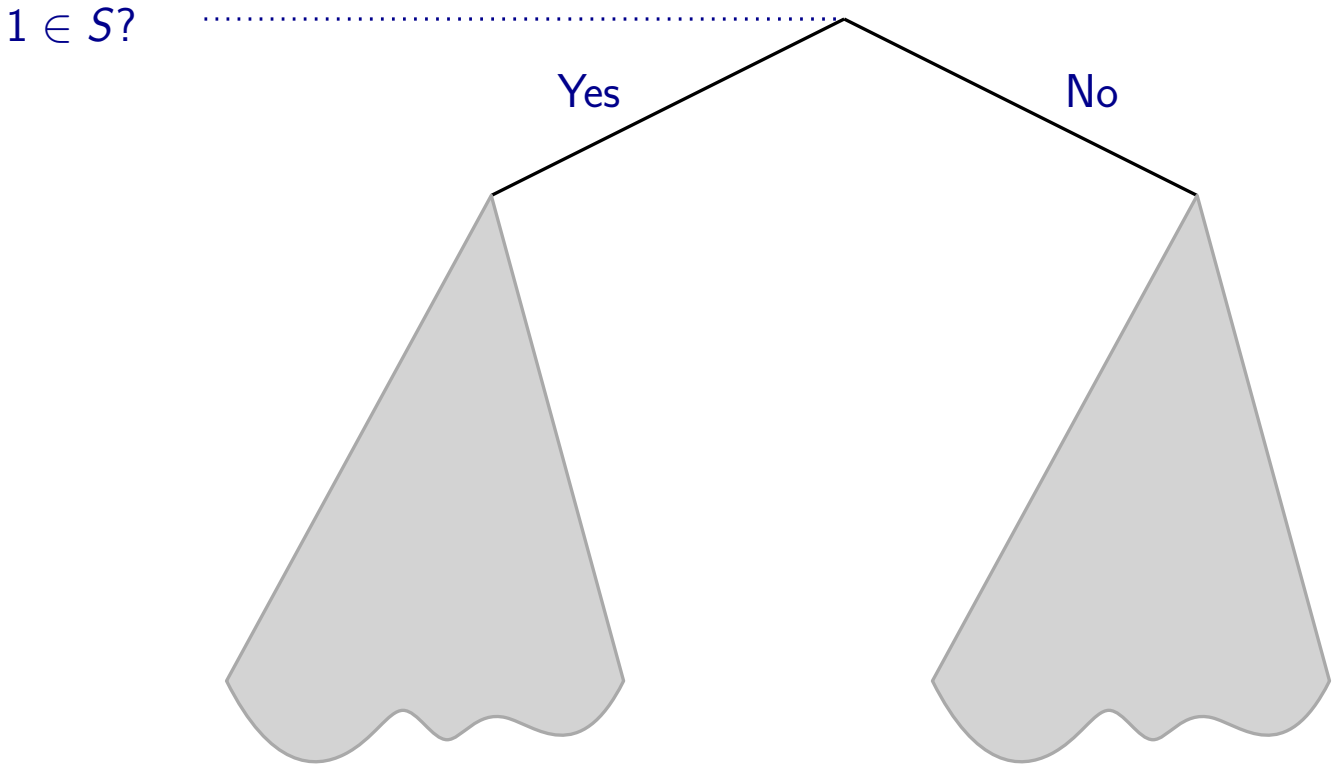
Flashlight search or *Binary partition*



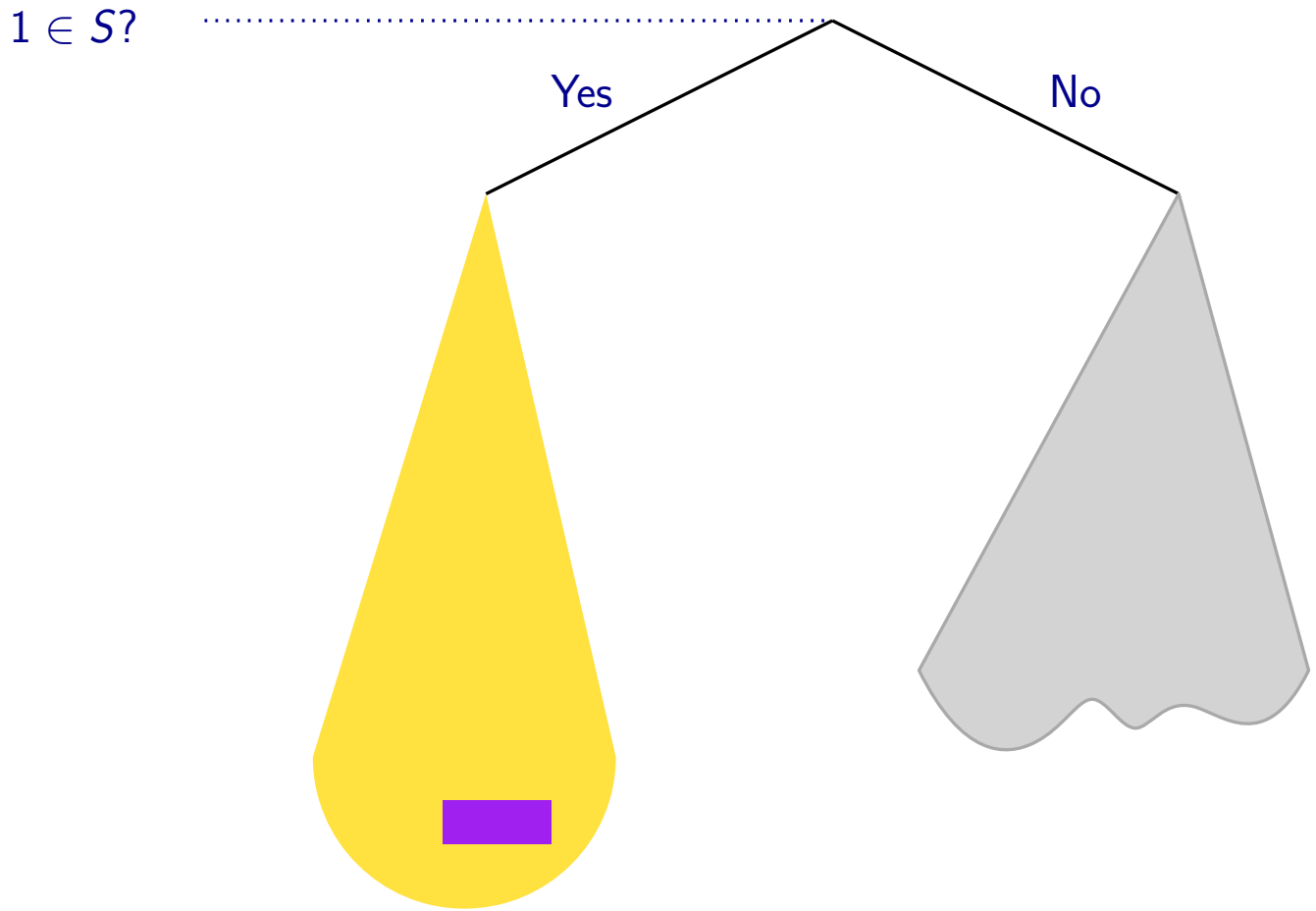
Flashlight search or *Binary partition*



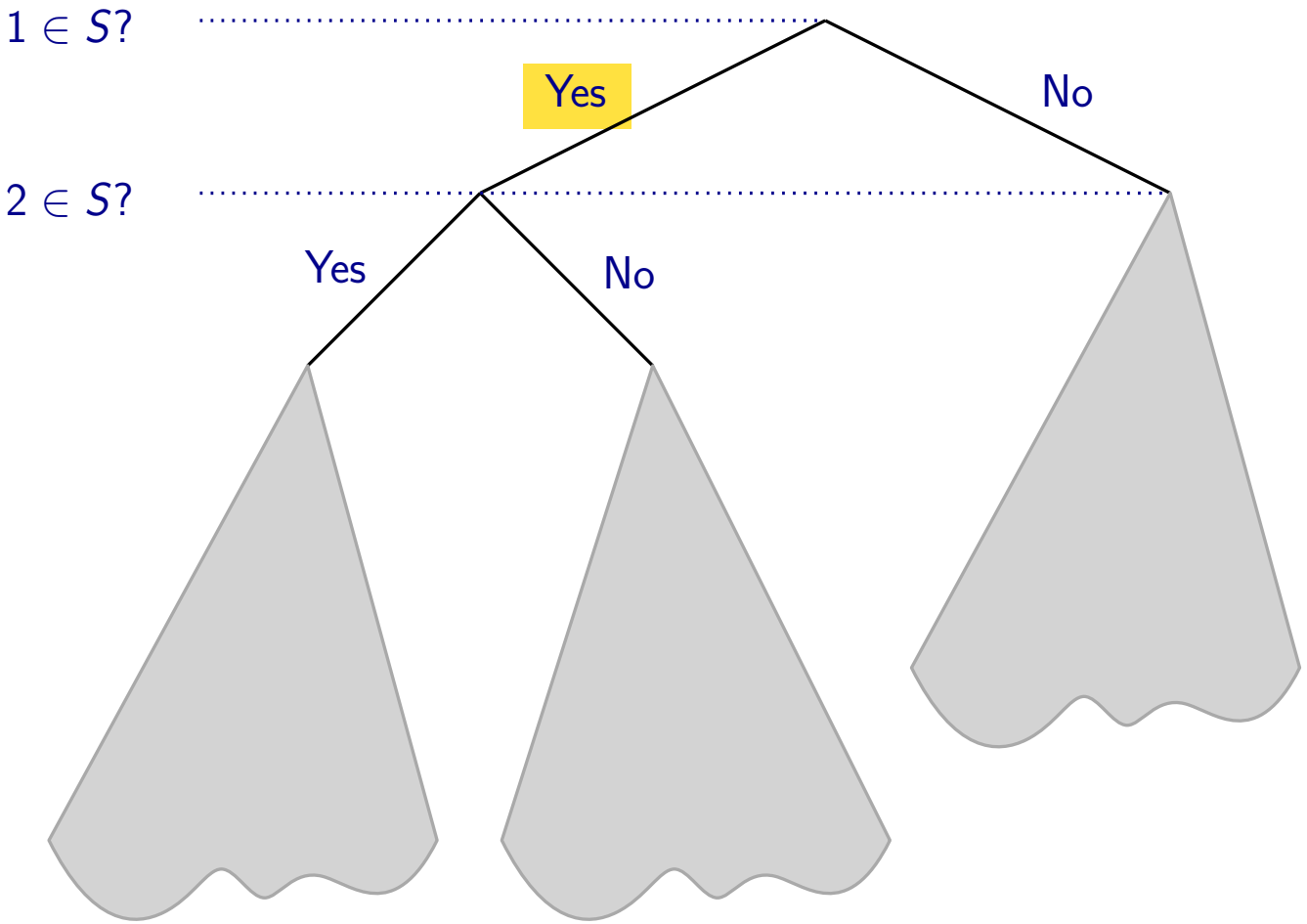
Flashlight search or *Binary partition*



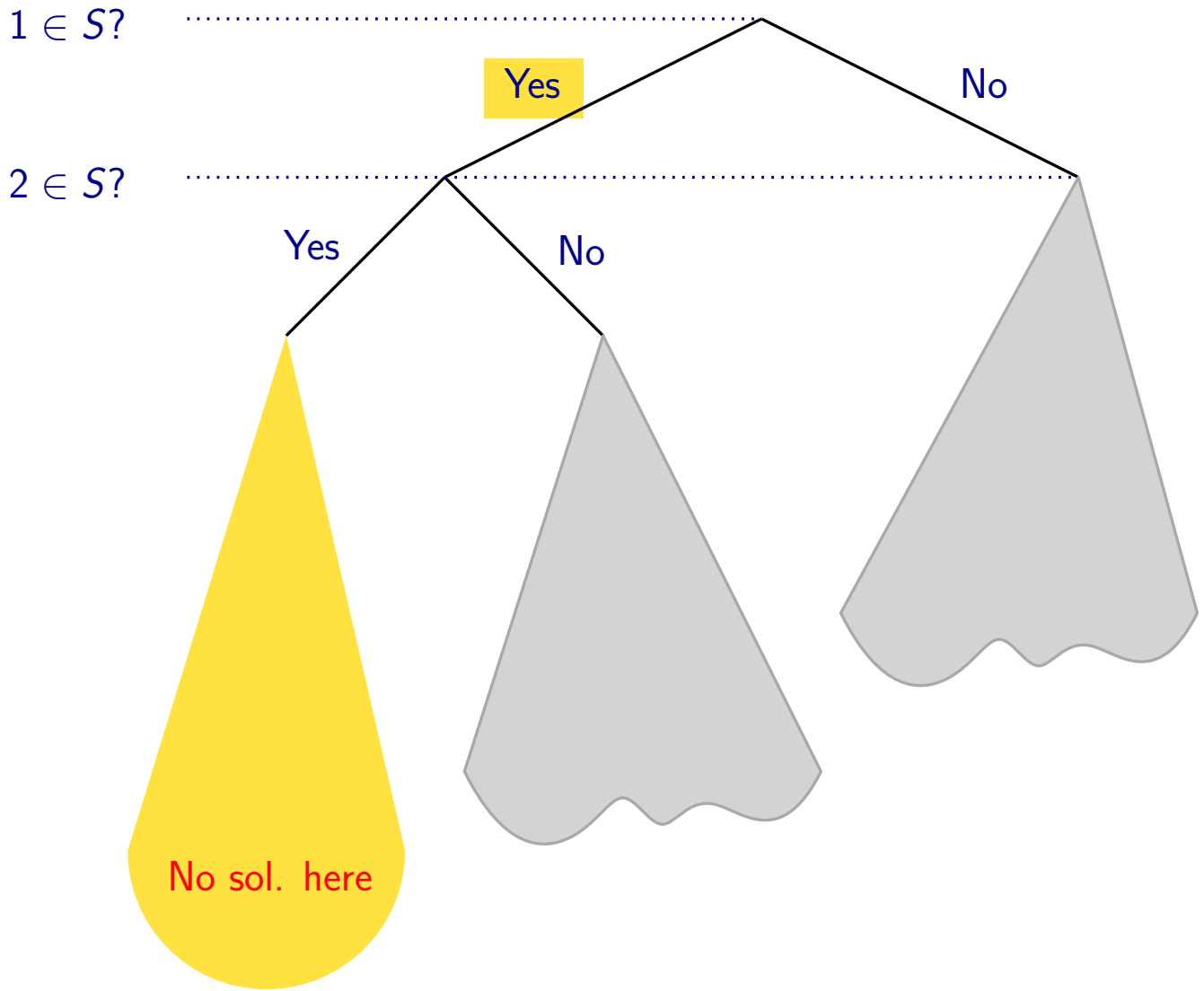
Flashlight search or *Binary partition*



