Search-Order Independent State Caching

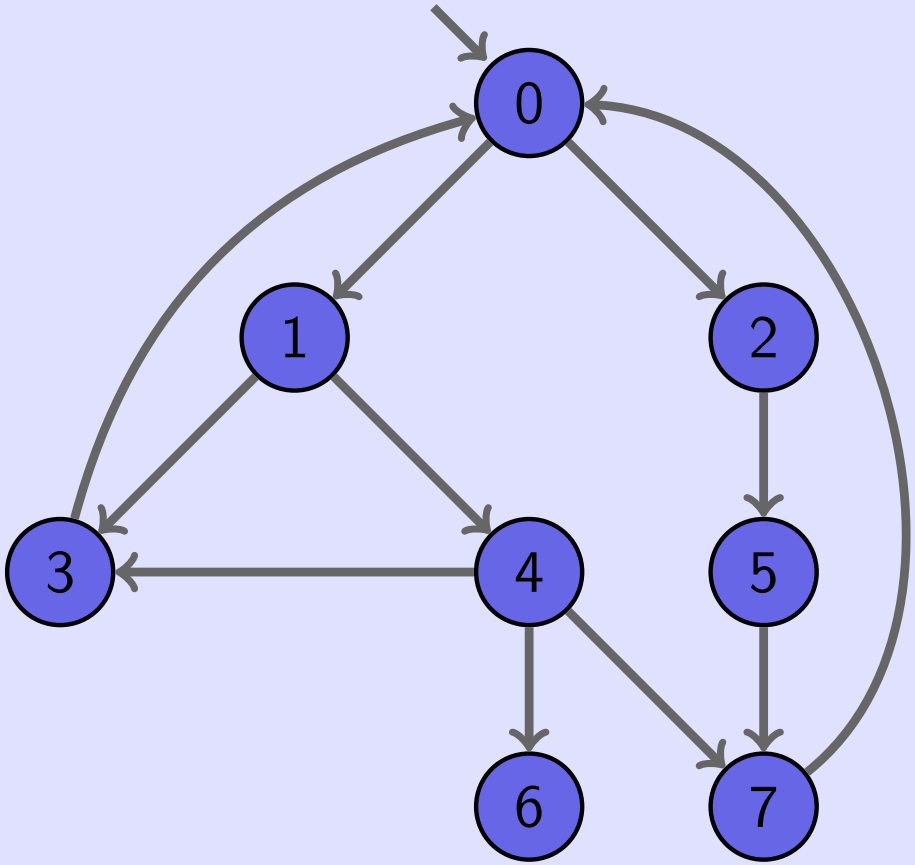
Sami Evangelista – Université Paris 13, France

Lars Michael Kristensen – Bergen University College, Norway

CPN'2009 – October 2009

GSEA — General State Exploring Algorithm

Example of a BFS



open state

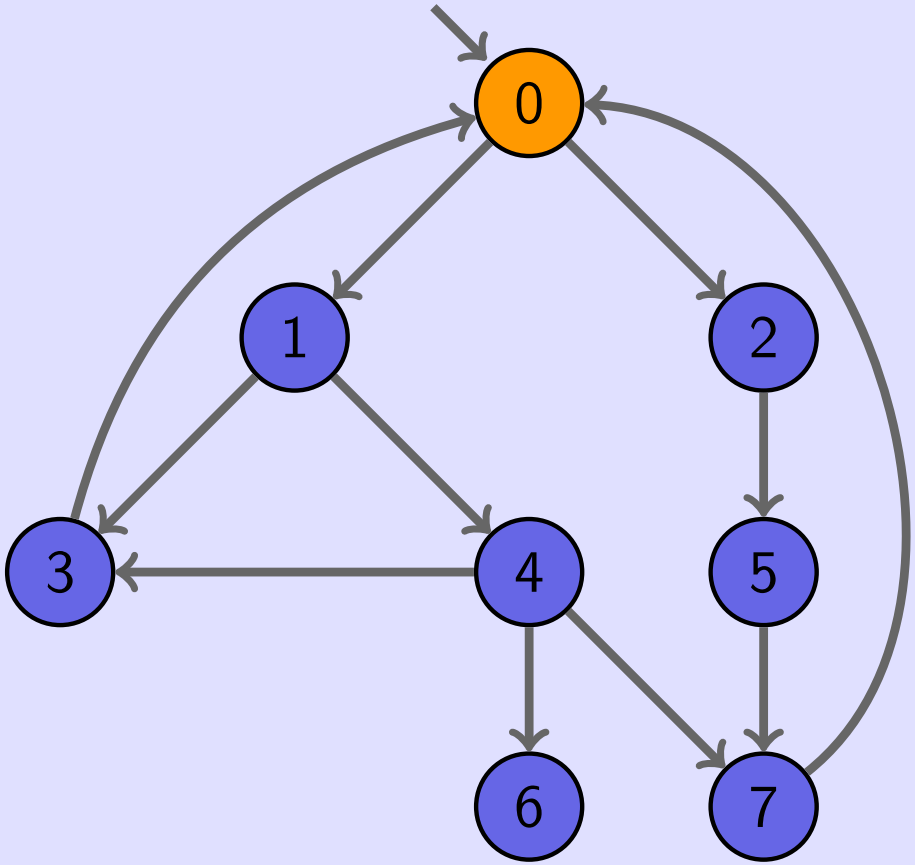
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

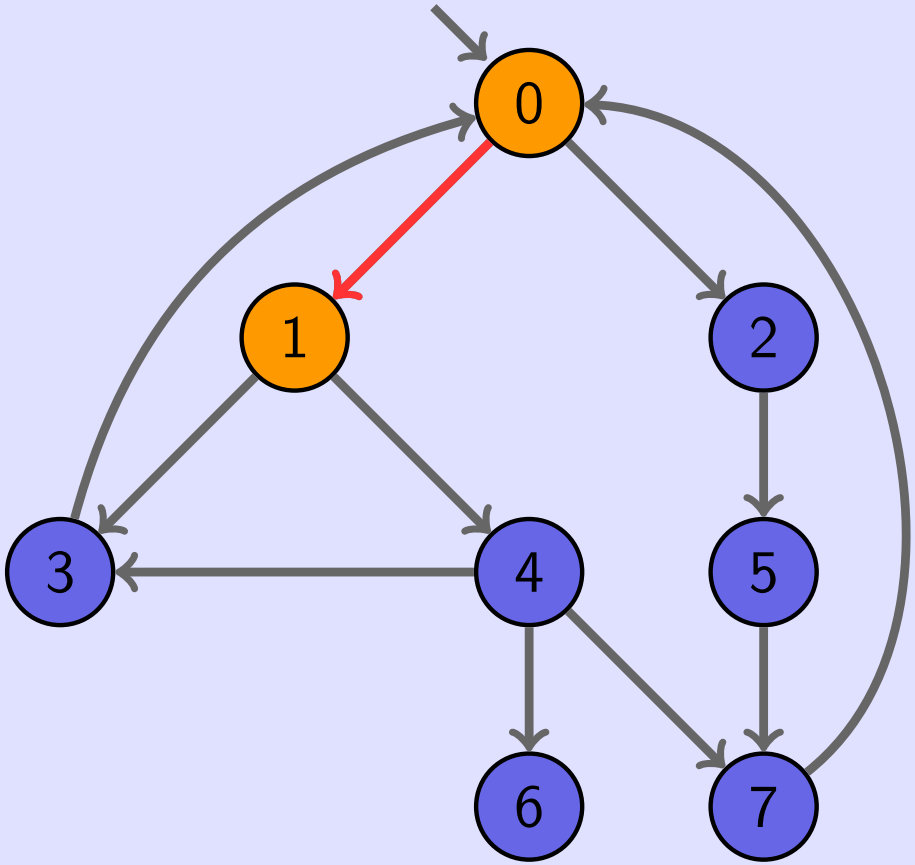
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

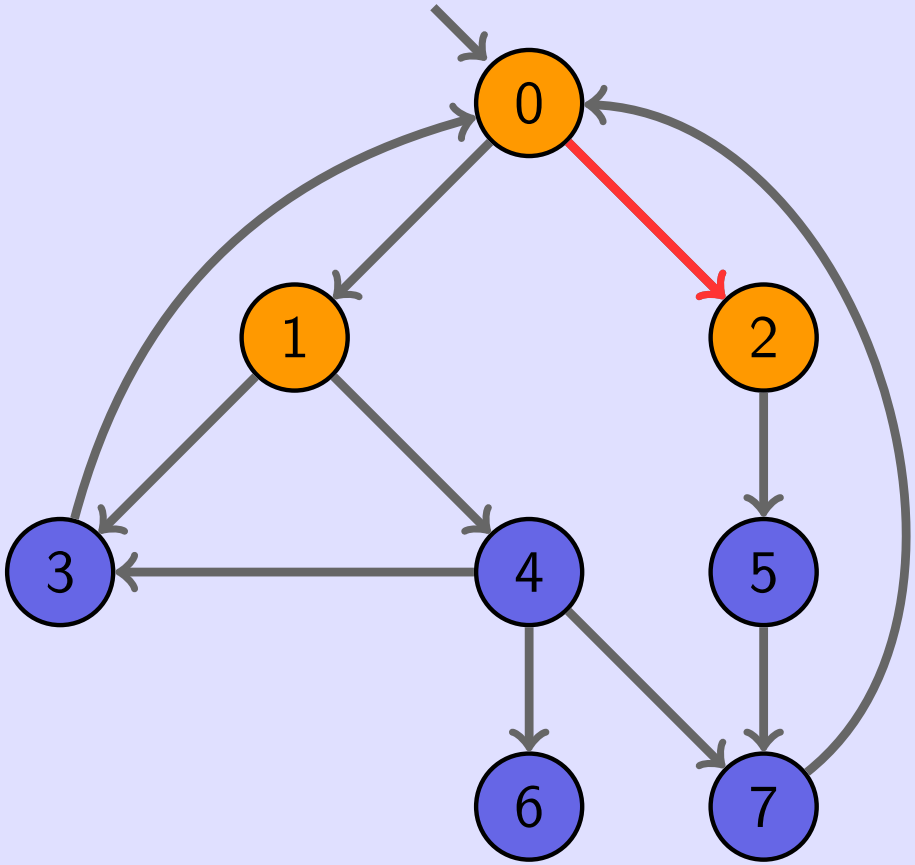
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

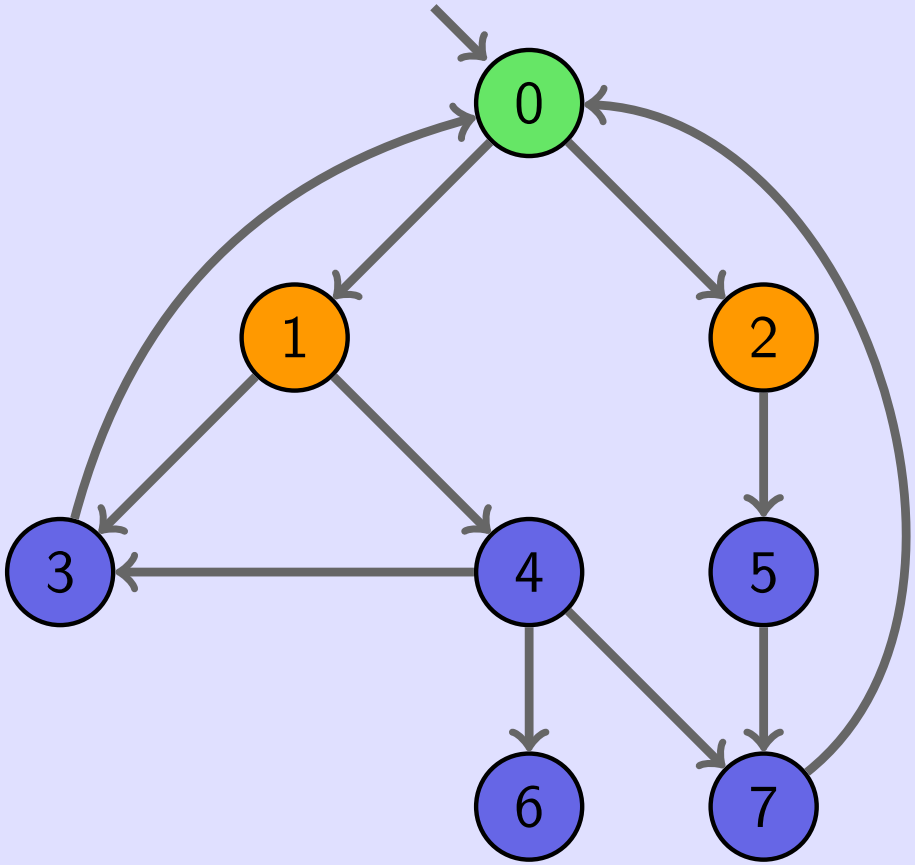
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

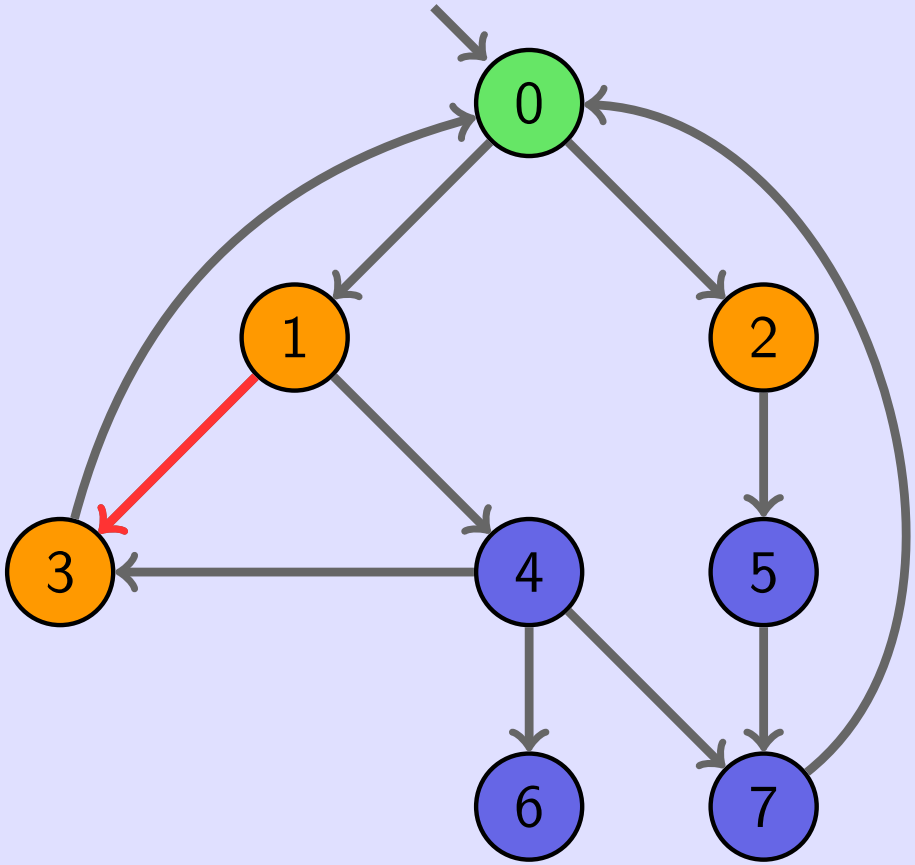
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

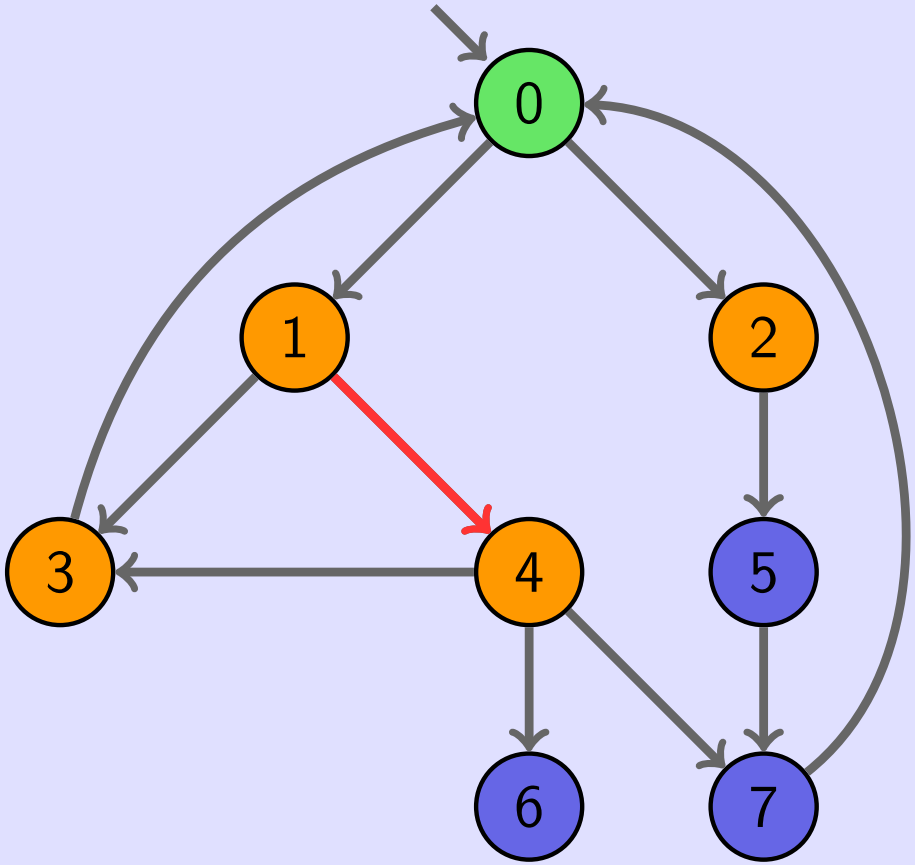
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

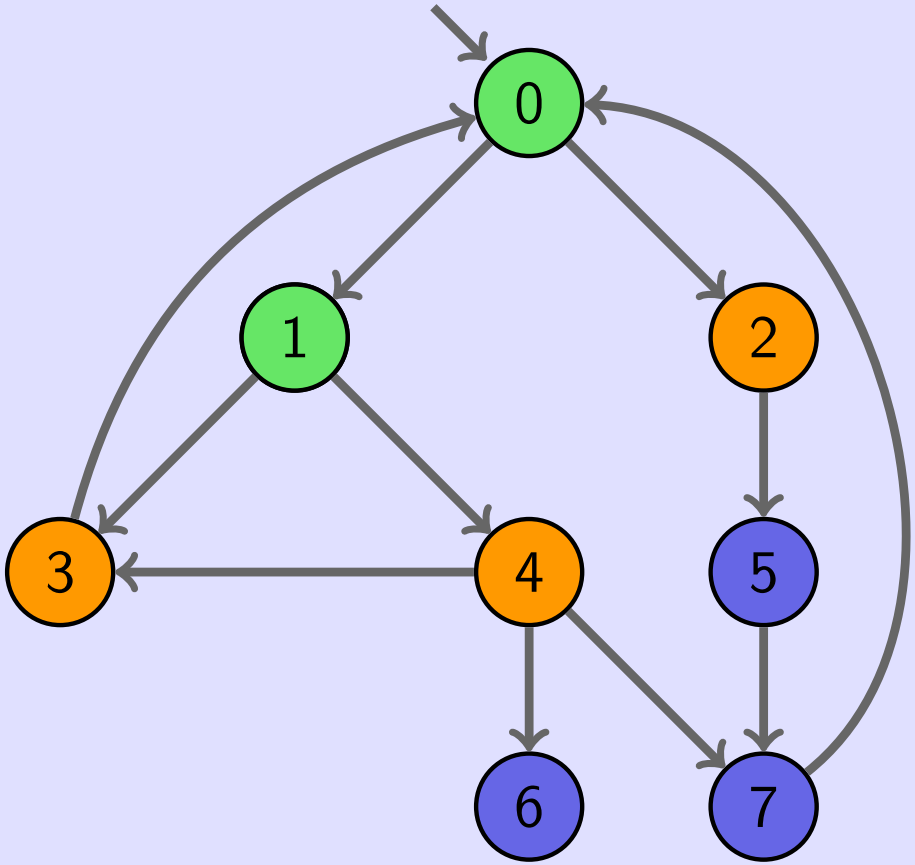
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