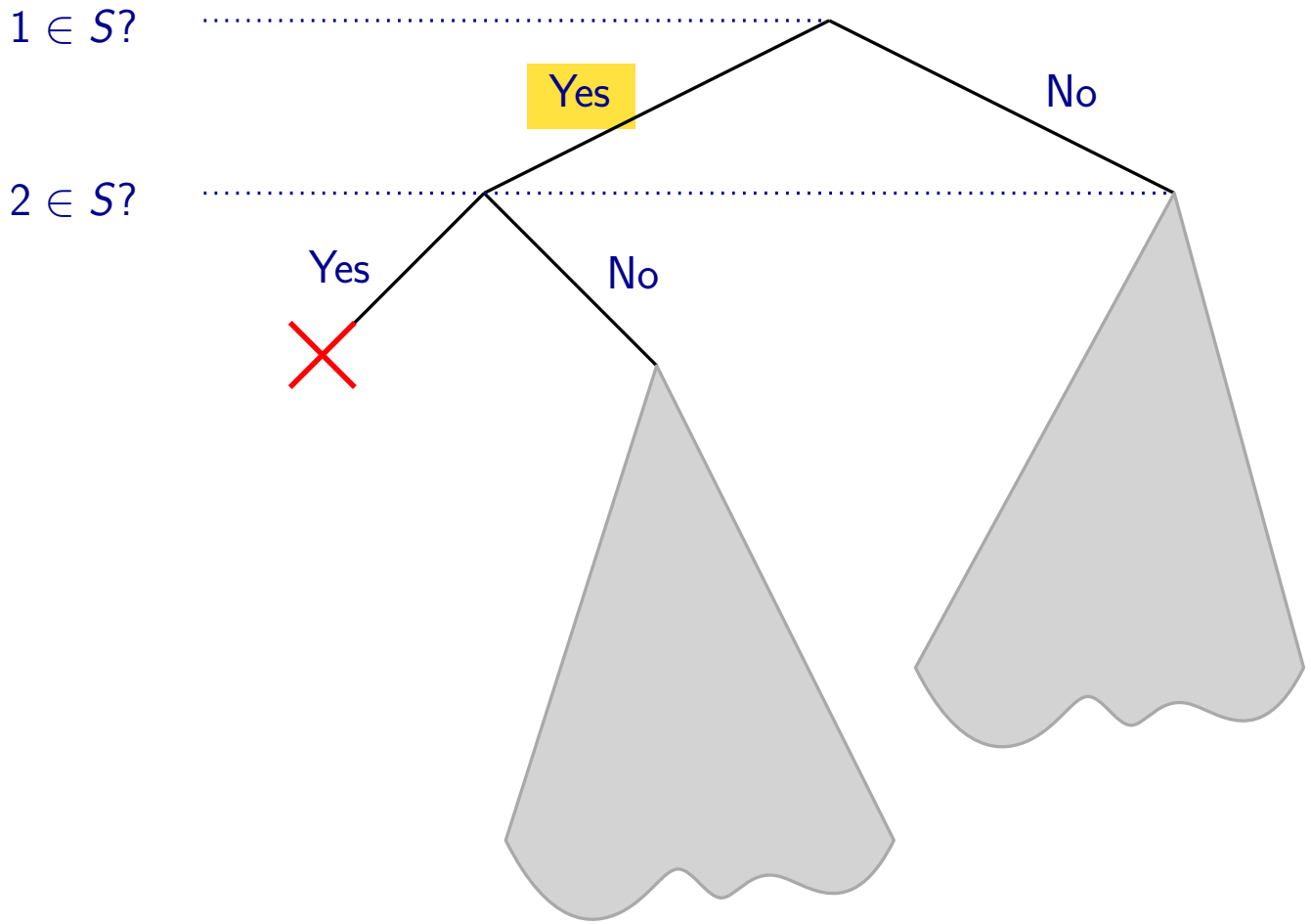
Flashlight search or *Binary partition*



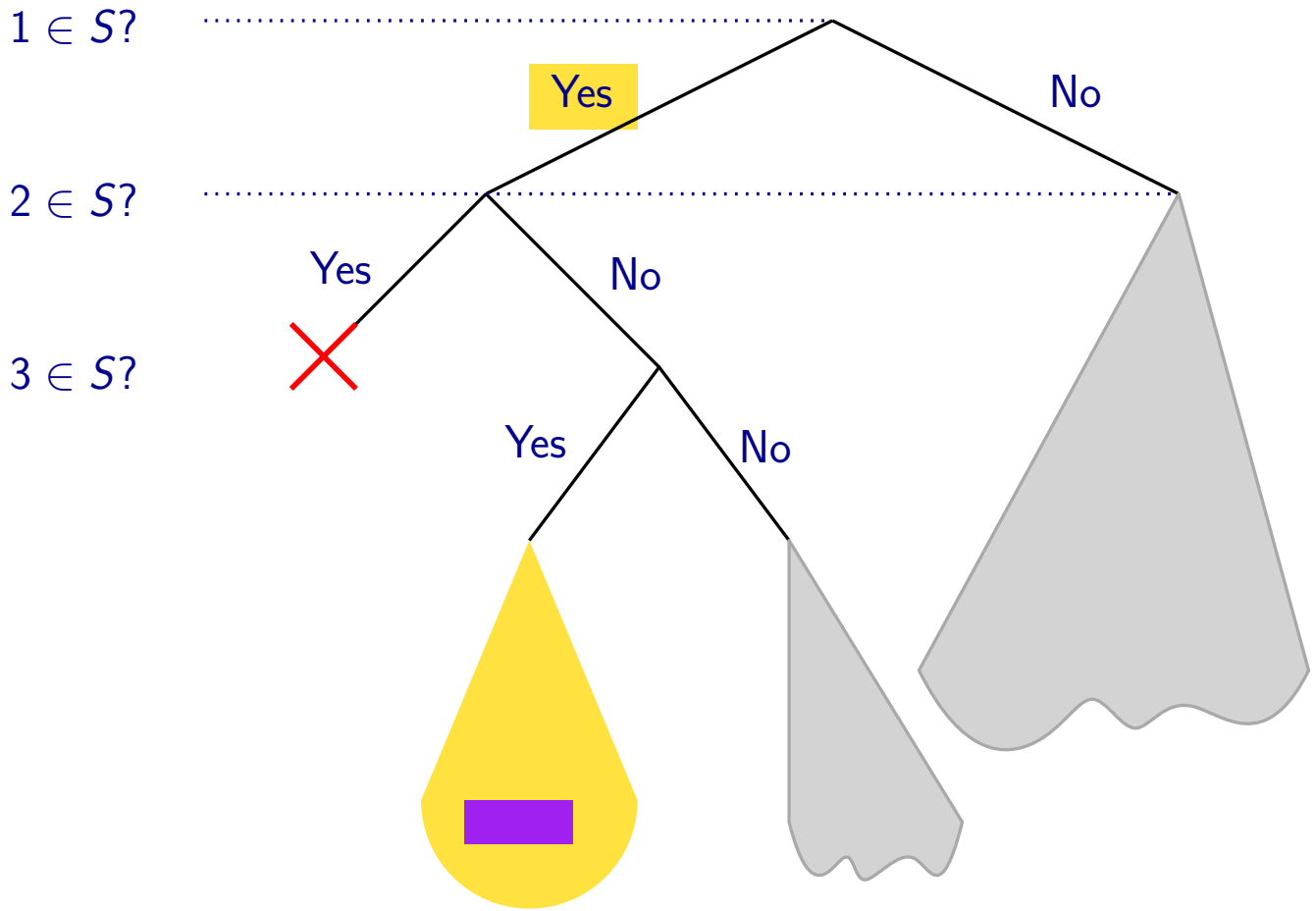
Flashlight search or *Binary partition*



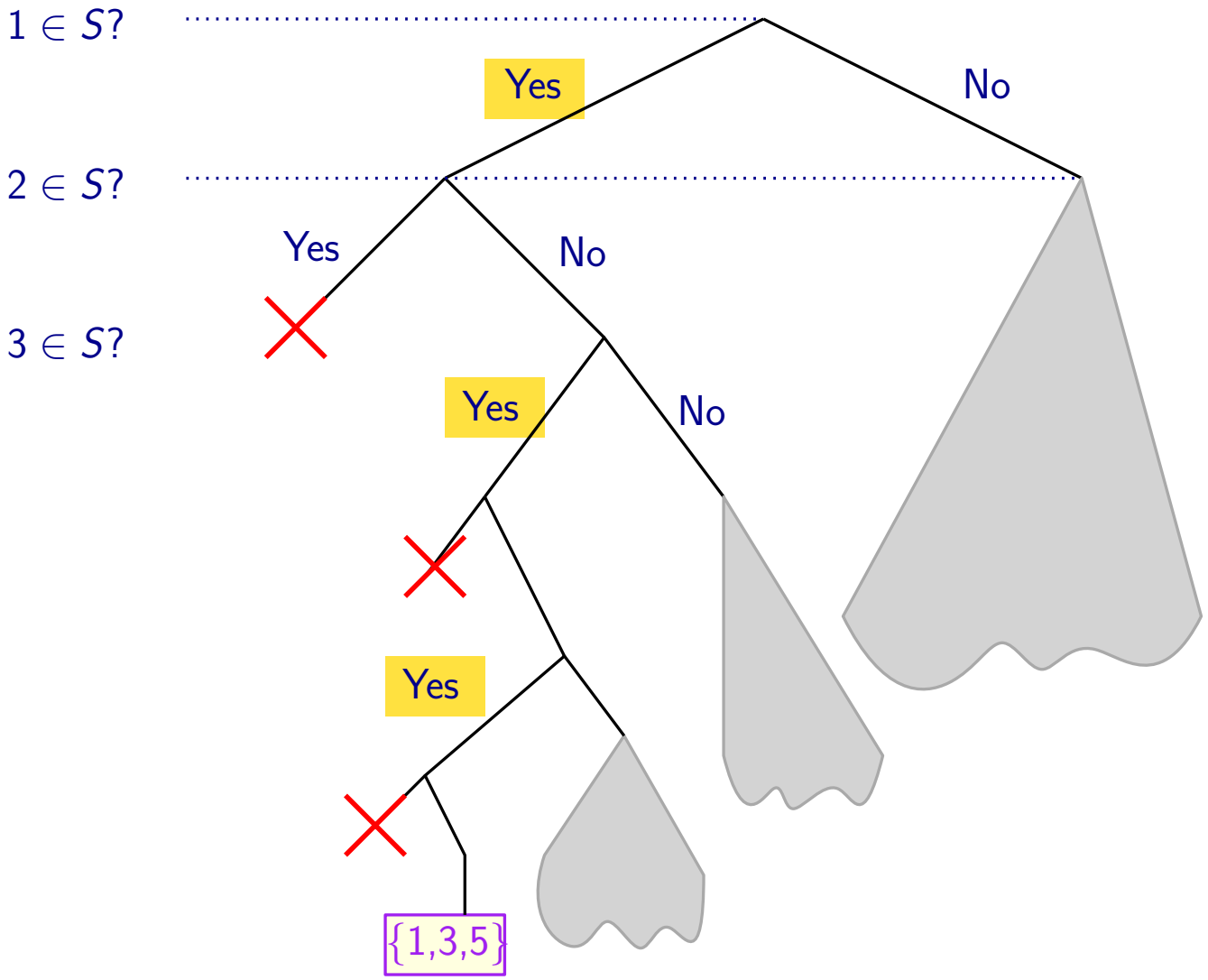
Flashlight search or *Binary partition*



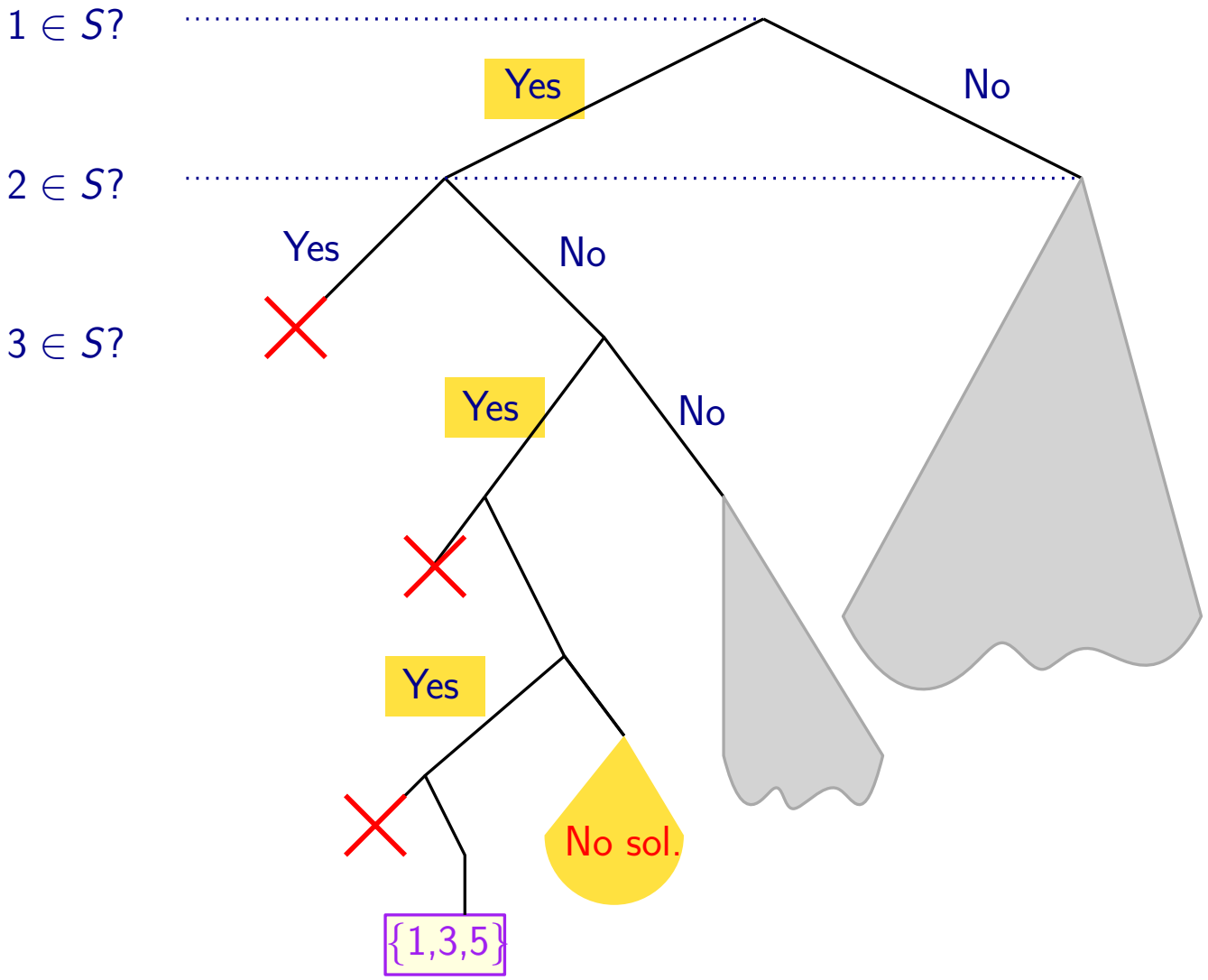
Flashlight search or *Binary partition*



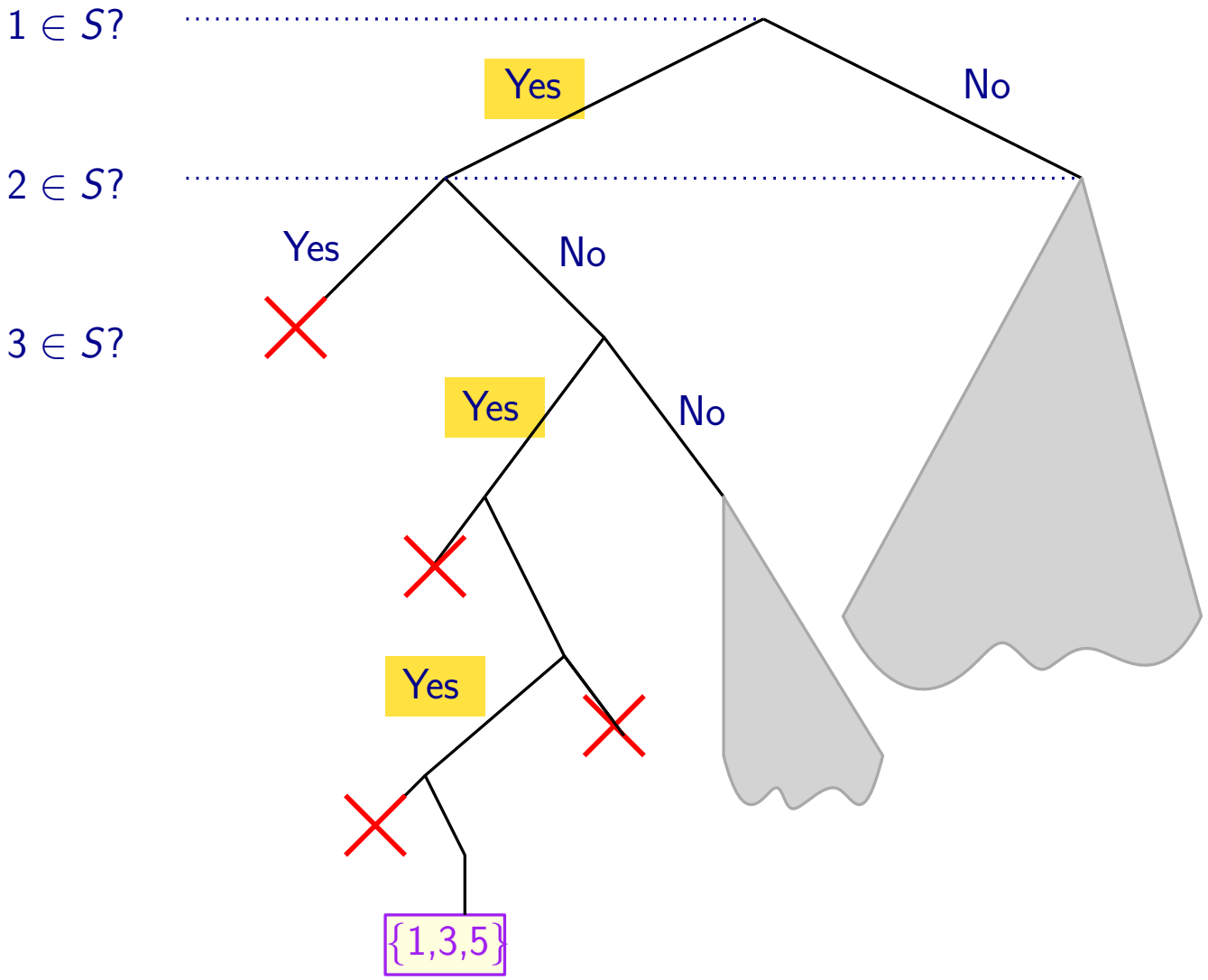
Flashlight search or *Binary partition*



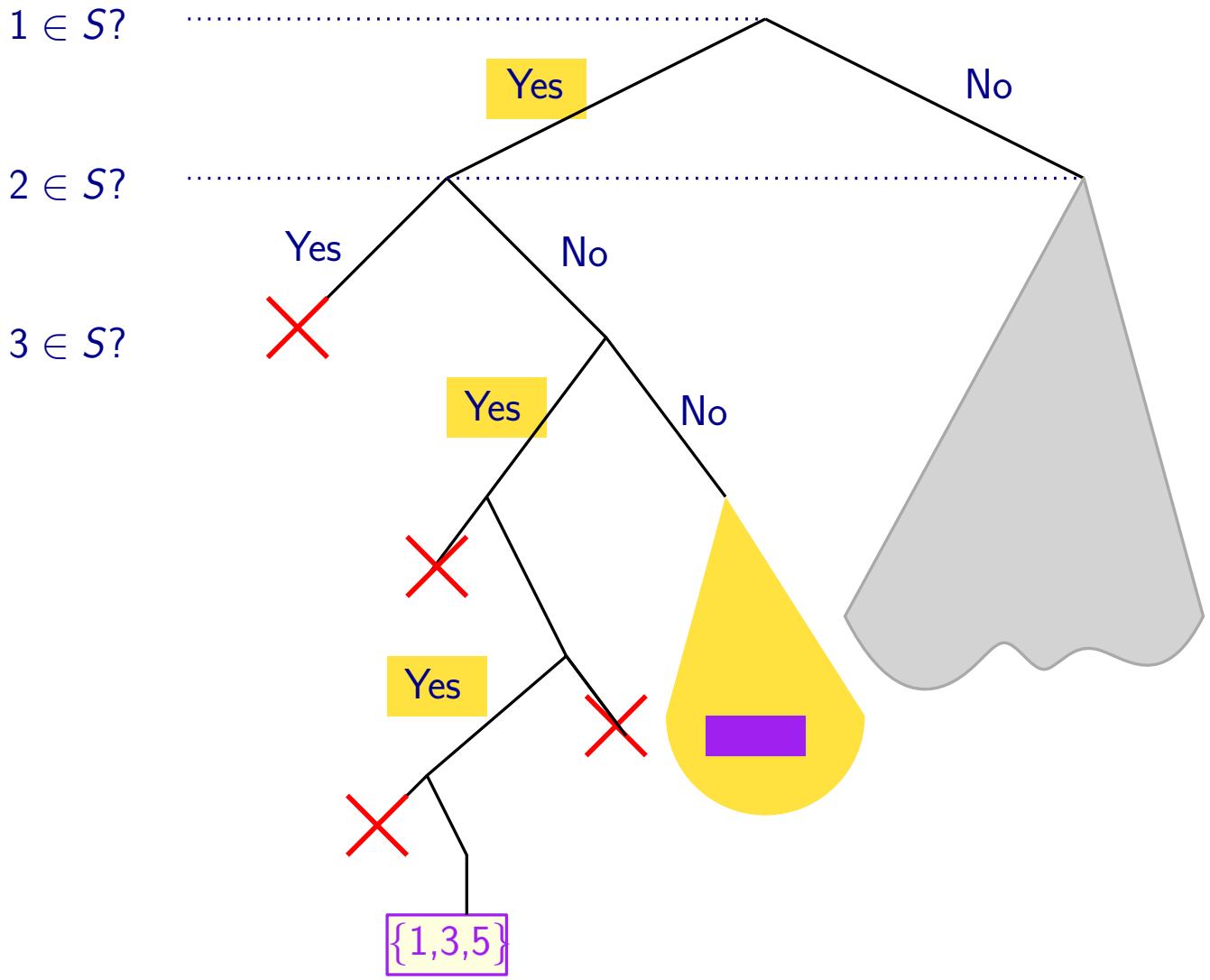
Flashlight search or *Binary partition*



Flashlight search or *Binary partition*



Flashlight search or *Binary partition*

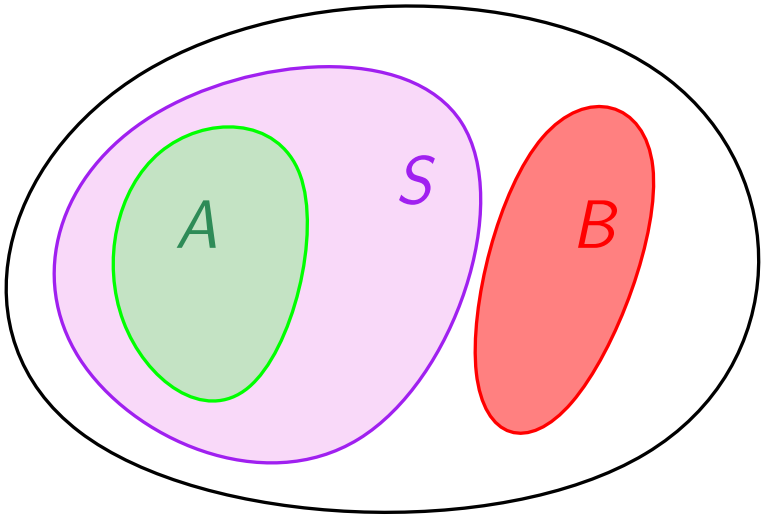


Flashlight search or *Binary partition*

For *Flashlight search* to run in poly delay (and space) :

Answer in poly time to the *extension problem* of $\mathcal{P}$:

Let A and B be two disjoint sets
is there a solution S of $\mathcal{P}$ such that $A \subseteq S$ and $S \cap B = \emptyset$?