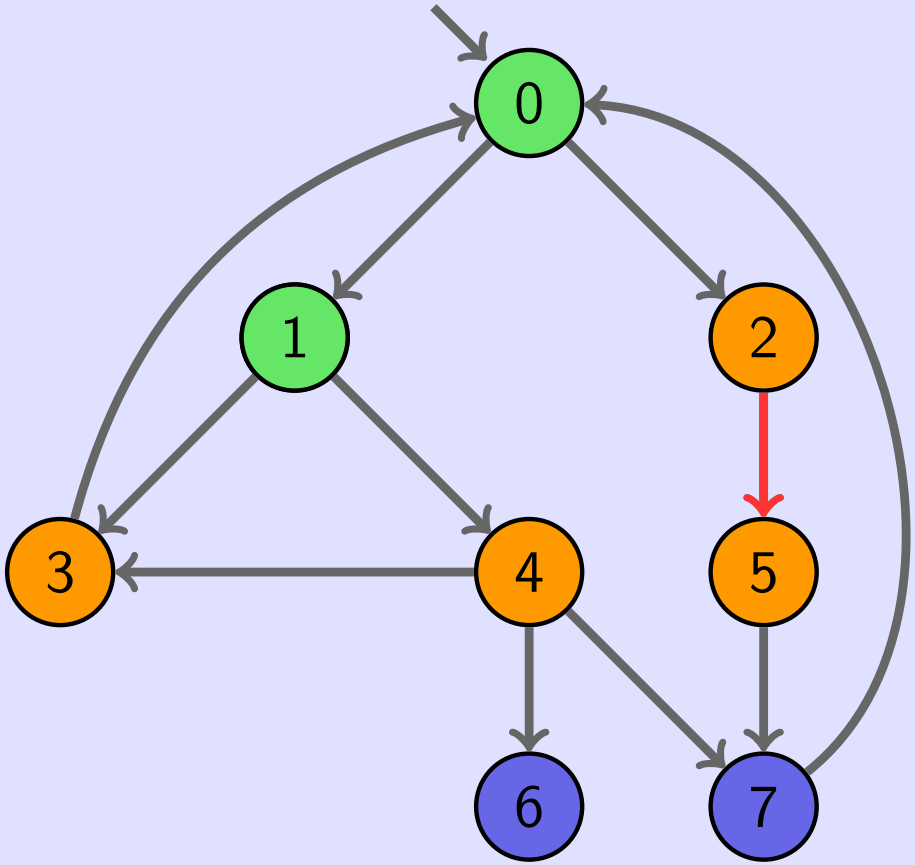
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

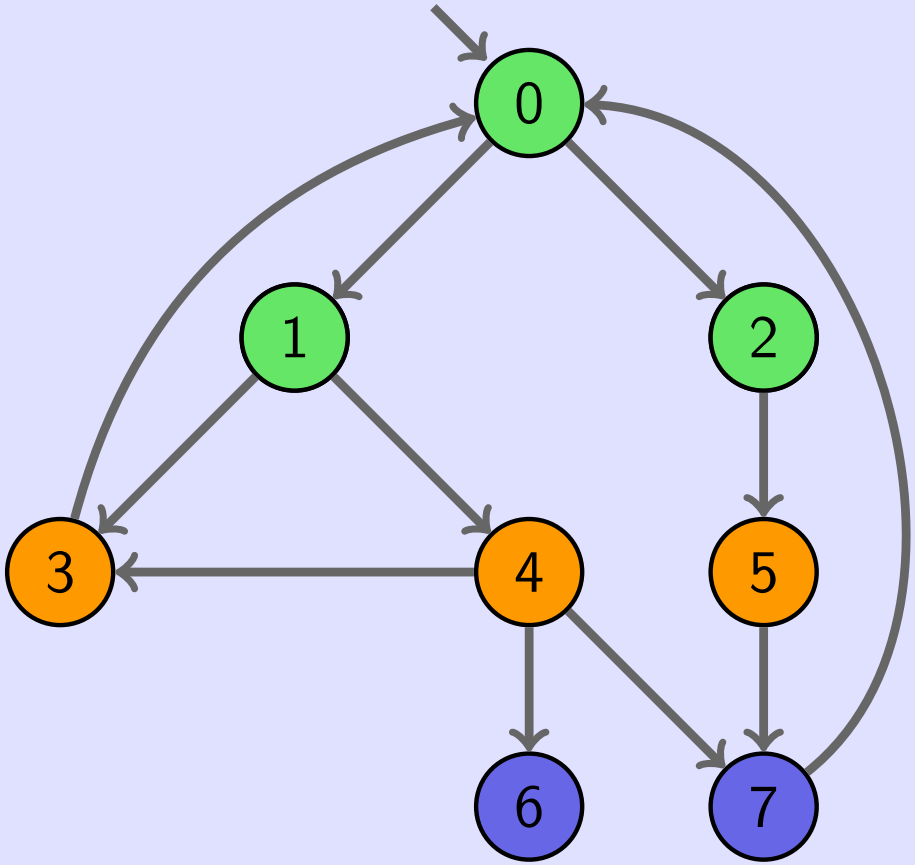
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

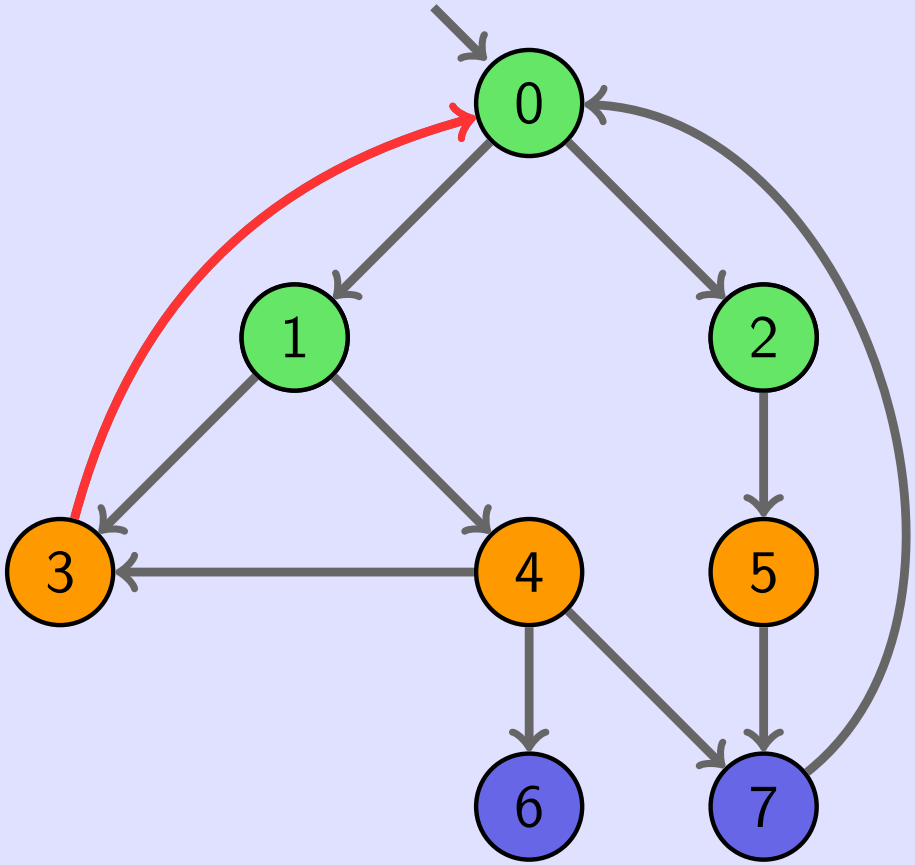
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

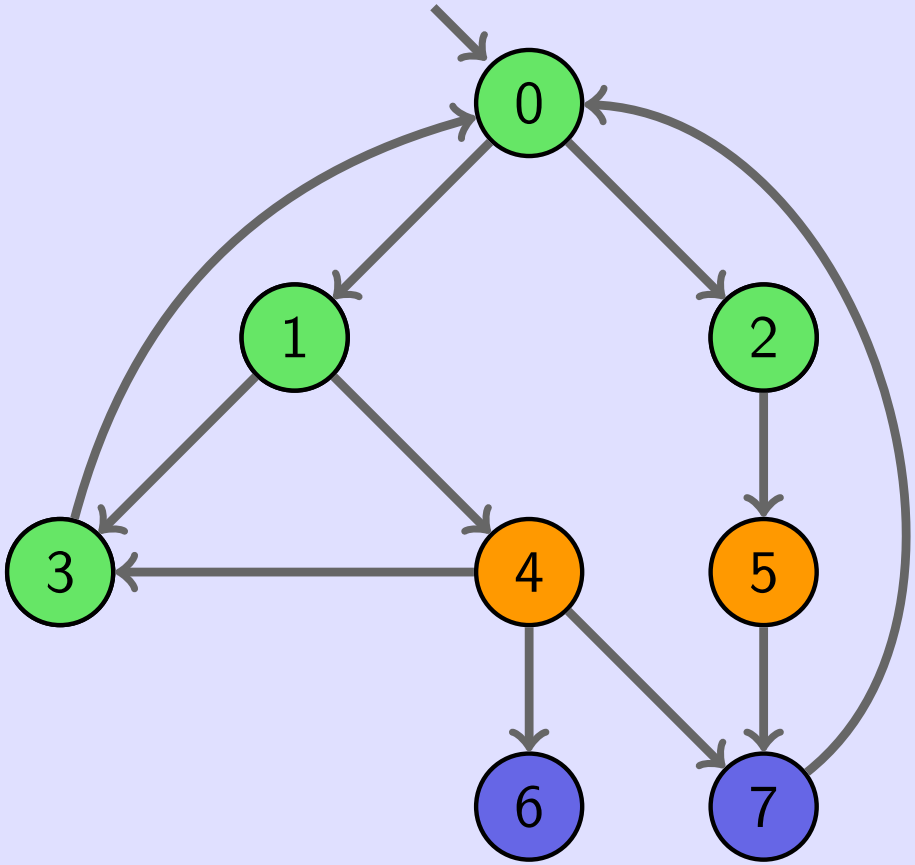
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

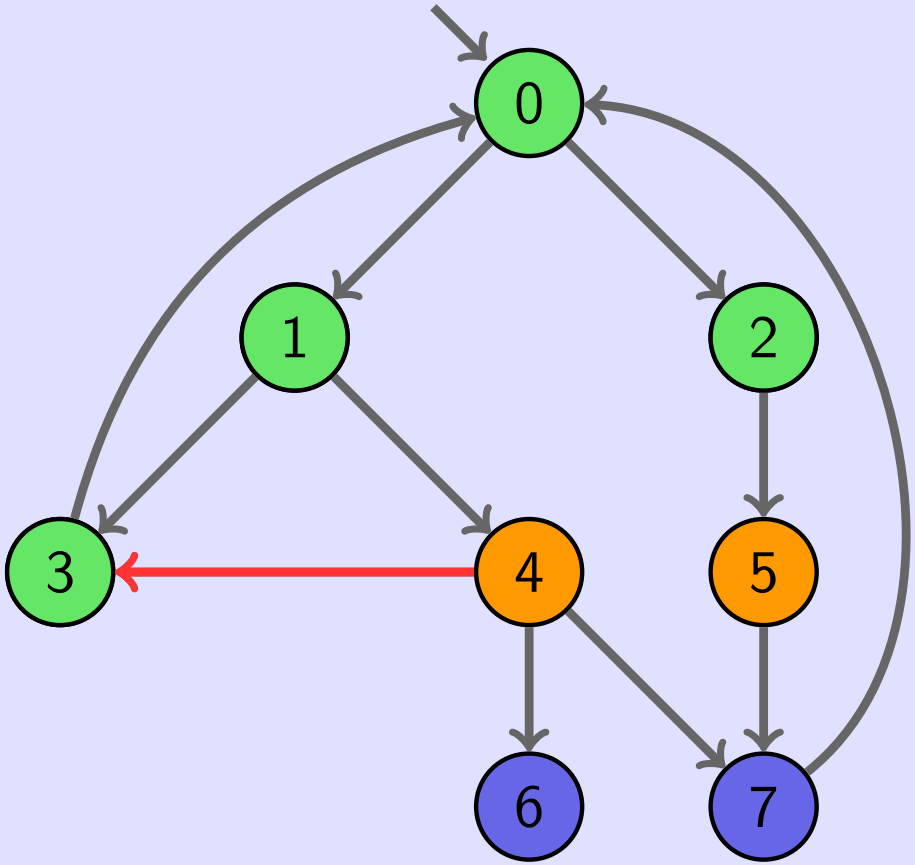
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

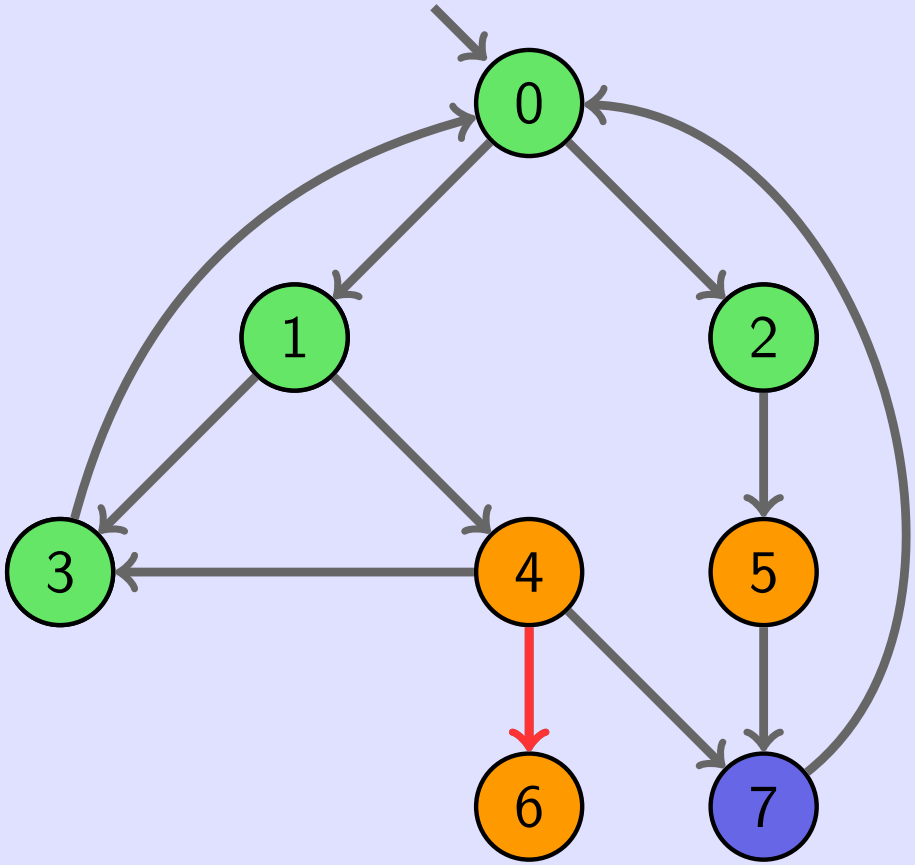
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

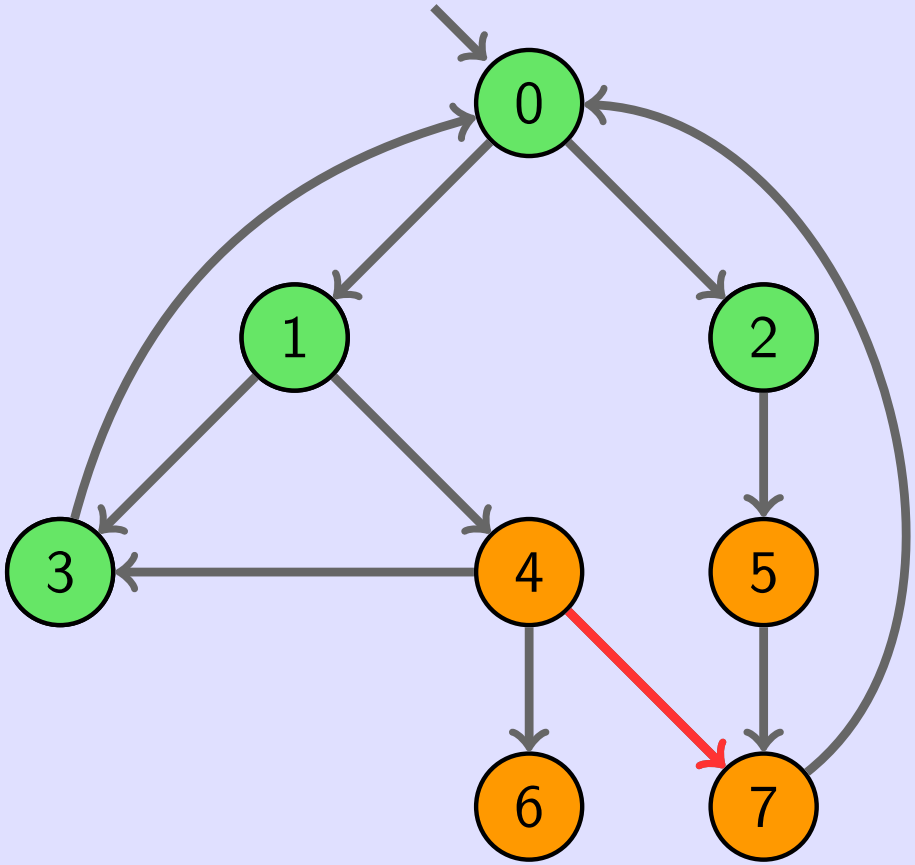
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

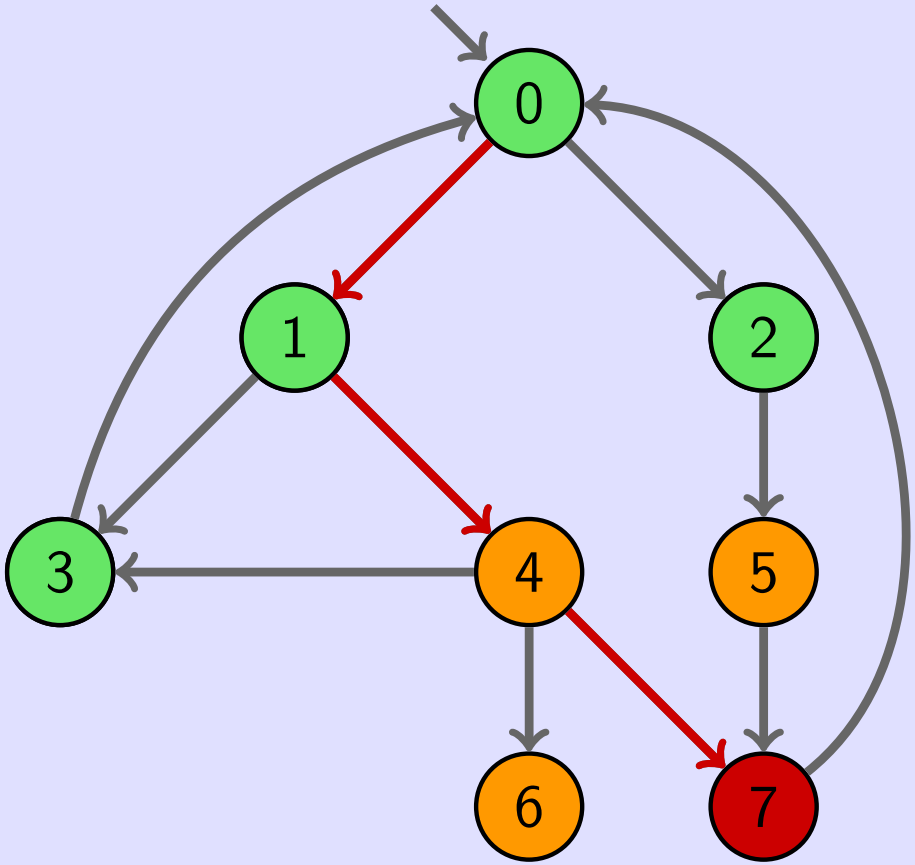
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS