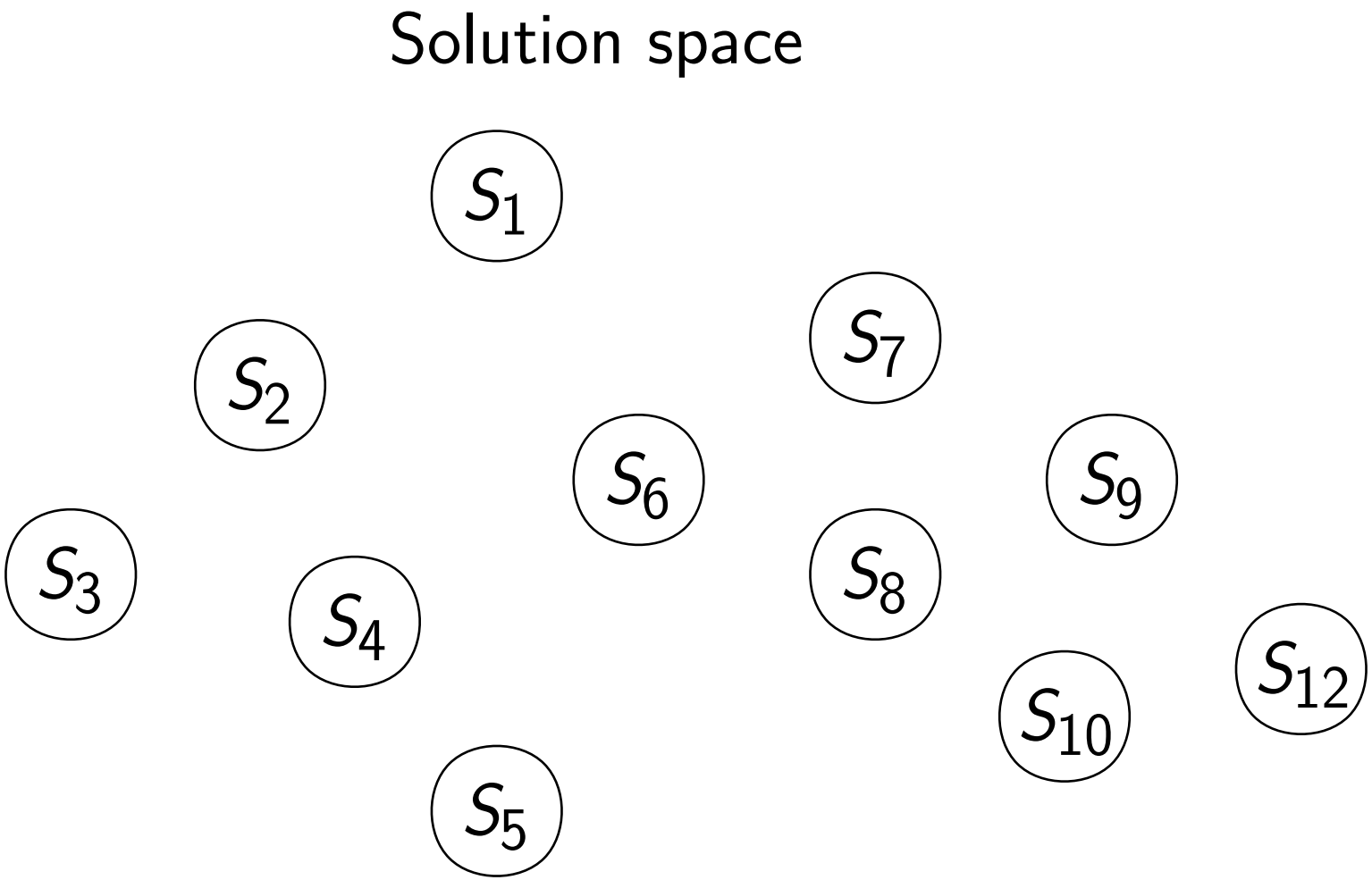
All of A must be in solution
and all of B is forbidden in the solution.

Three classical methods :

Goal : get polynomial delay
+ poly space for 1 et 2 (and sometimes 3)

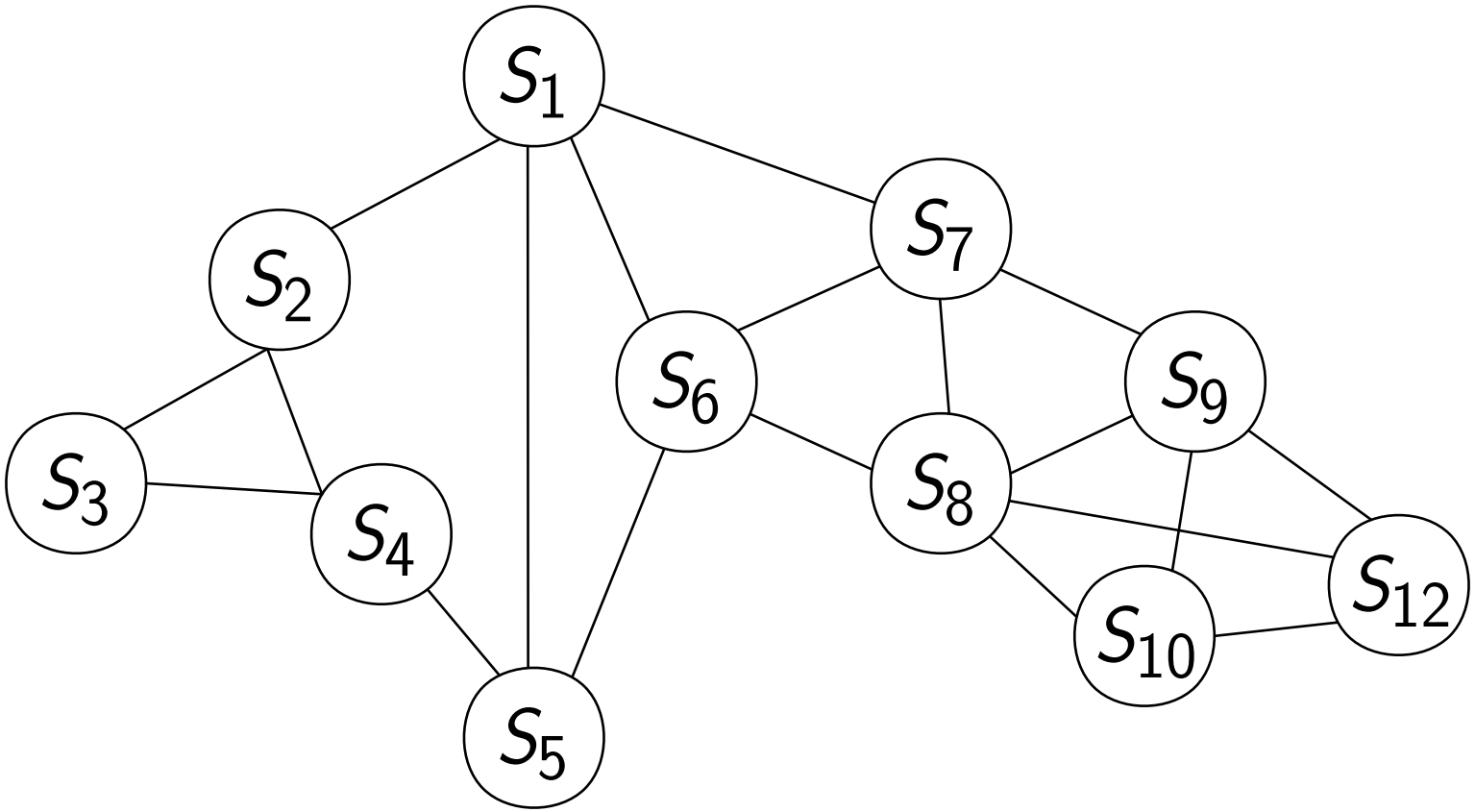
1. *Flashlight search* or *Binary partition*
   [Read, Tarjan '75]
2. ***Reverse search***
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