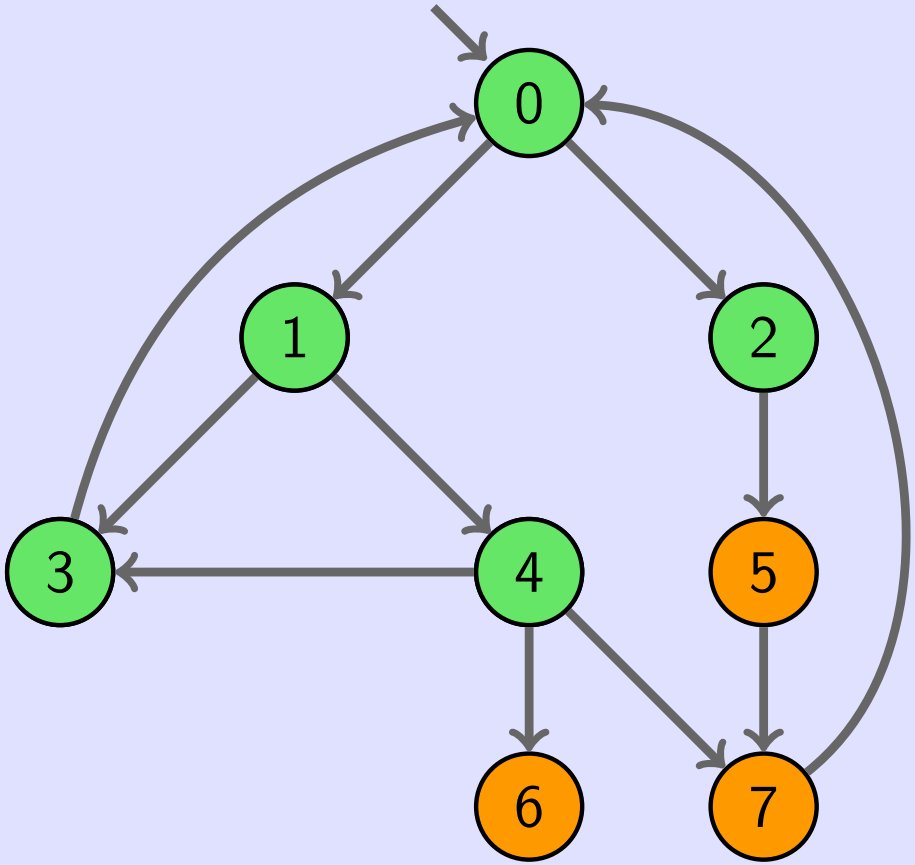


We found an error trace

$0 \rightarrow 1 \rightarrow 4 \rightarrow 7$

GSEA — General State Exploring Algorithm

Example of a BFS



open state

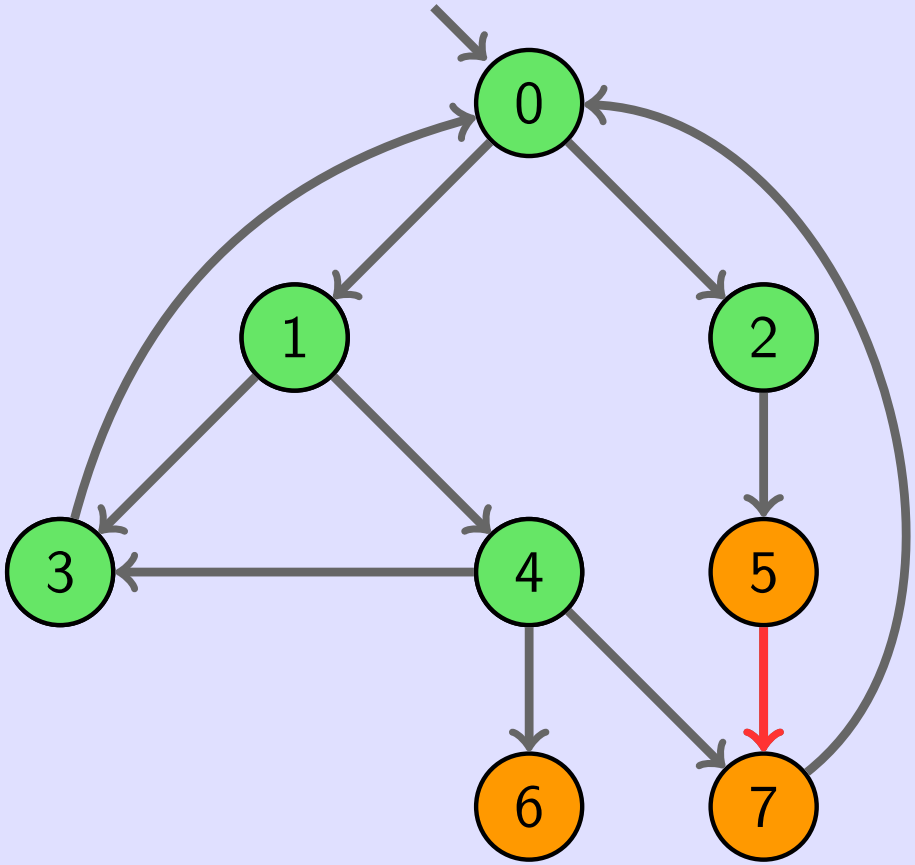
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

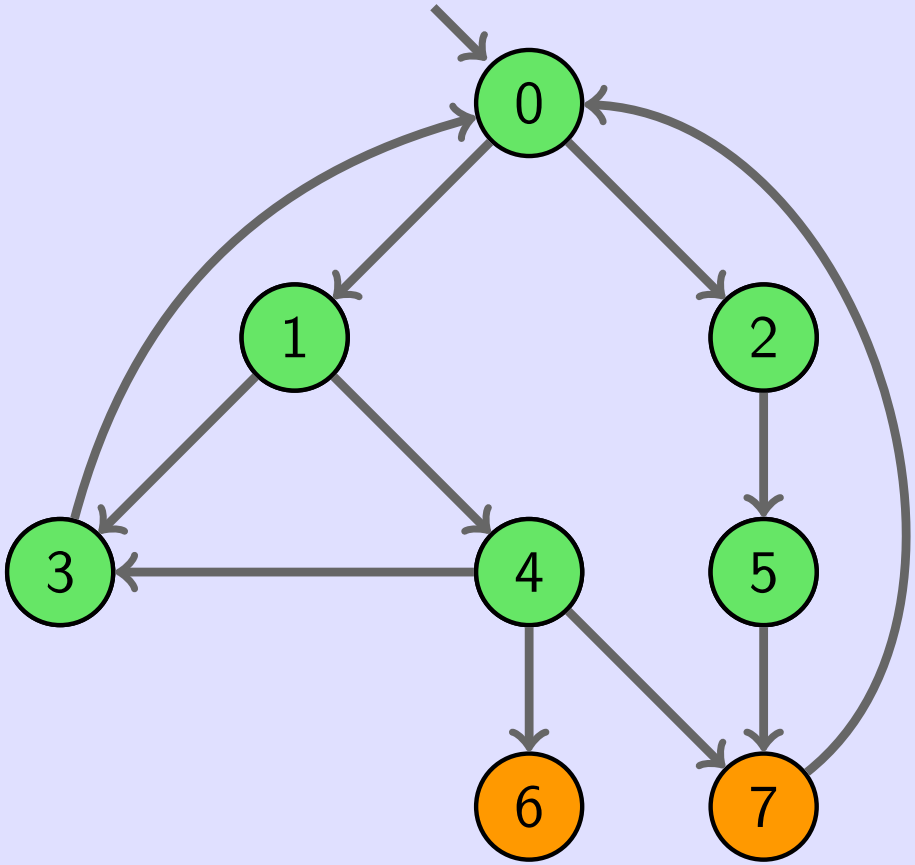
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

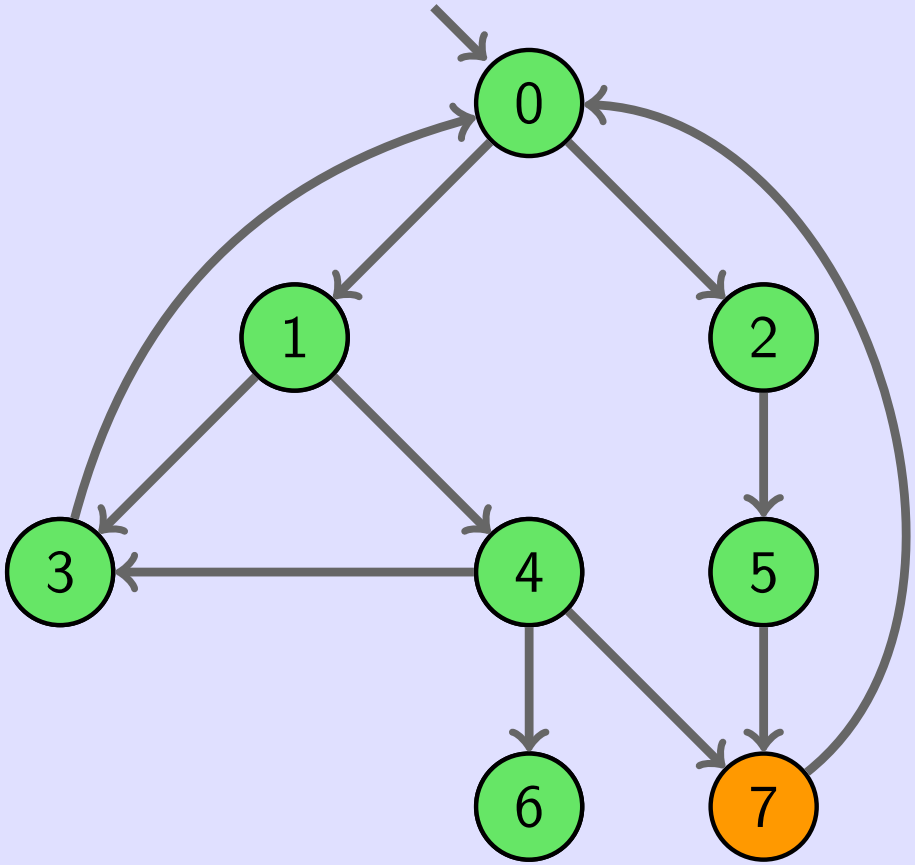
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

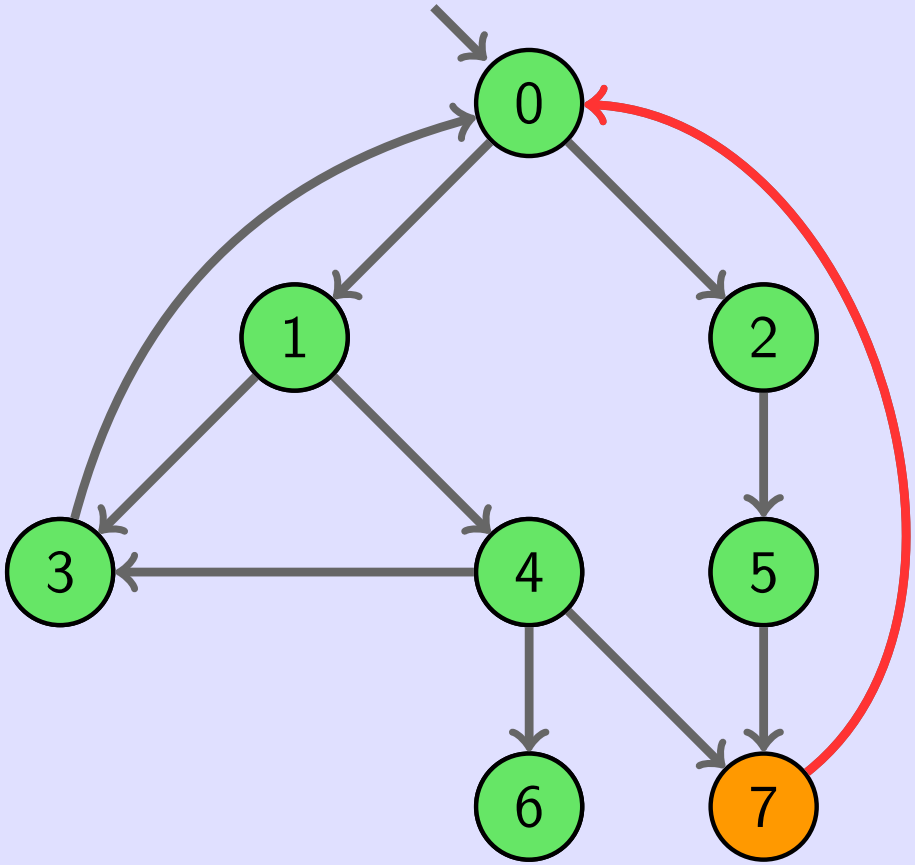
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

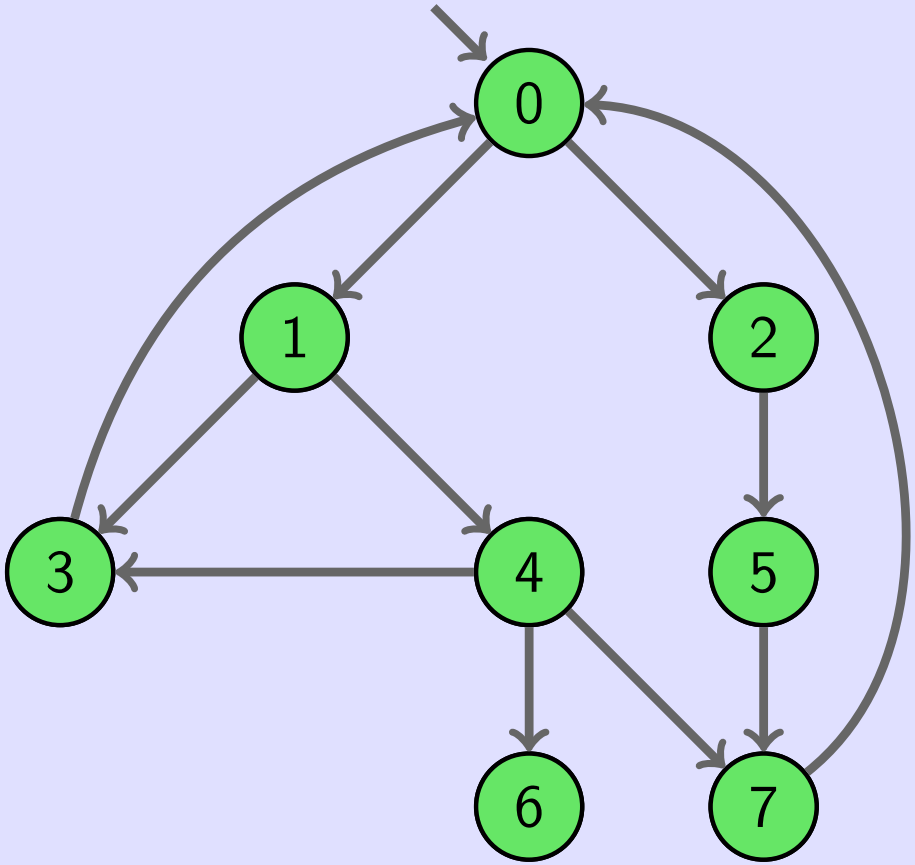
closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Example of a BFS



open state

closed state

unmet state

$\longrightarrow$ transition executed

GSEA — General State Exploring Algorithm

Principle

A GSEA

- is a generic graph search algorithm that partitions the state space upon

  open states (seen but not explored yet)
  closed states (explored)
  unmet states

- can be instantiated in various ways depending on the implementation of the open set
  - DFS $\Leftrightarrow$ the open set is a stack
  - BFS $\Leftrightarrow$ the open set is a fifo-queue
  - Best-First search $\Leftrightarrow$ the open set is a priority-queue
- can be implemented by model checkers for, e.g., checking safety properties

The State Explosion Problem

- for real-world systems the state space can be huge
- typical state vector size: $10 - 10^4$ bytes

$\Rightarrow$ the RAM hash table implementing the closed set cannot contain more than $10^7 - 10^8$ states

The State Explosion Problem

- for real-world systems the state space can be huge
- typical state vector size: $10 - 10^4$ bytes

$\Rightarrow$ the RAM hash table implementing the closed set cannot contain more than $10^7 - 10^8$ states

- to reduce the state space
  - partial-order reduction
  - symmetry reduction

The State Explosion Problem

- for real-world systems the state space can be huge
- typical state vector size: $10 - 10^4$ bytes

$\Rightarrow$ the RAM hash table implementing the closed set cannot contain more than $10^7 - 10^8$ states

- to reduce the state space
  - partial-order reduction
  - symmetry reduction
- to push the limit further
  - store an approximation of the closed set, e.g., the hash value of states
  - store states on disk
  - distribute the search over a network of workstations
  - forget some previously visited states states
    - **state caching**
    - **sweep-line reduction**

  termination is the main issue for these methods

State Caching
Principle

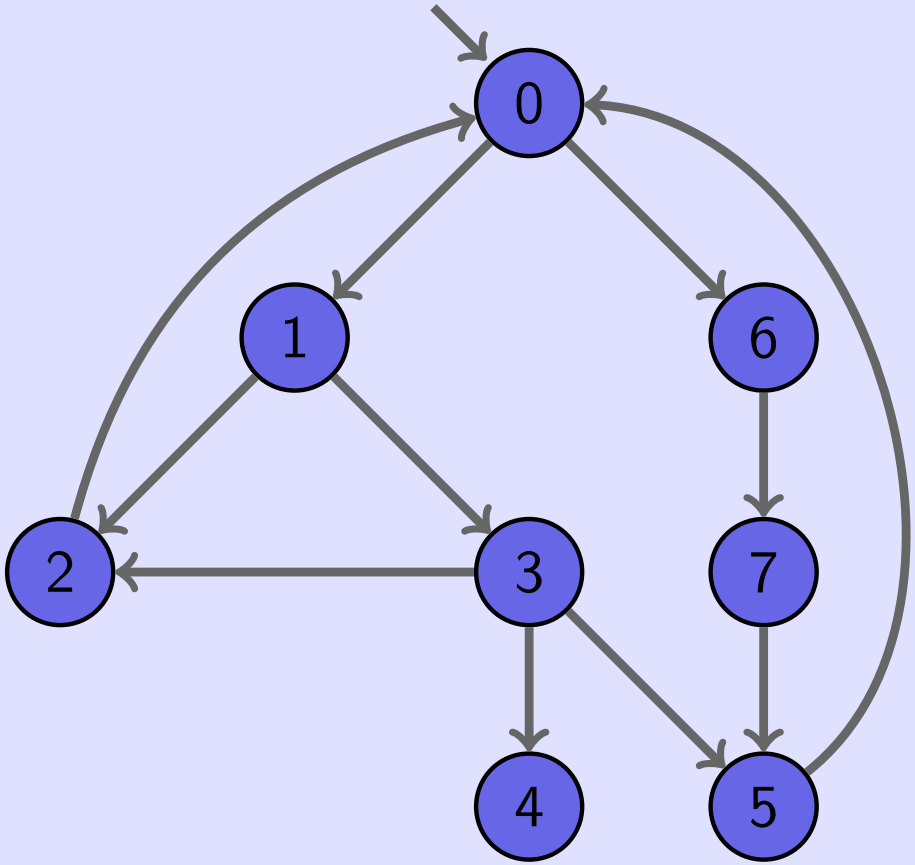
- using DFS, any cycle will reach a state on the DFS stack