Reverse search



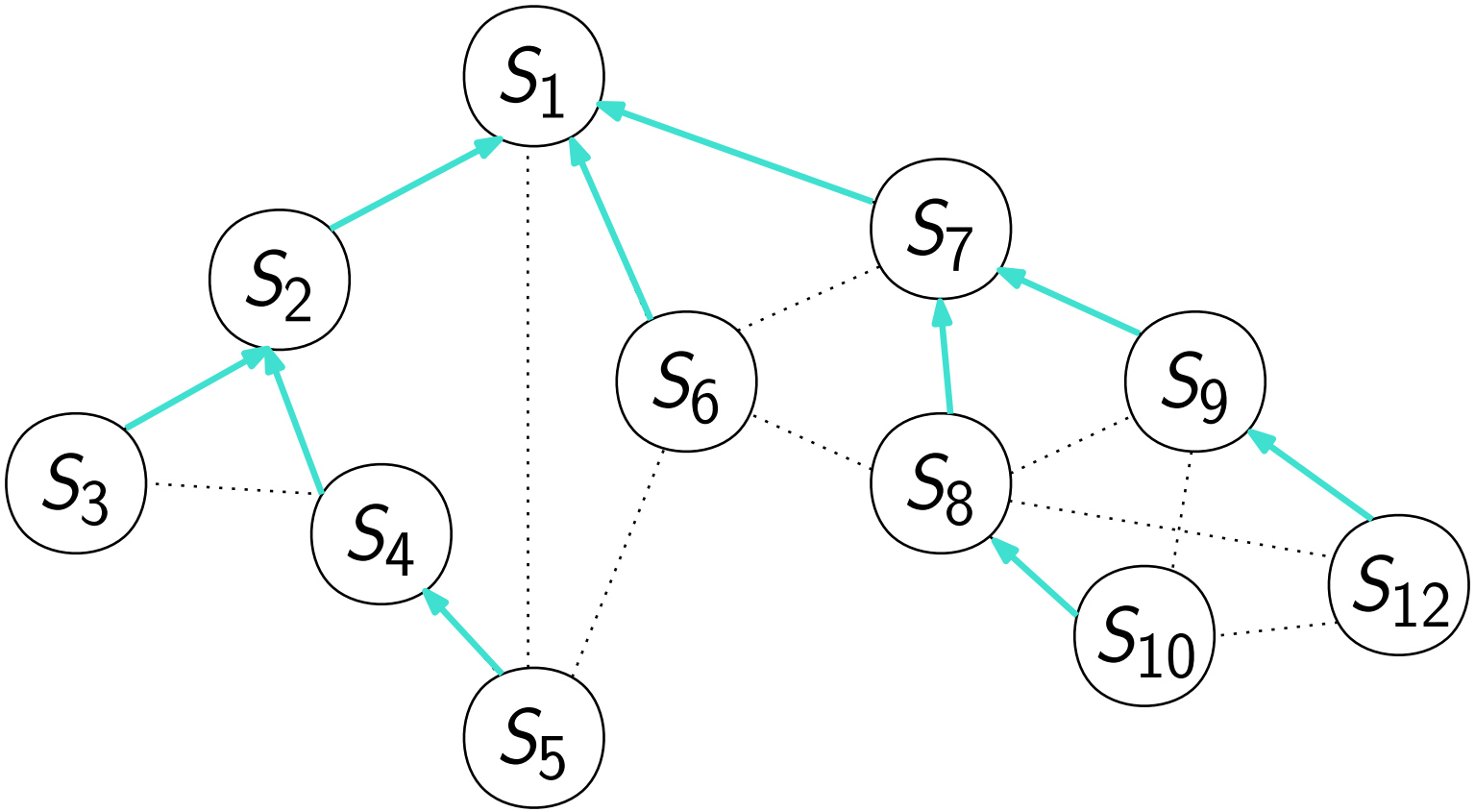
Reverse search

Solution metagraph



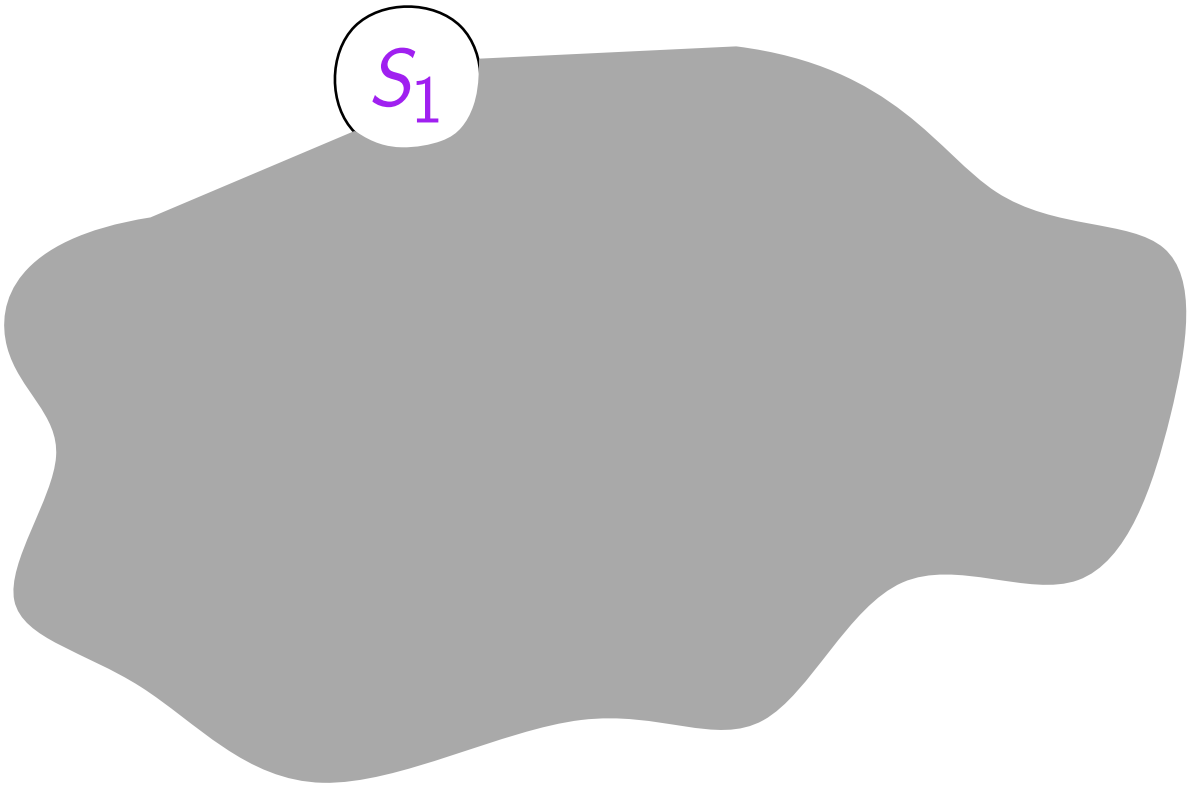
Reverse search

Solution tree

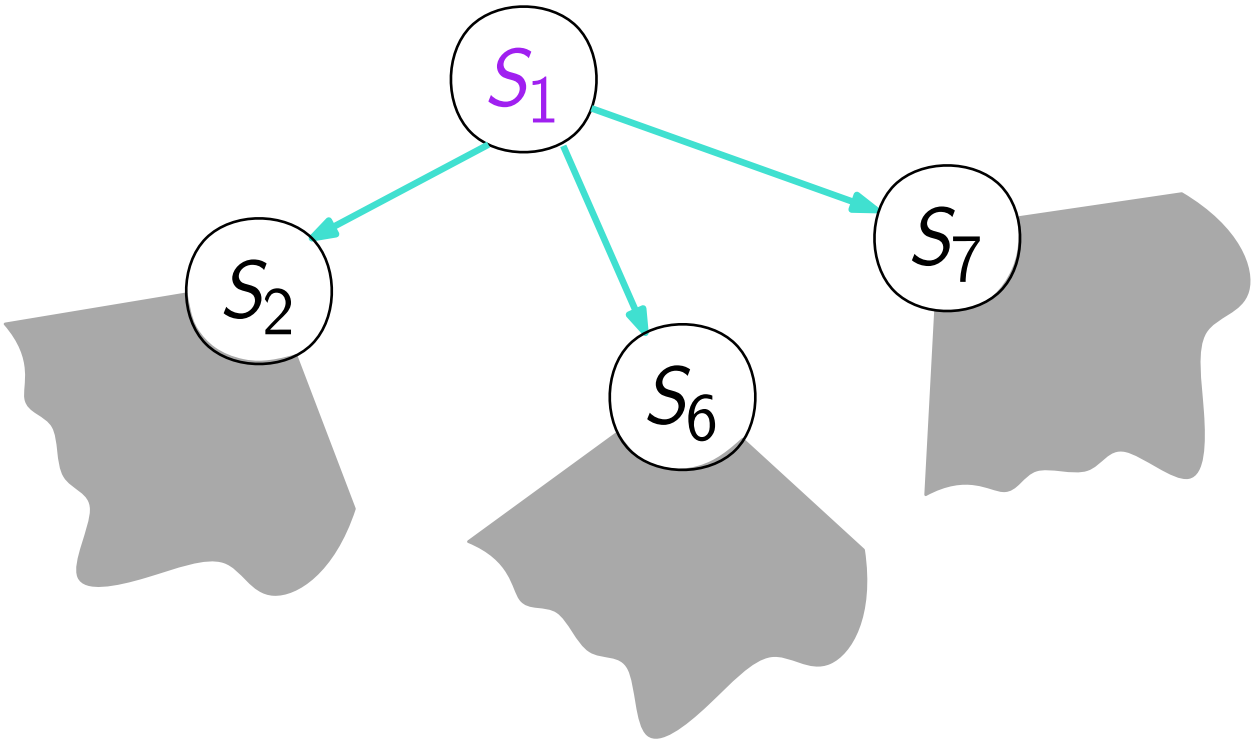


Reverse search

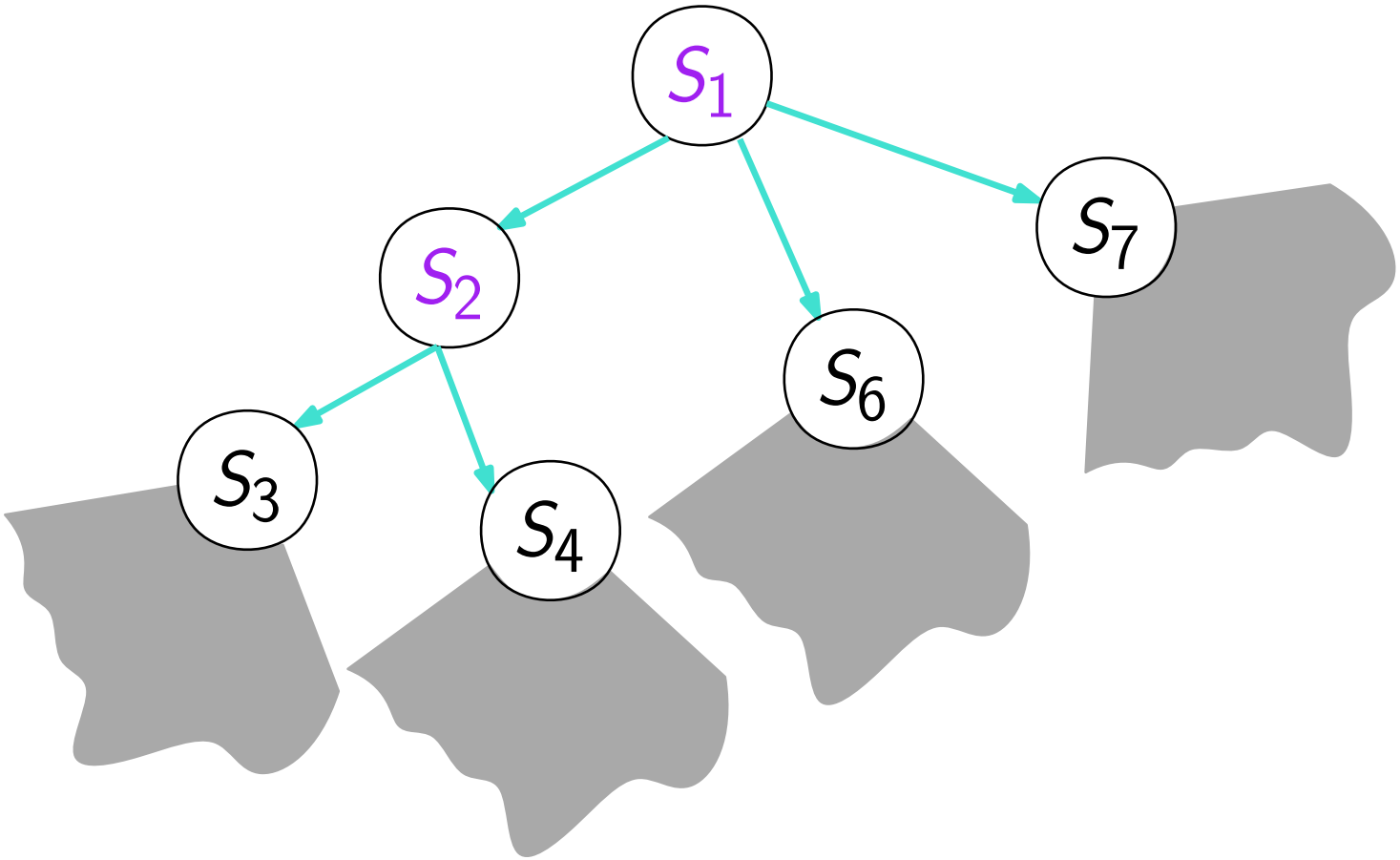