$\Rightarrow$ termination is guaranteed by storing the stack

- principle of state caching:
  - always keep the DFS stack in memory, i.e., the open set
  - if we lack memory delete some closed states

  $\Rightarrow$ we may revisit a visited state that has been deleted from the closed set

State Caching
Example



open state

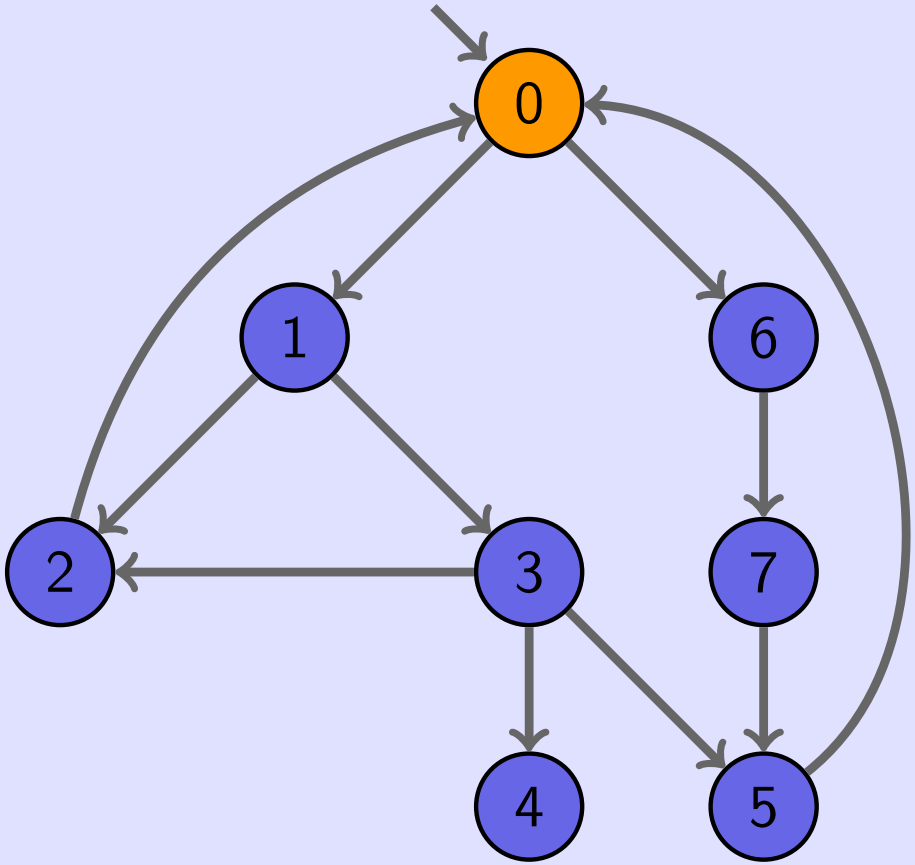
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

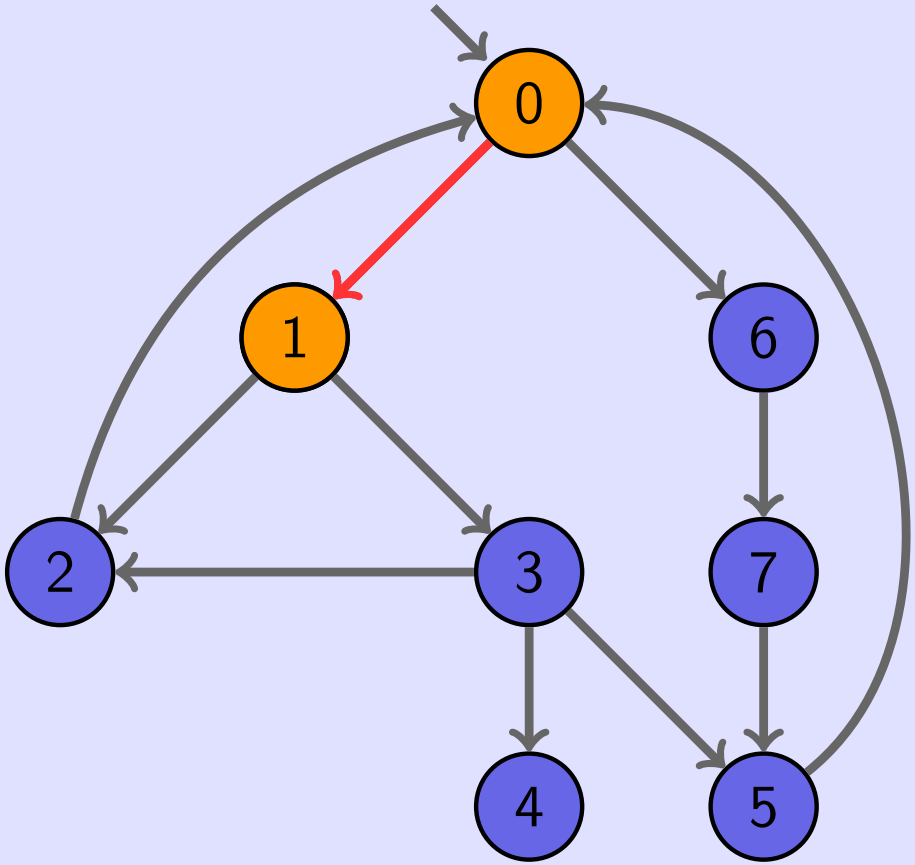
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

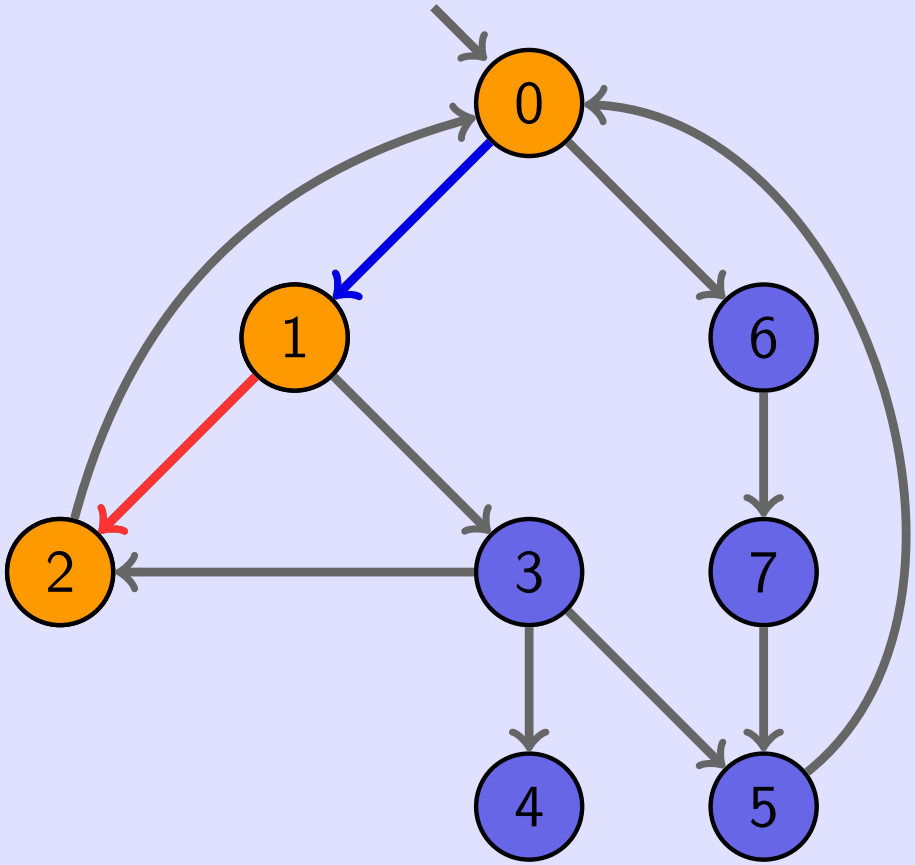
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

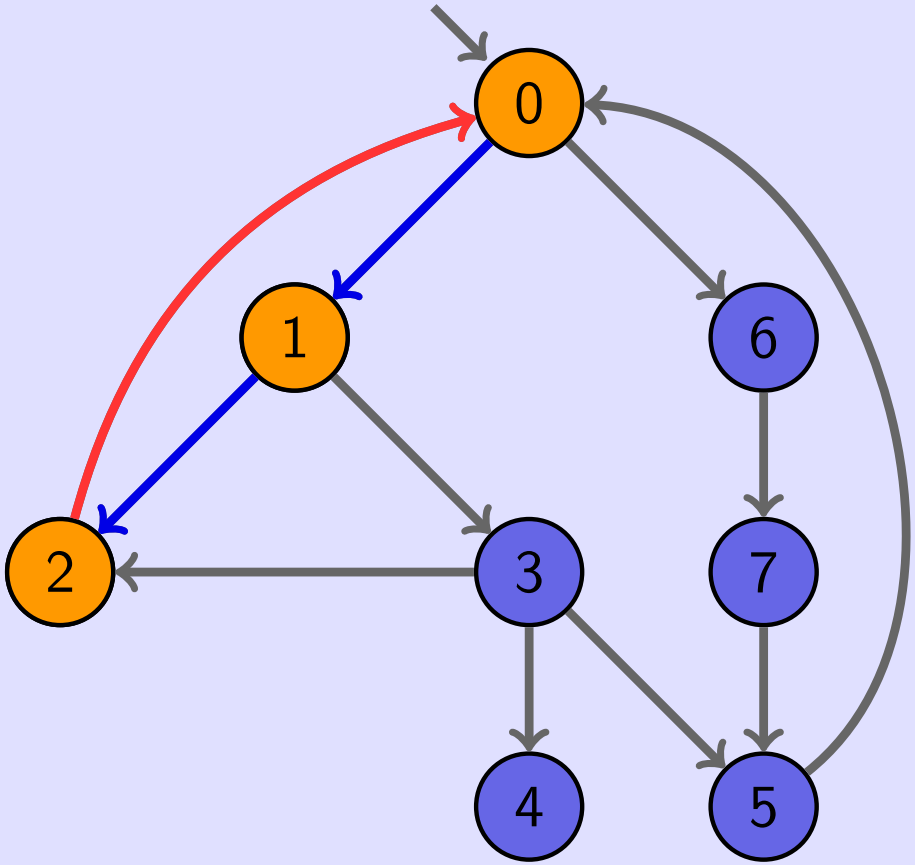
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

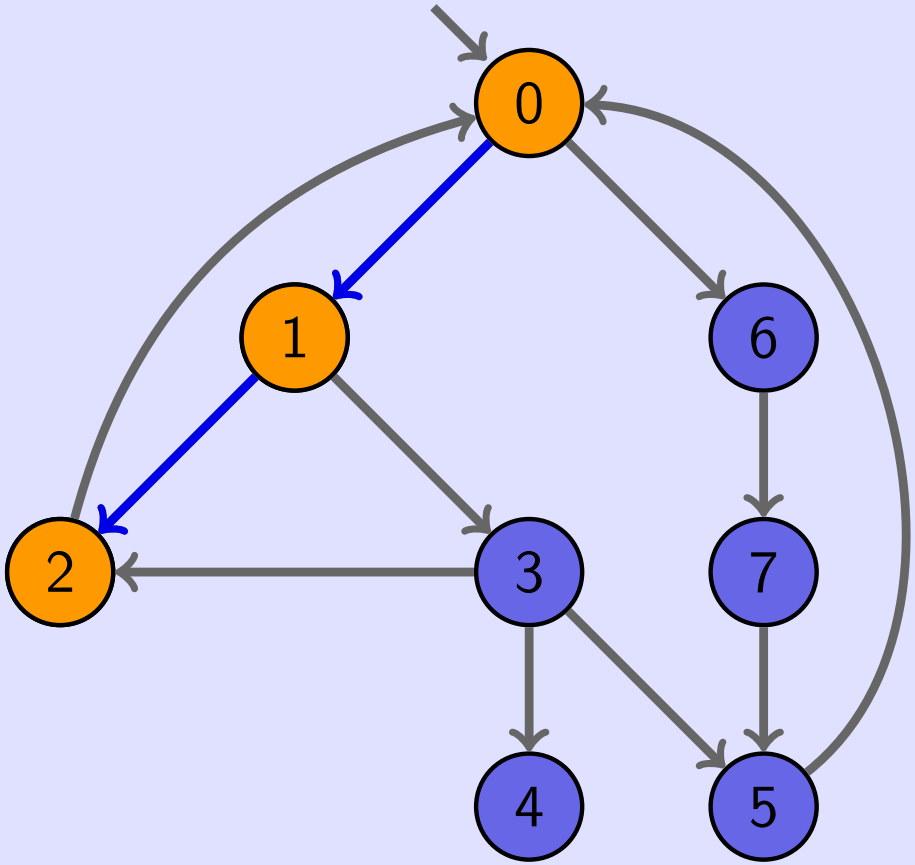
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

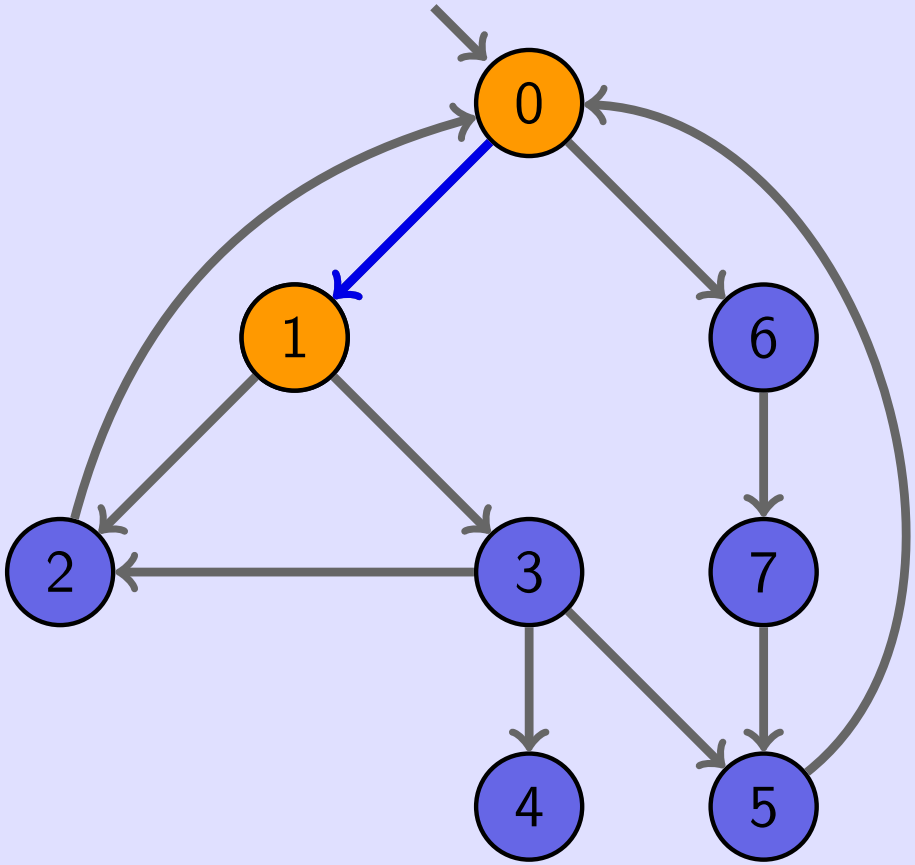
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

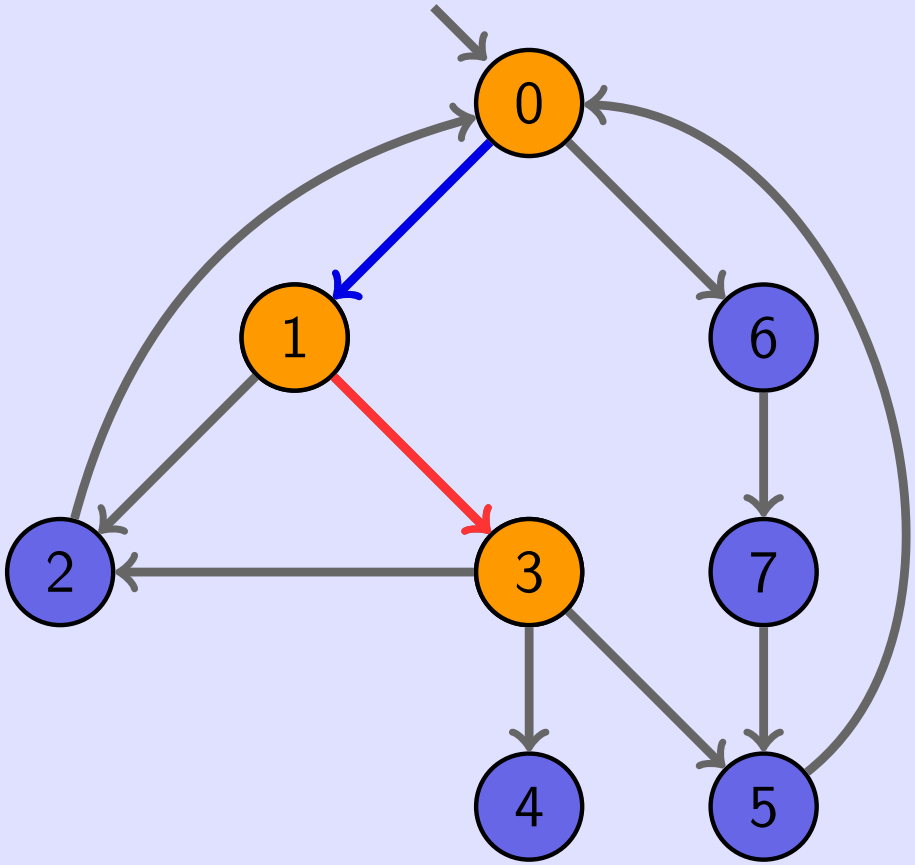
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

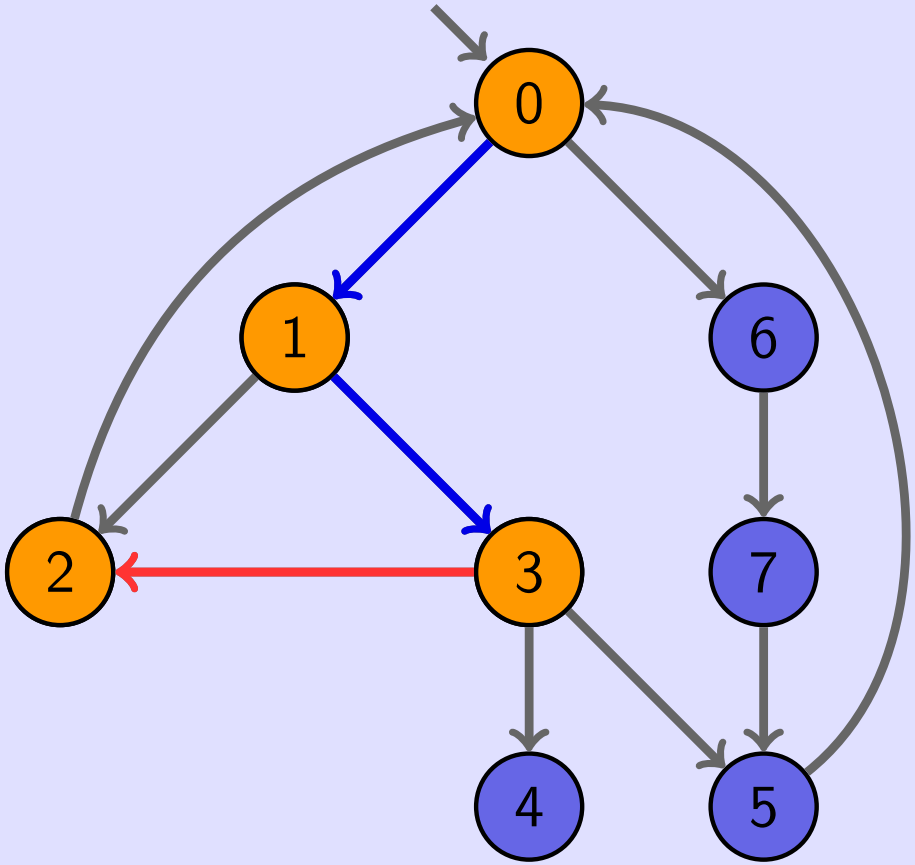
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

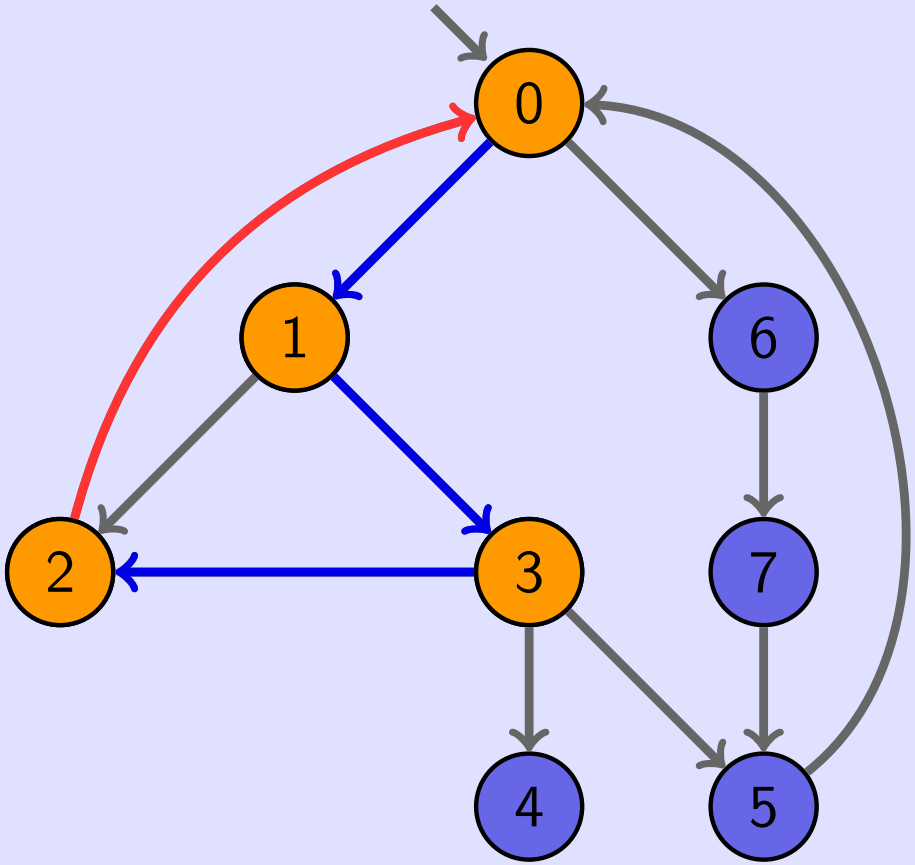
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

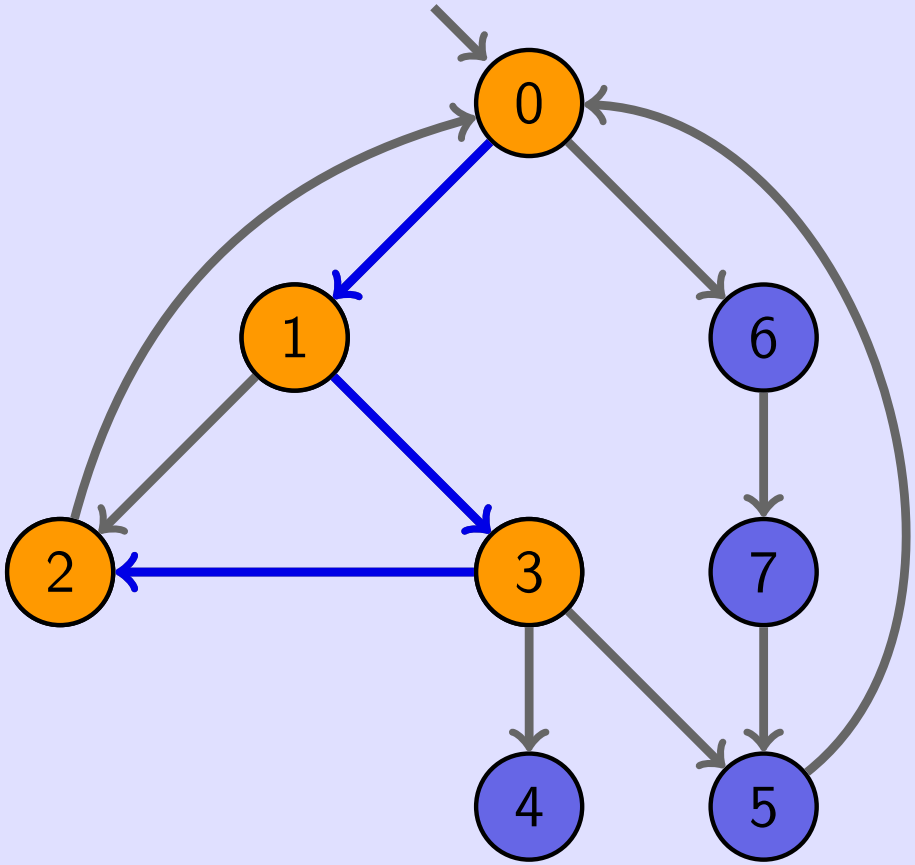
closed state

unmet state

⟶ transition executed

⟶ transition on the DFS stack

State Caching
Example



open state

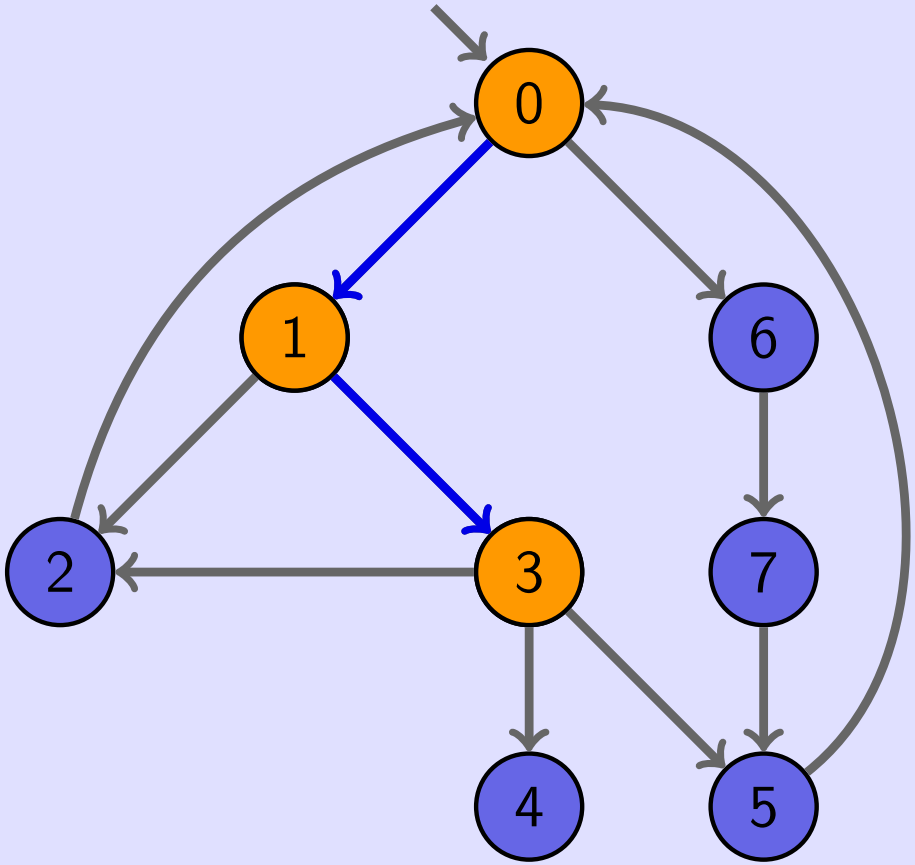
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

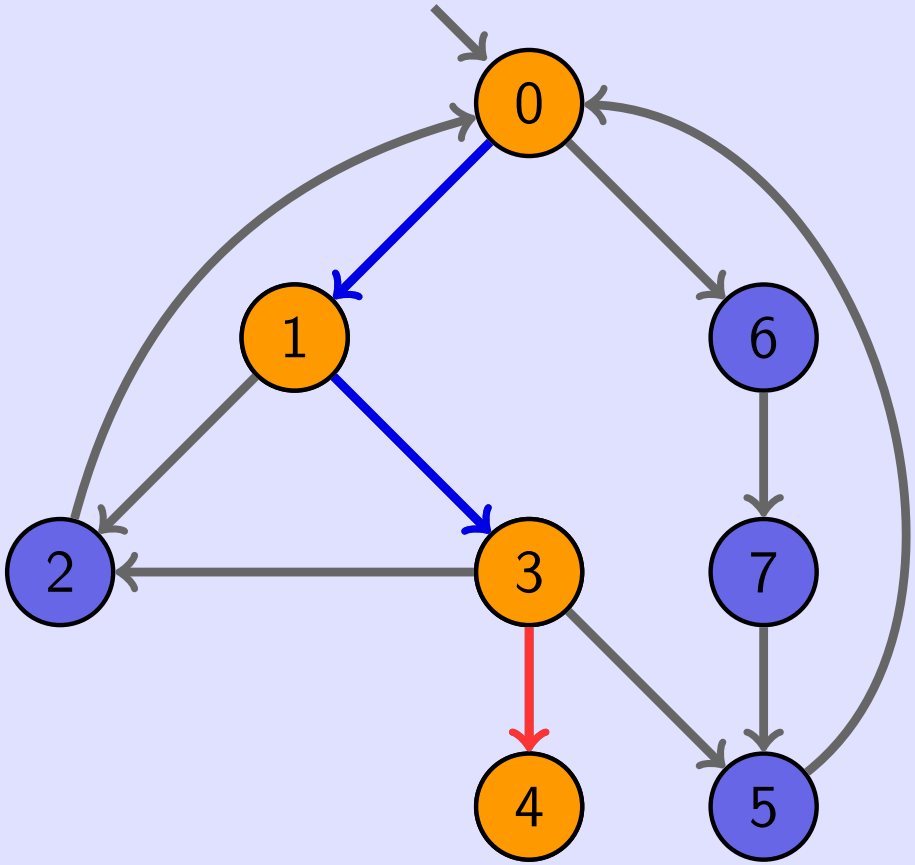
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

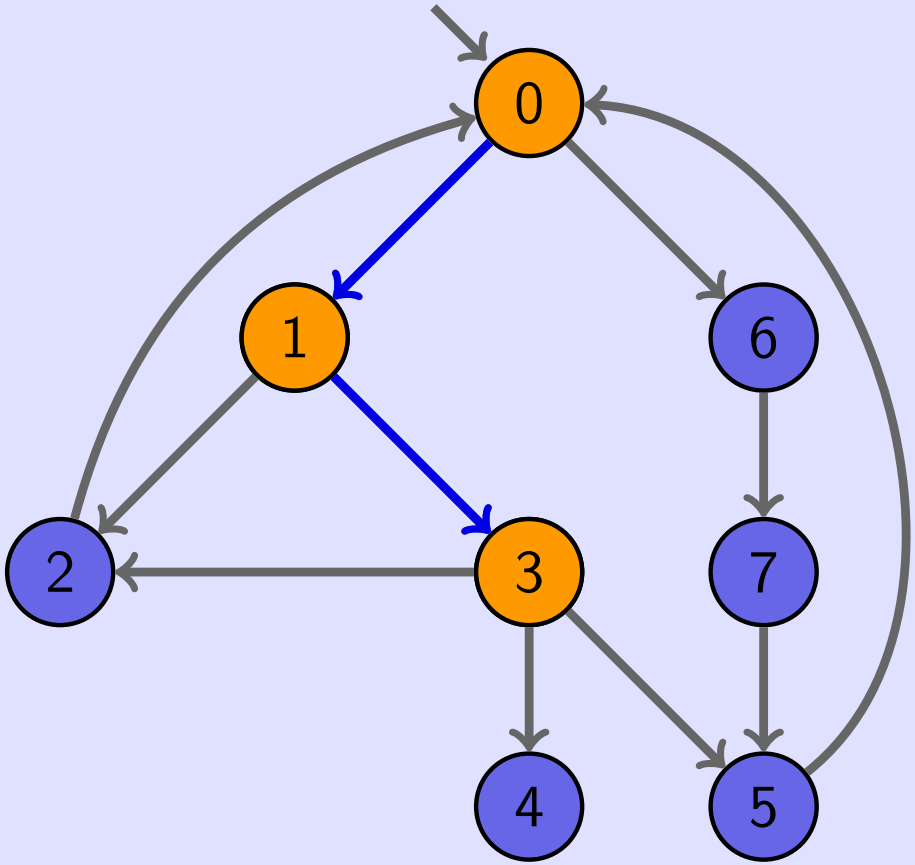
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

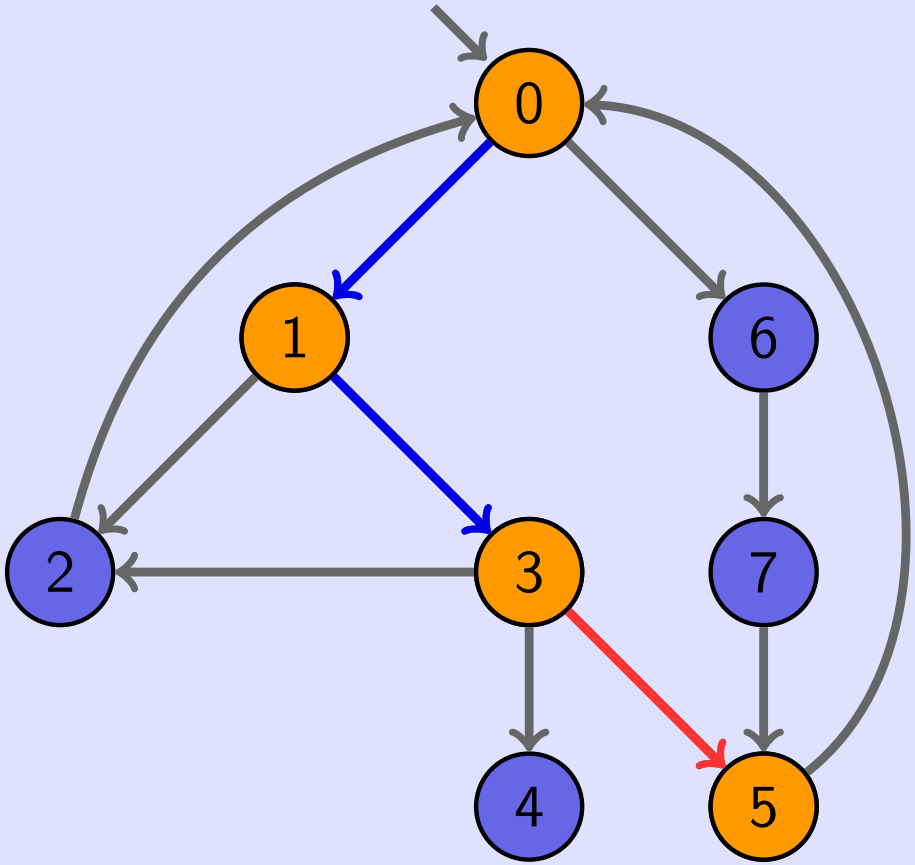
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

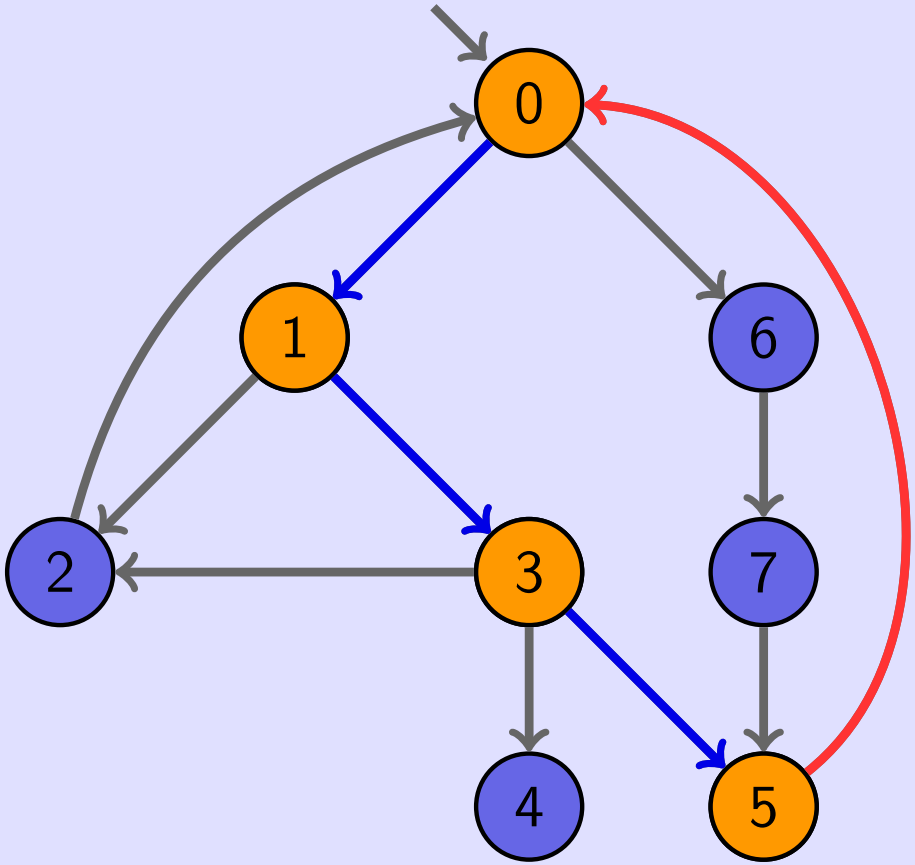
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