Solution tree



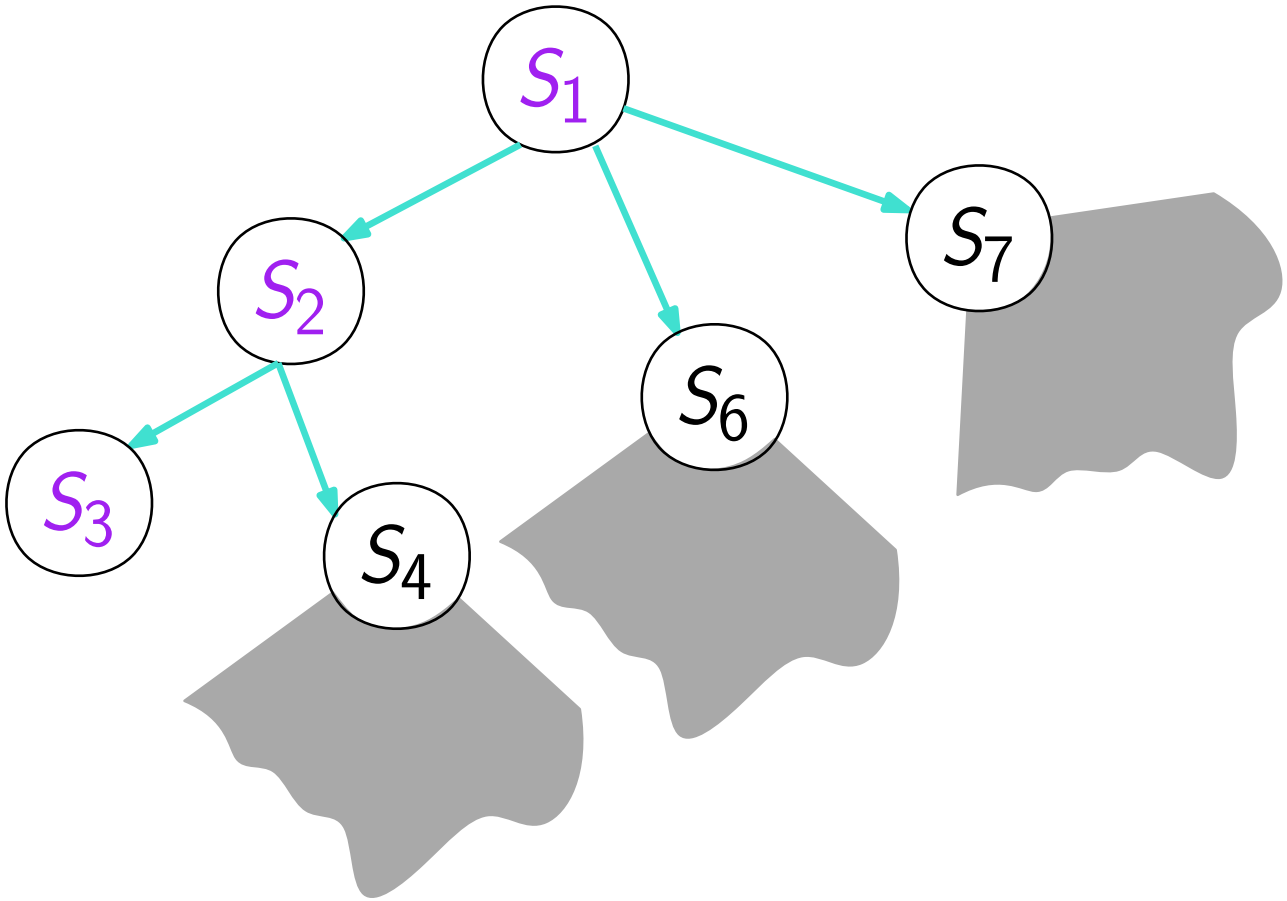
Reverse search



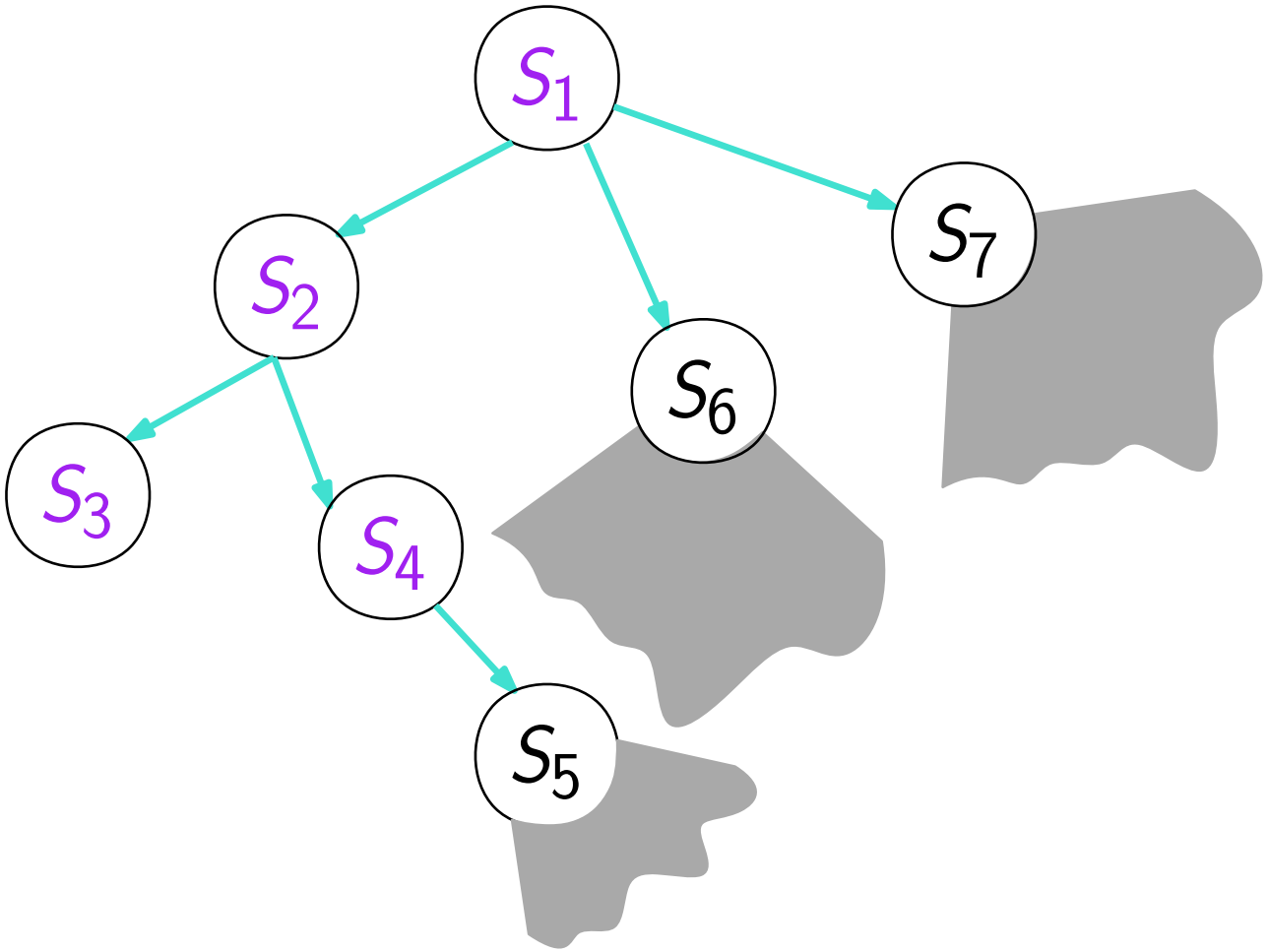
Reverse search



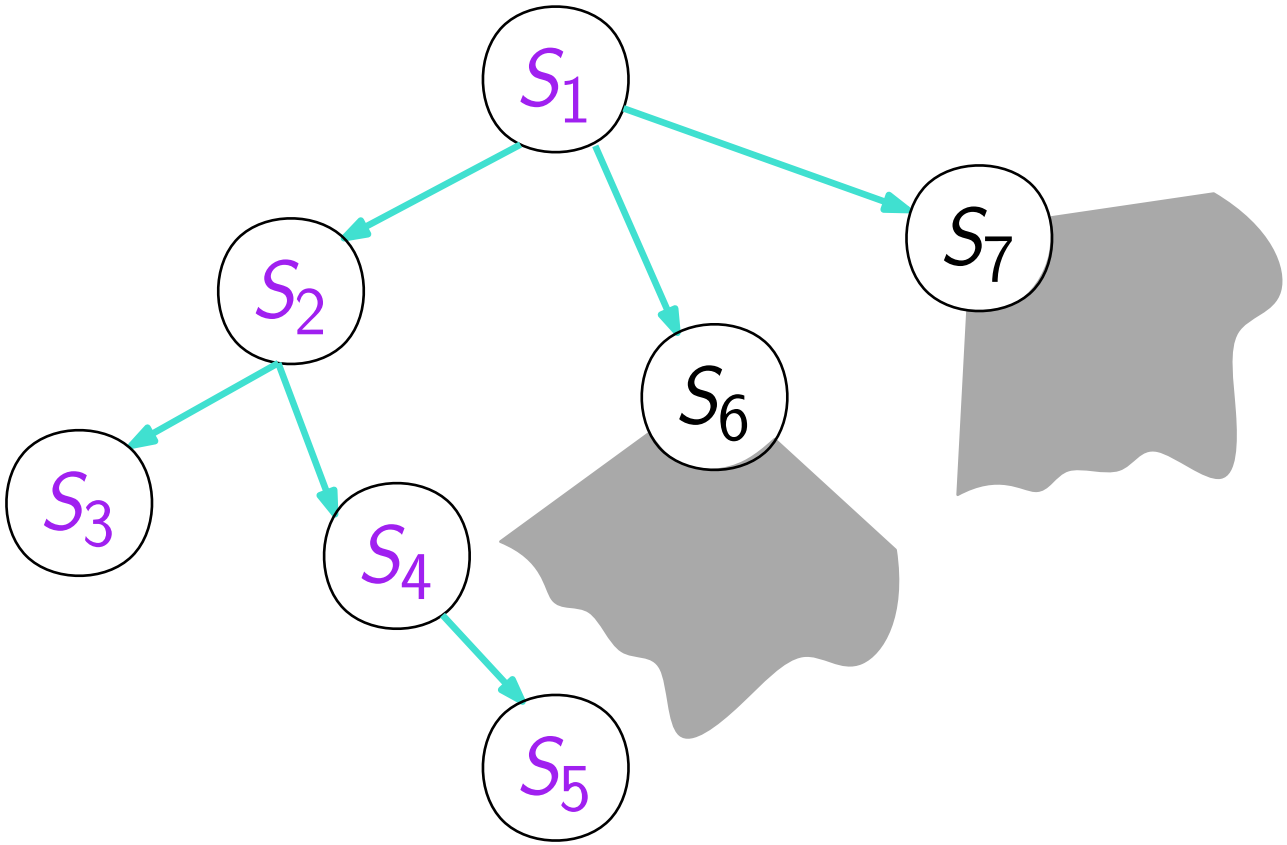
Reverse search



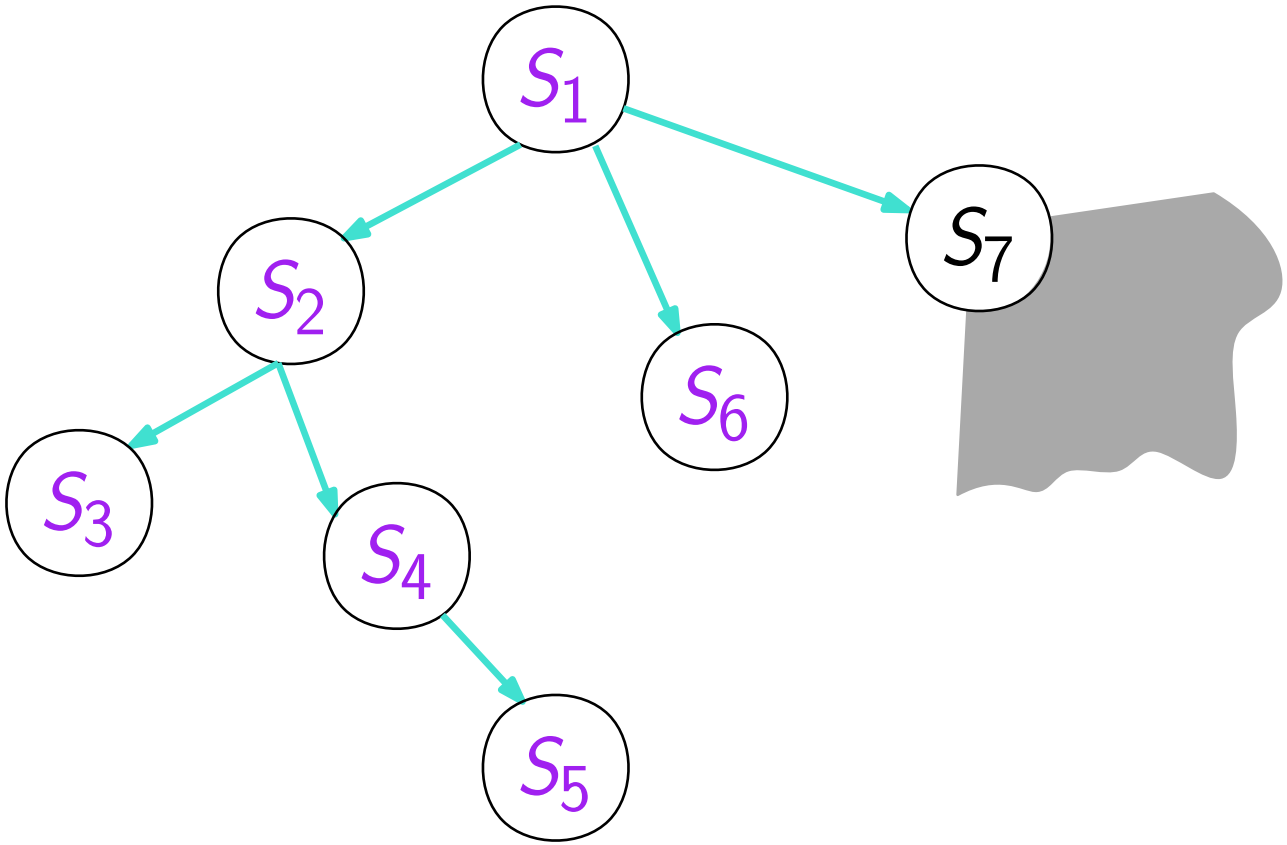
Reverse search



Reverse search