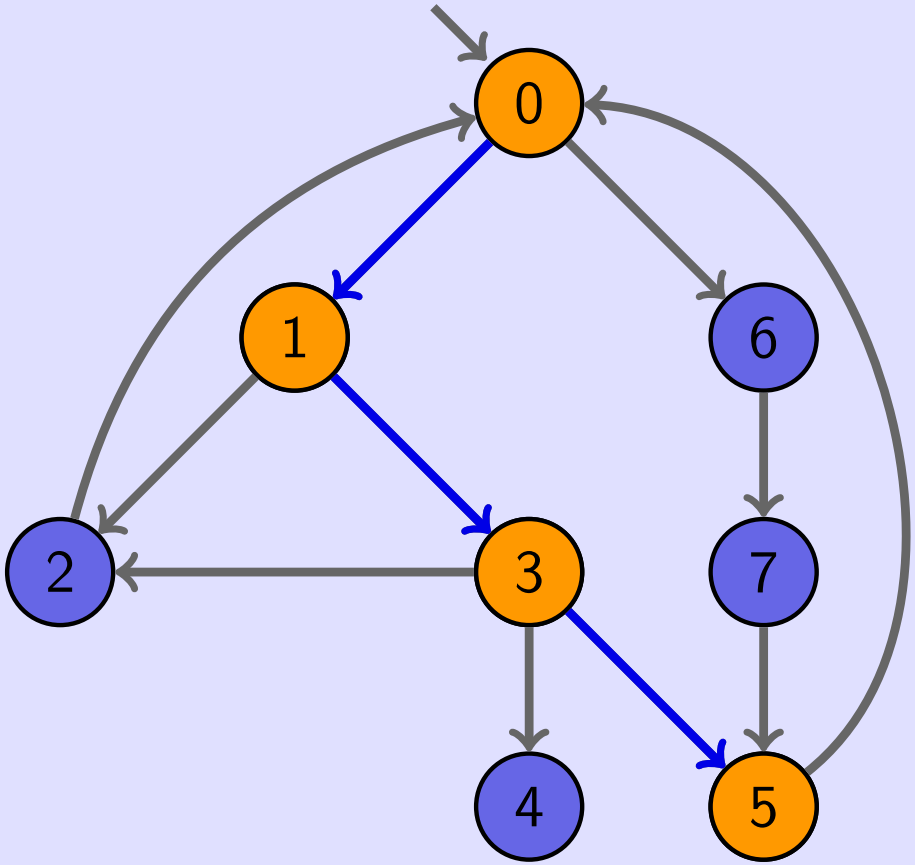
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

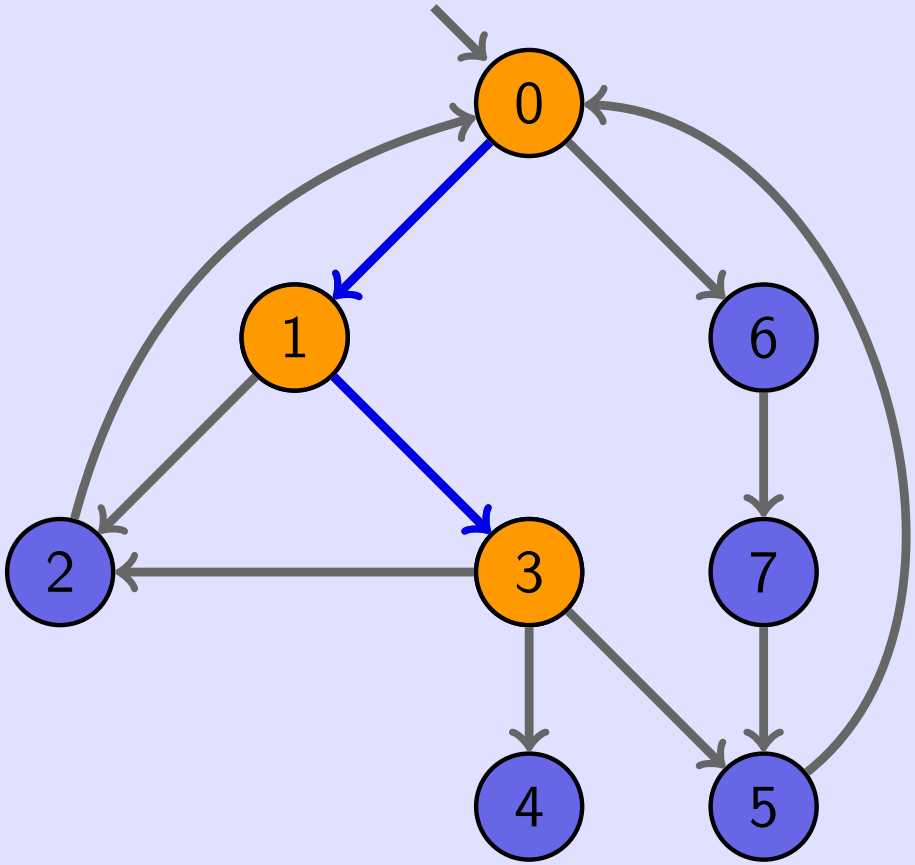
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

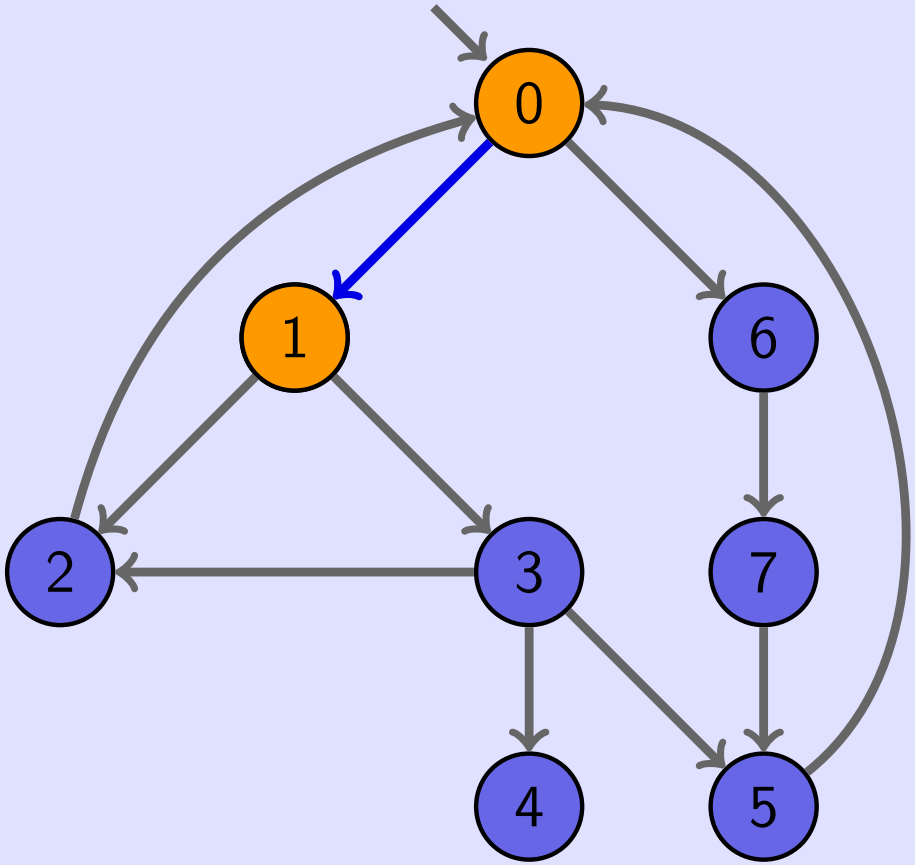
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

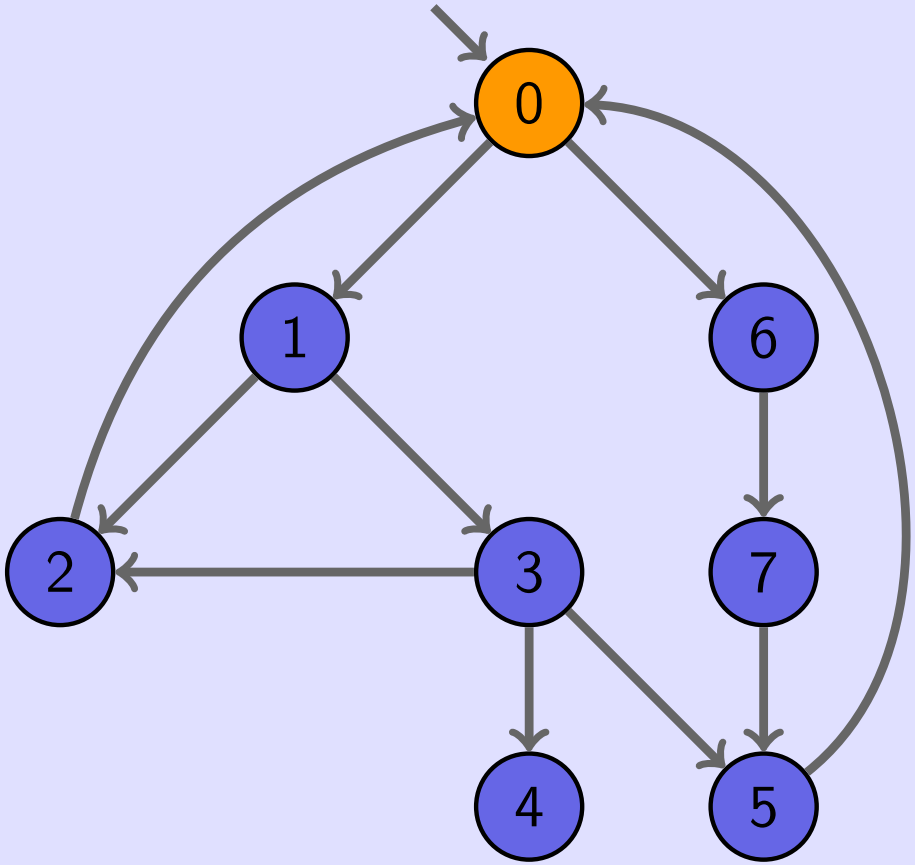
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

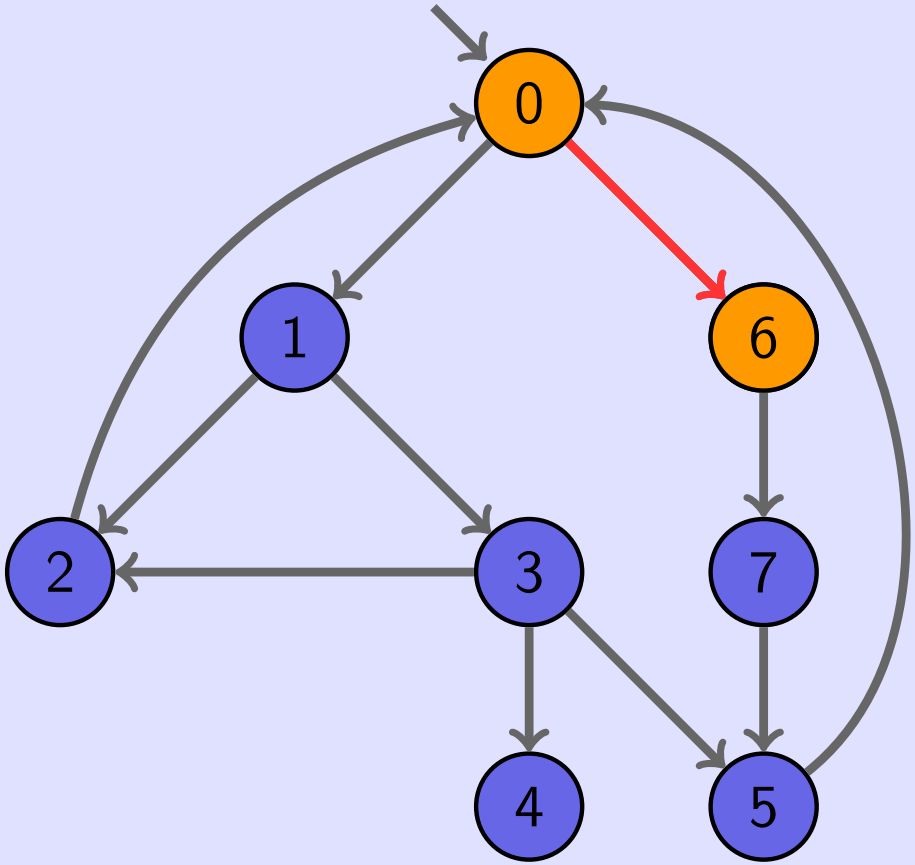
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

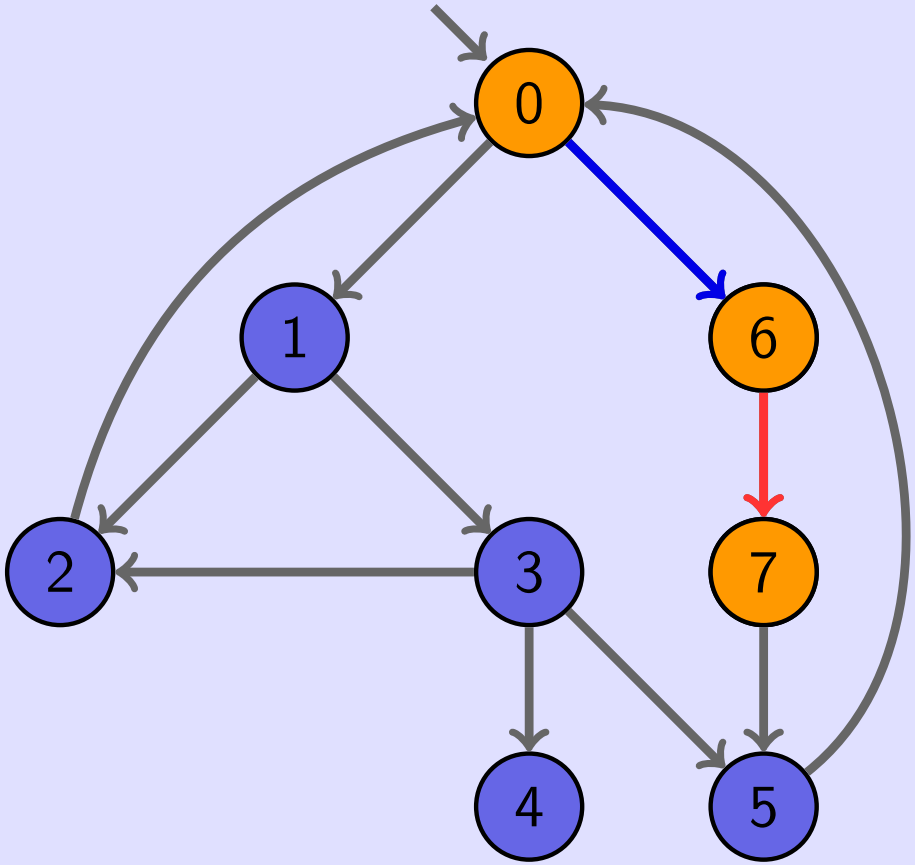
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

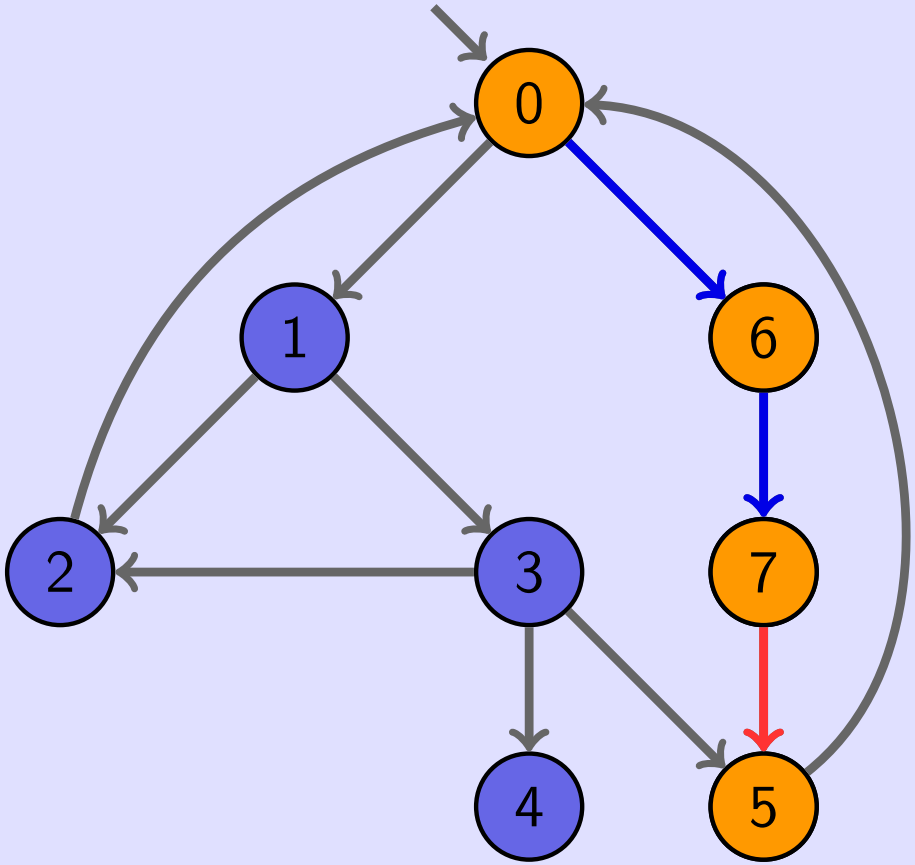
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

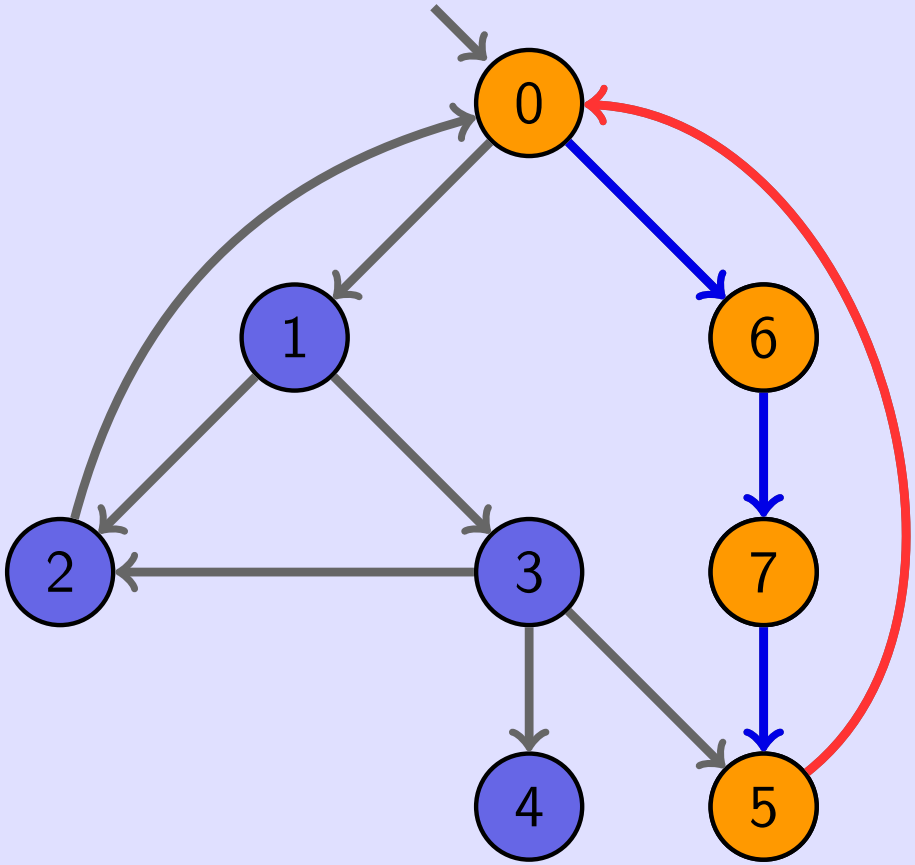
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

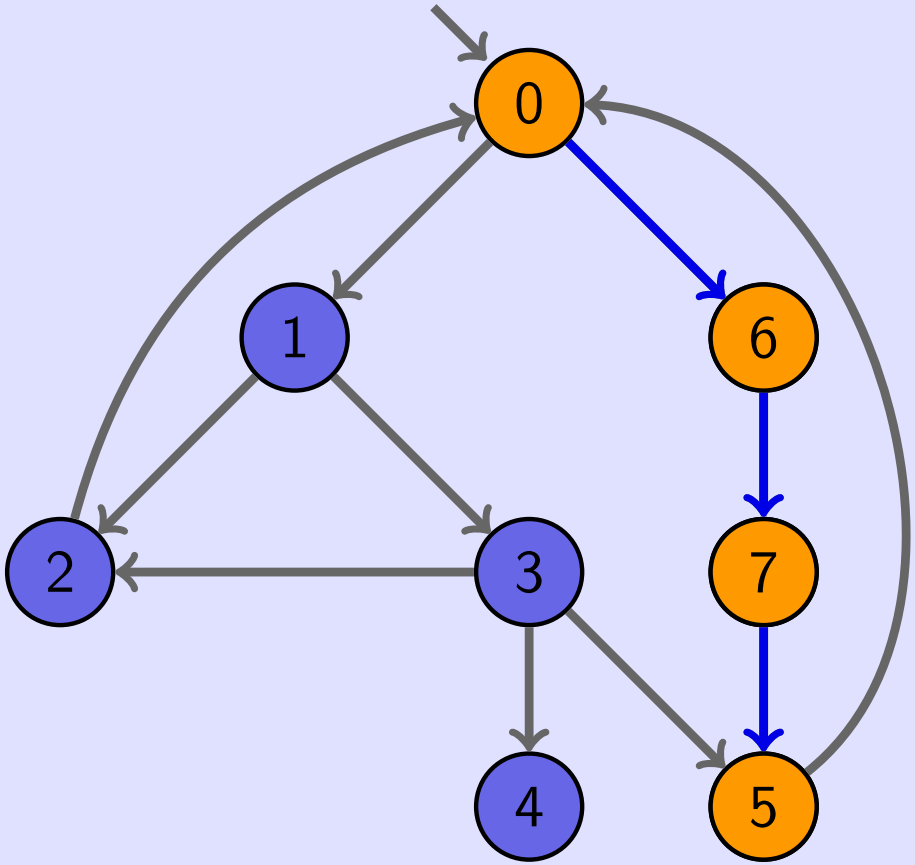
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

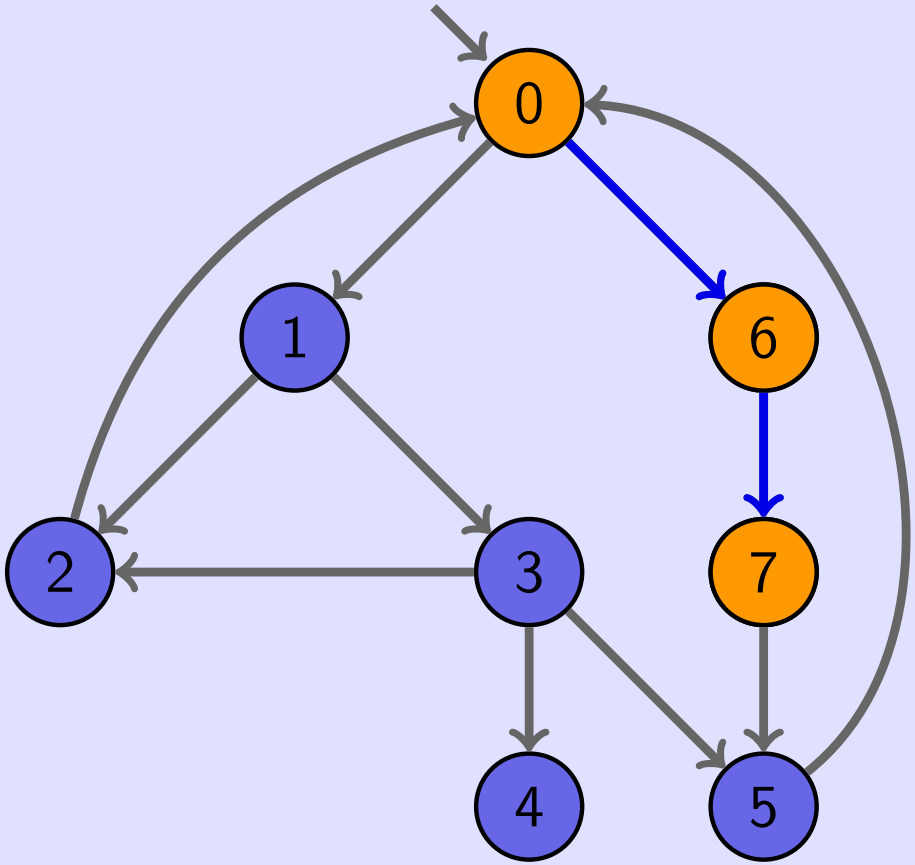
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

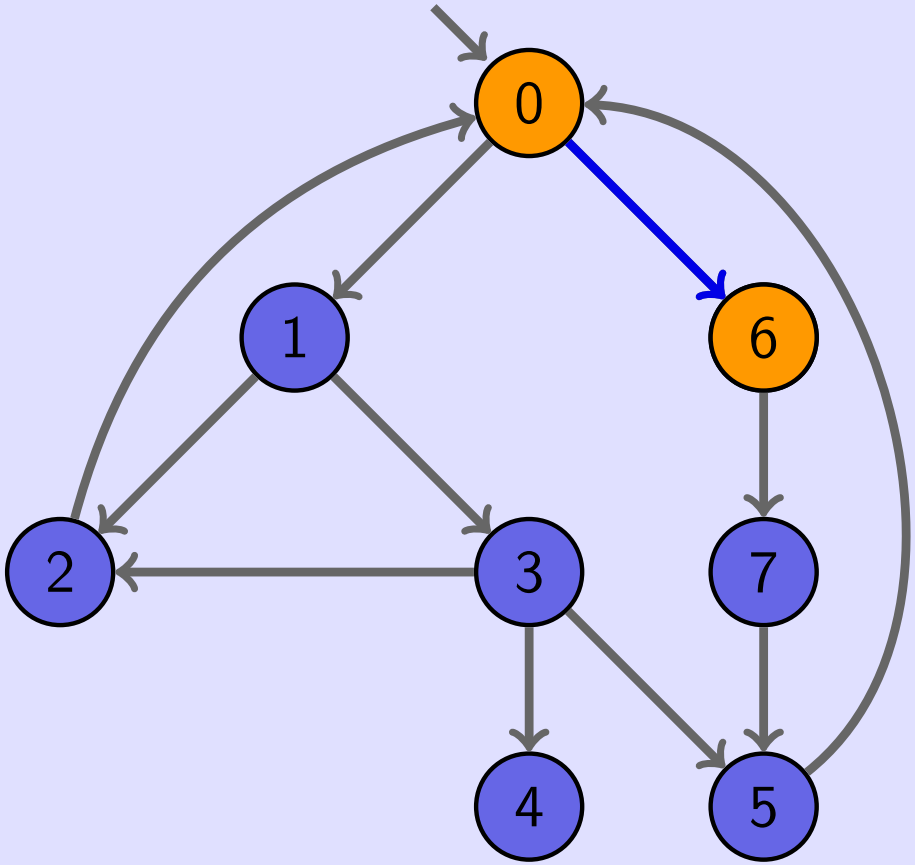
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

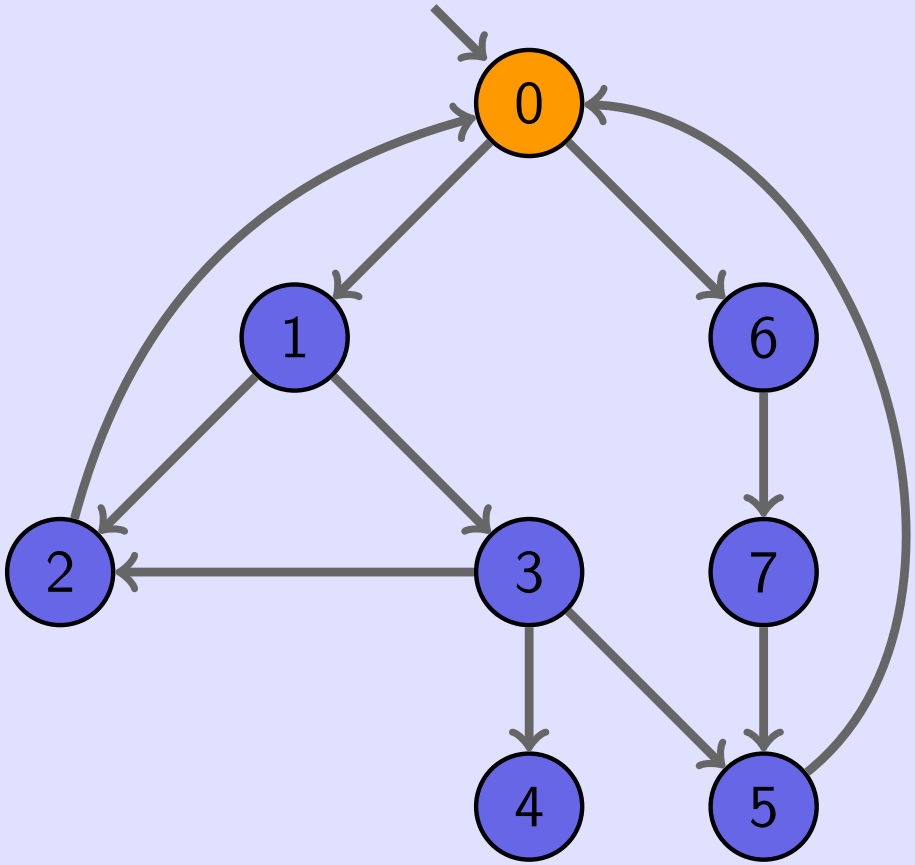
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

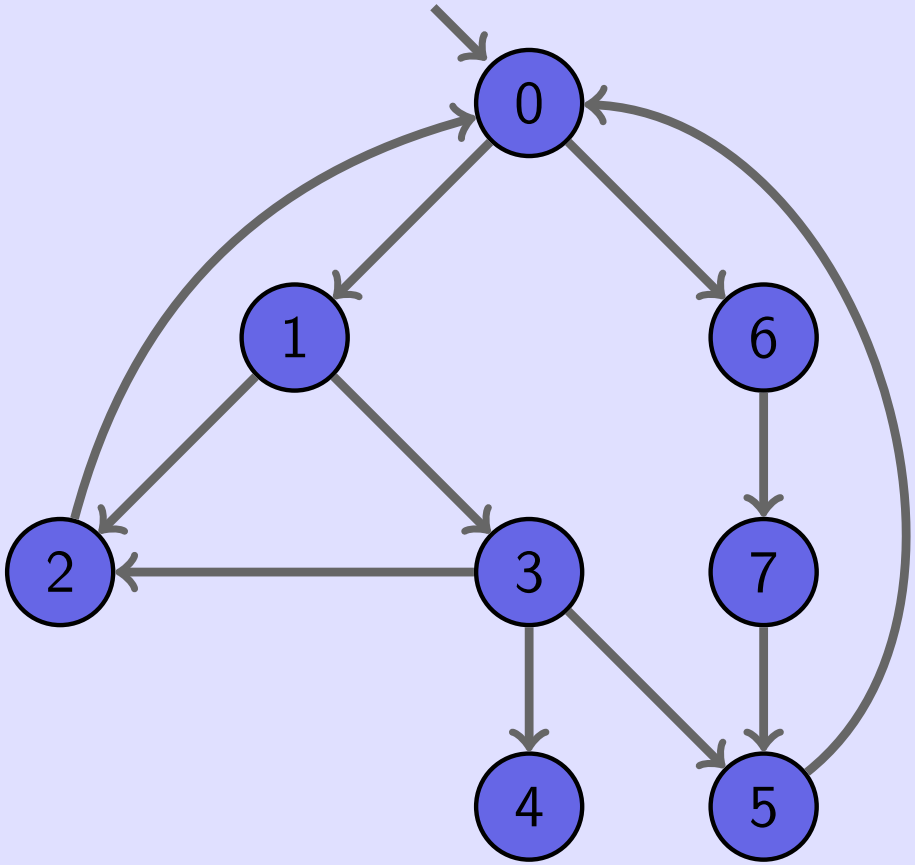
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

State Caching
Example



open state

closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition on the DFS stack

Motivations

- state caching only works for DFS
- DFS is useful for, e.g., detecting cycles, finding SCCs
- but
  - DFS cannot be (efficiently) parallelized/distributed
  - BFS is better to find short error traces
  - Best-First search can find errors faster

...

Motivations

- state caching only works for DFS
- DFS is useful for, e.g., detecting cycles, finding SCCs
- but
  - DFS cannot be (efficiently) parallelized/distributed
  - BFS is better to find short error traces
  - Best-First search can find errors faster

  ...
- this talk:
  1. generalization of state caching to GSEA
     - problem: GSEA does not have a stack to detect cycles

     $\Rightarrow$ how to guarantee termination?
  2. application to the sweep-line method
  3. experimental comparison of different algorithms using state caching

State caching for GSEA

- problem: how to guarantee termination with a GSEA?
- we have to prevent infinite loops due to cycles
- idea: maintain a tree, the TD-tree (Termination Detection tree) that
  - is a sub-tree of the search tree
  - keep tracks of open states
  - records for each state which predecessor generated it

$\Rightarrow$ states of the TD-tree generated some open state(s)

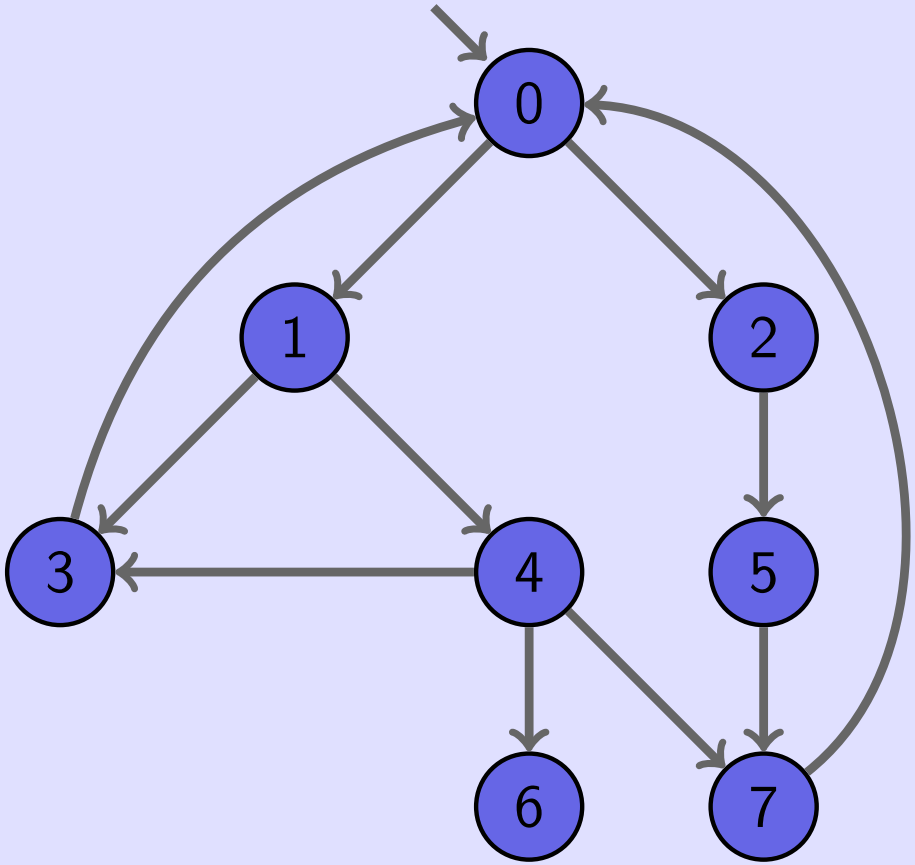
$\Rightarrow$ each cycle will eventually reach a state of the TD-tree

$\Rightarrow$ all closed states that are not part of the TD-tree

- are not required for termination
- can be deleted from memory if we lack of memory

State caching for GSEA

Example



open state

closed state

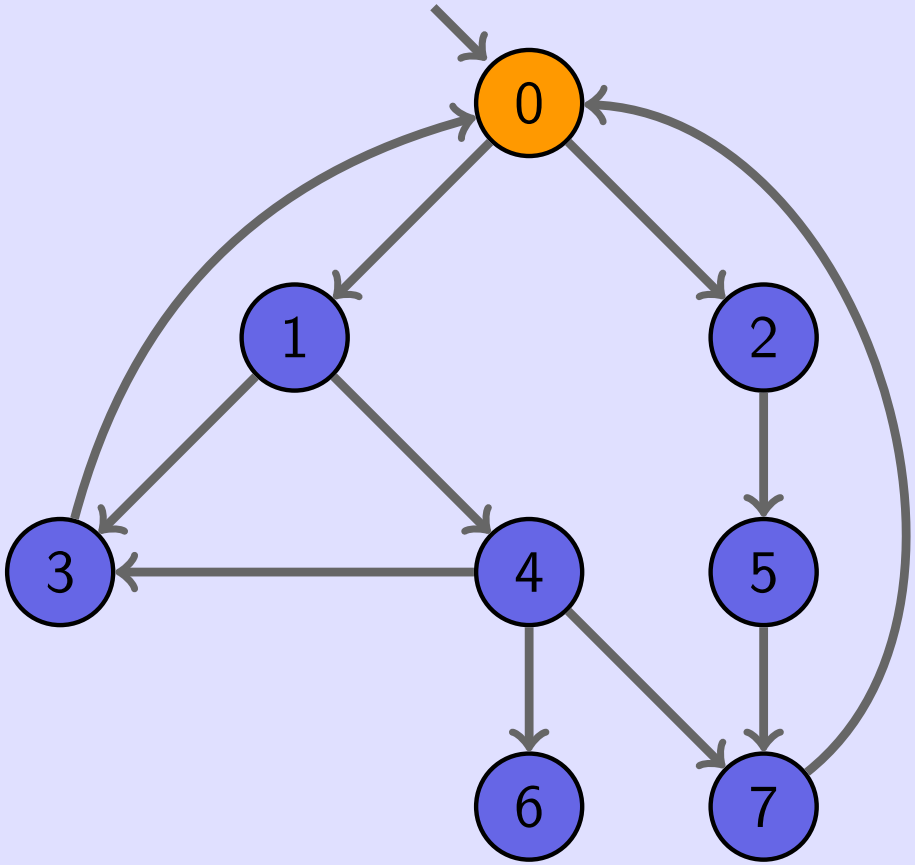
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