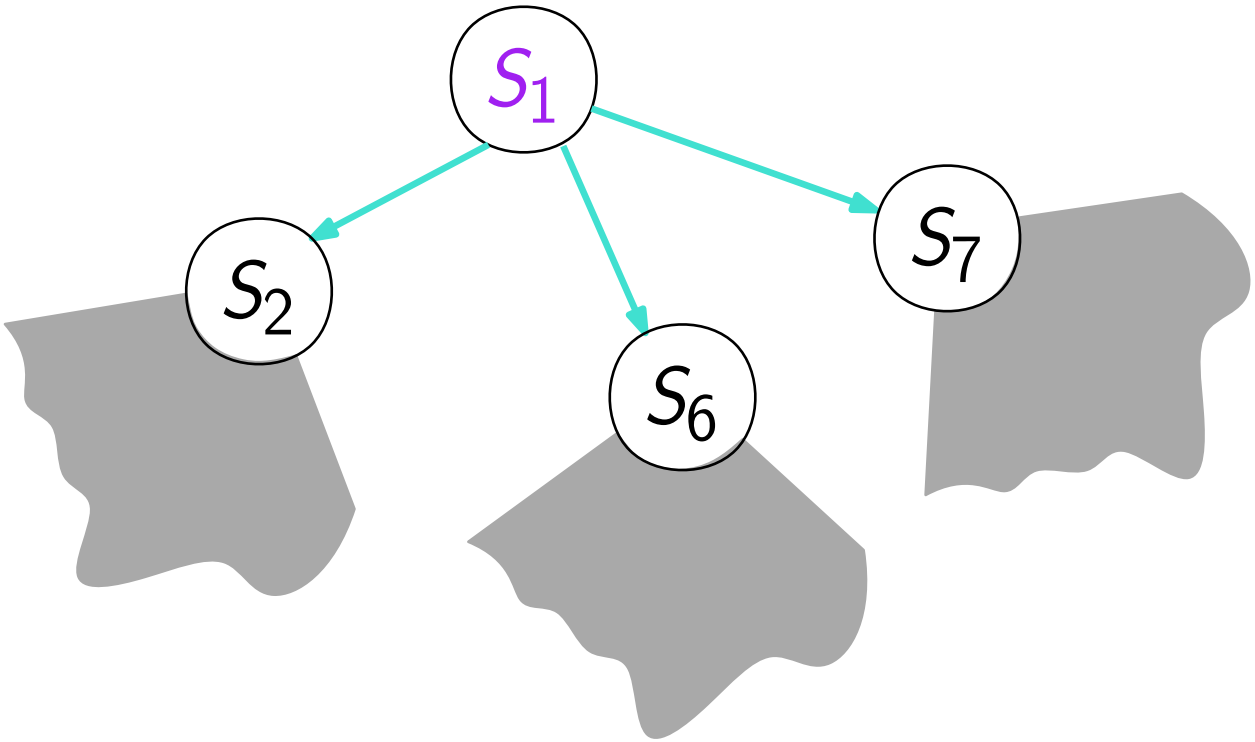
Reverse search



Reverse search

To have *Reverse search* run in poly delay and space :

Generate in poly time and space the *children* of a solution (each solution must have ***a single*** father)



Reverse Search