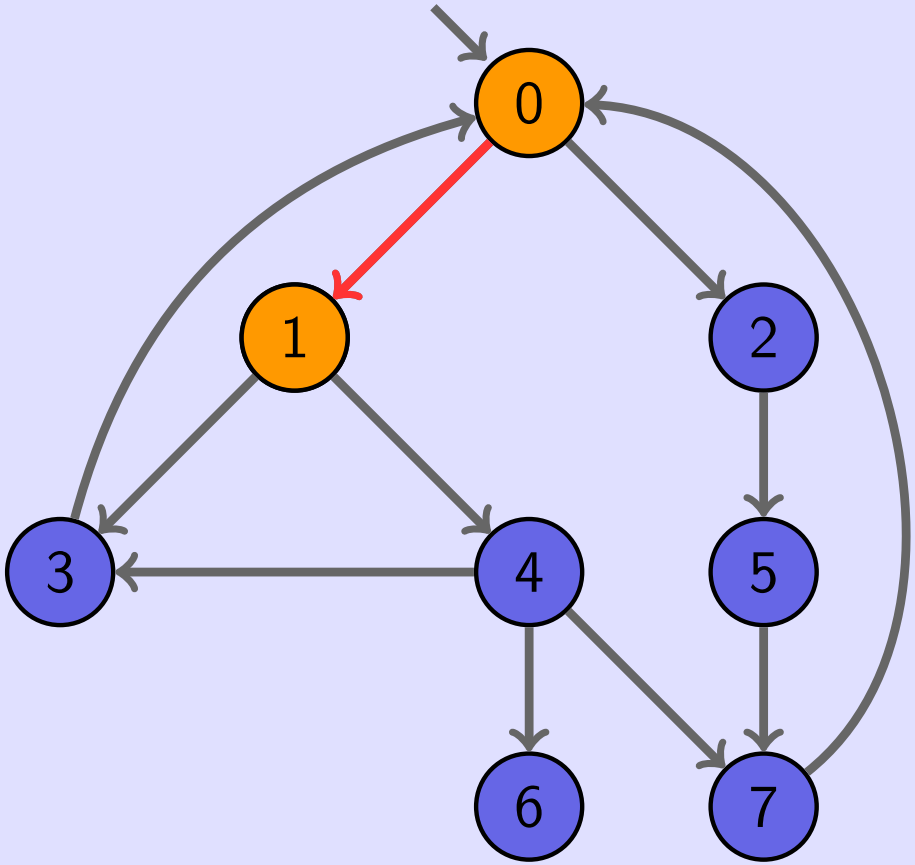
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

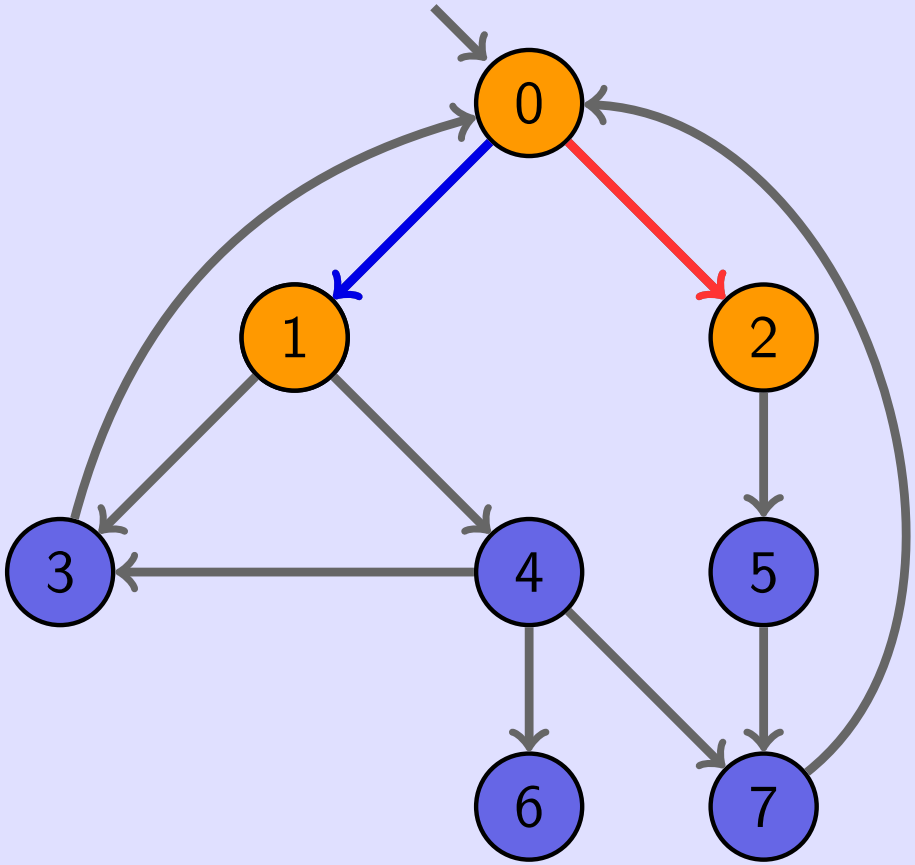
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

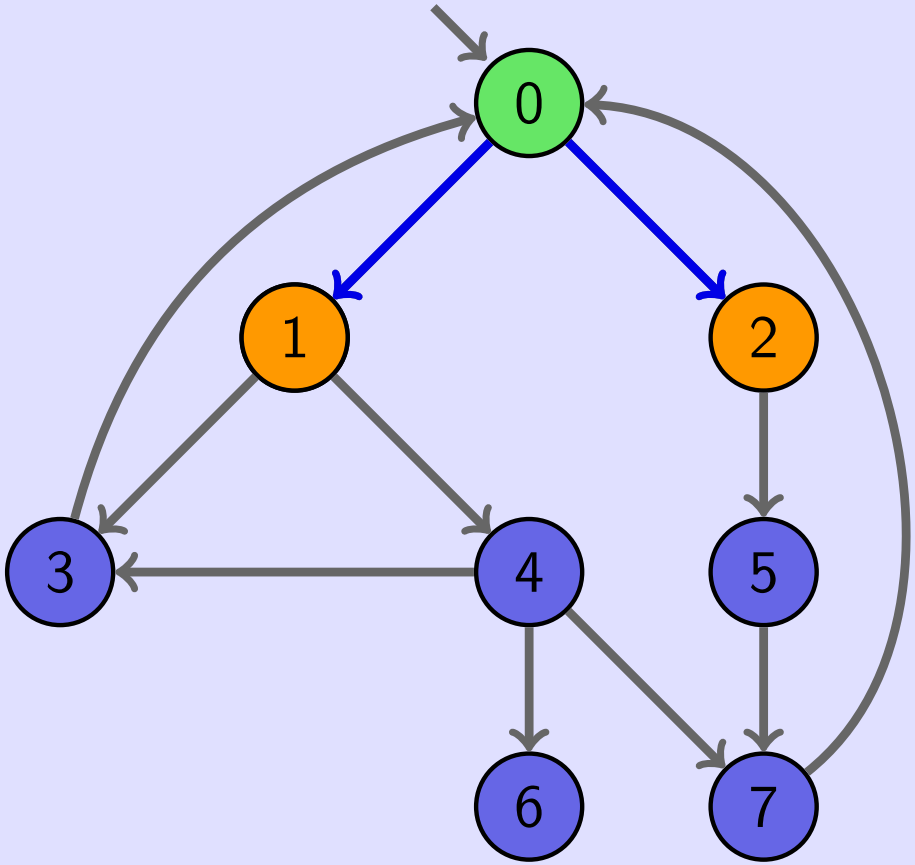
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

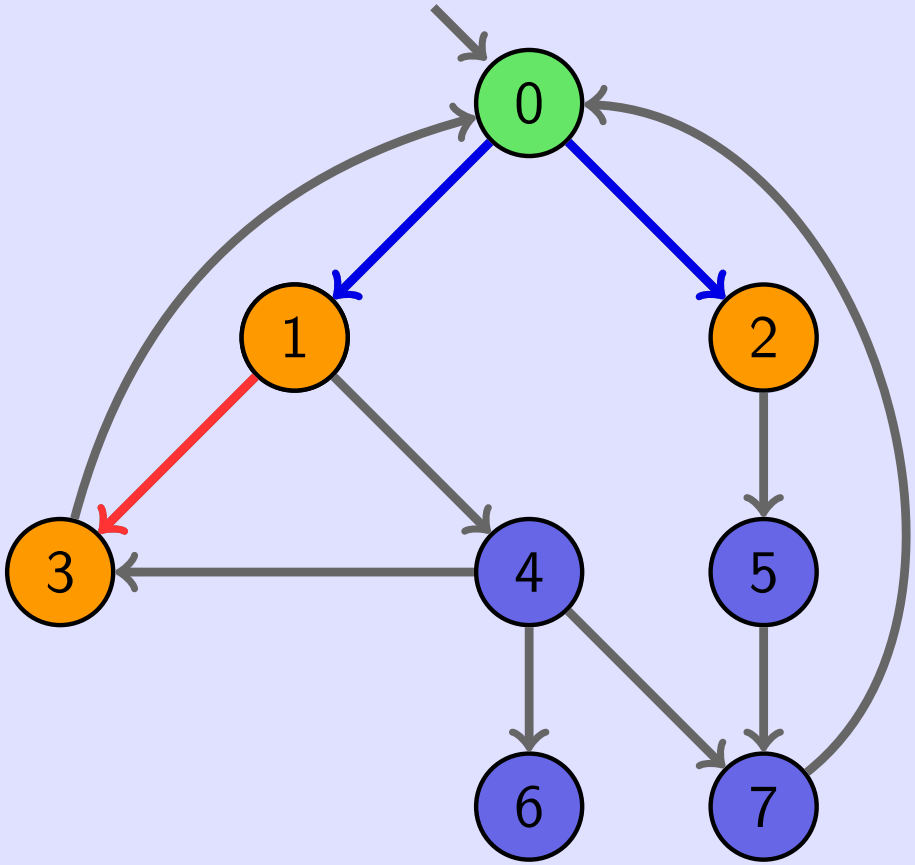
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

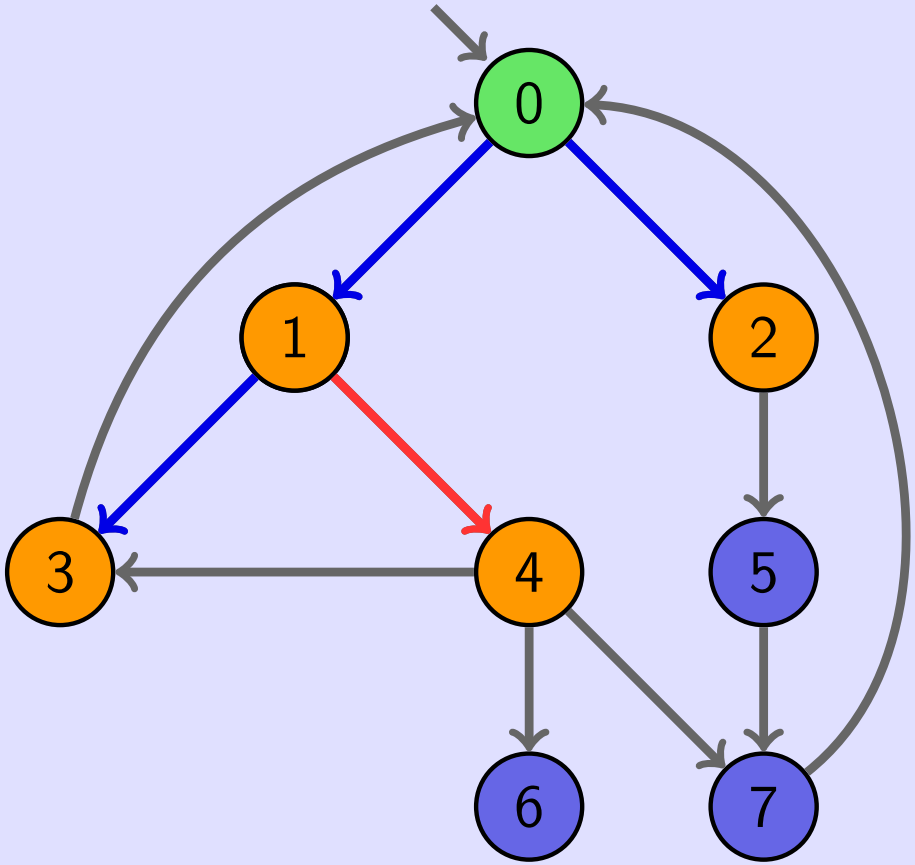
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

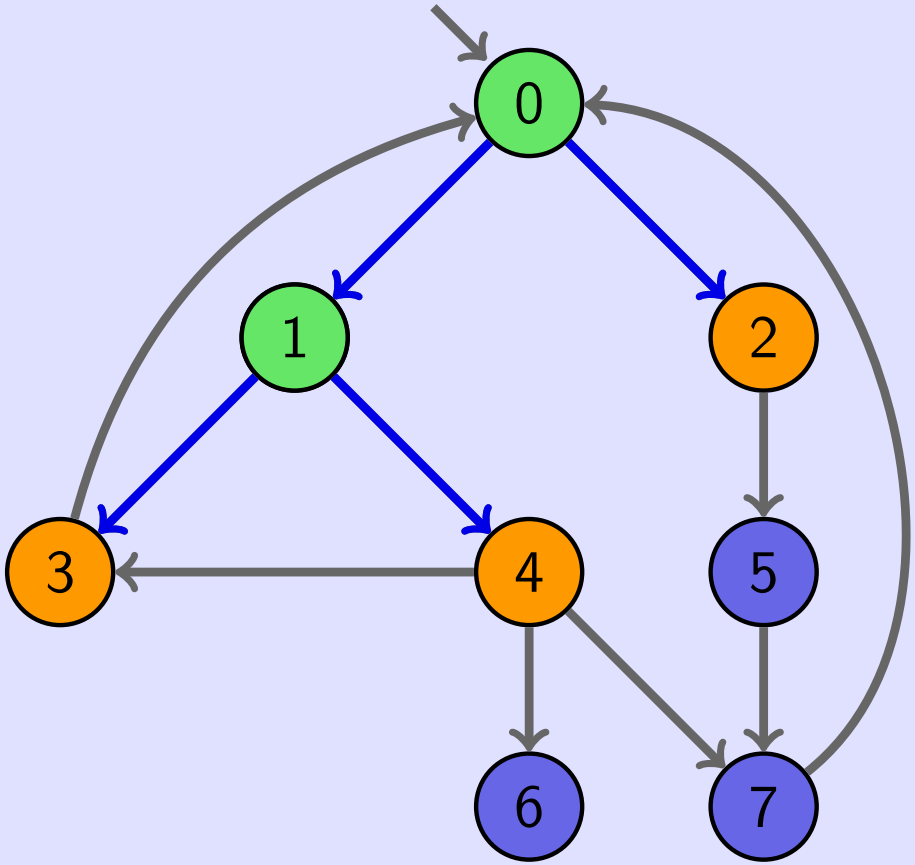
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

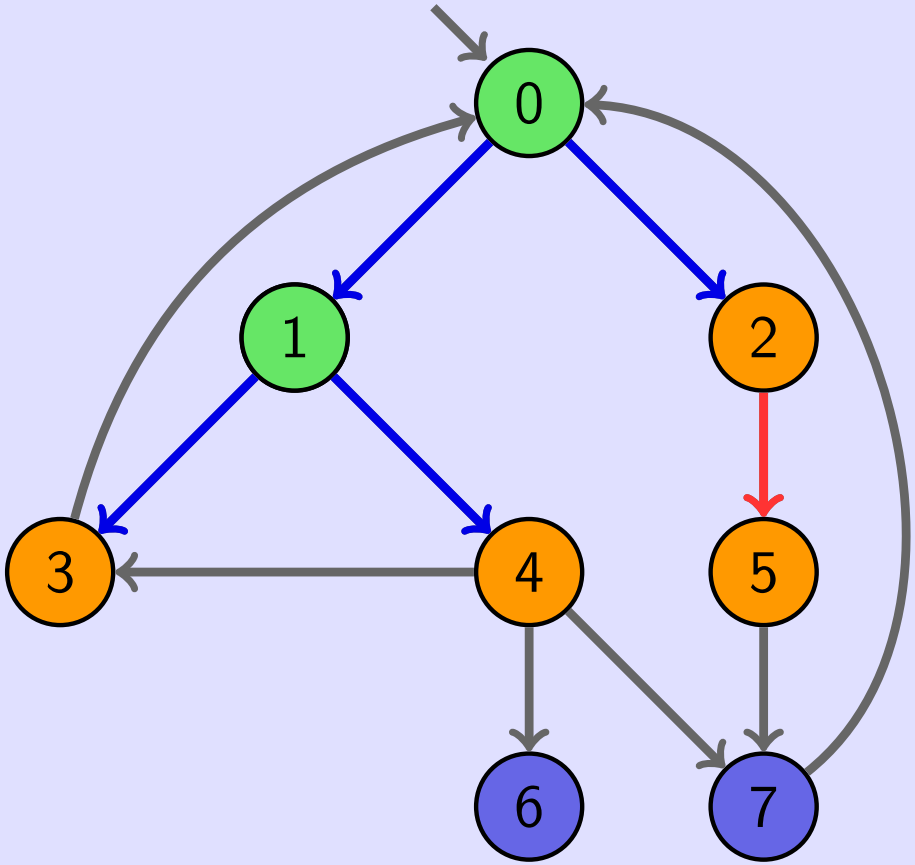
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

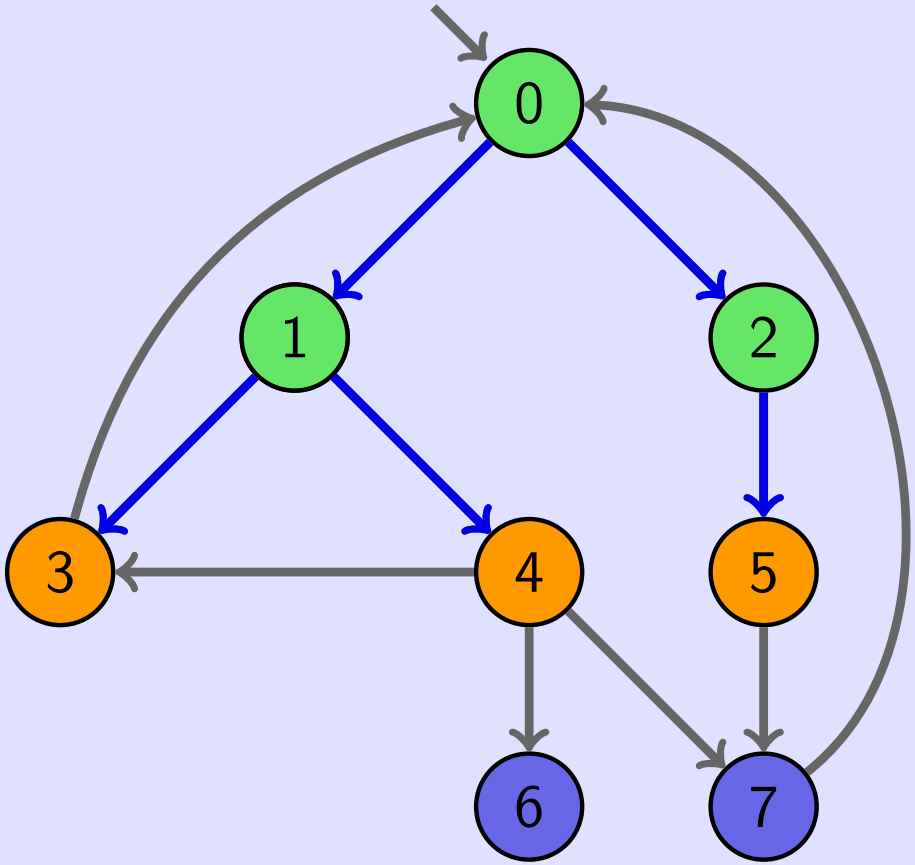
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA
Example



open state

closed state

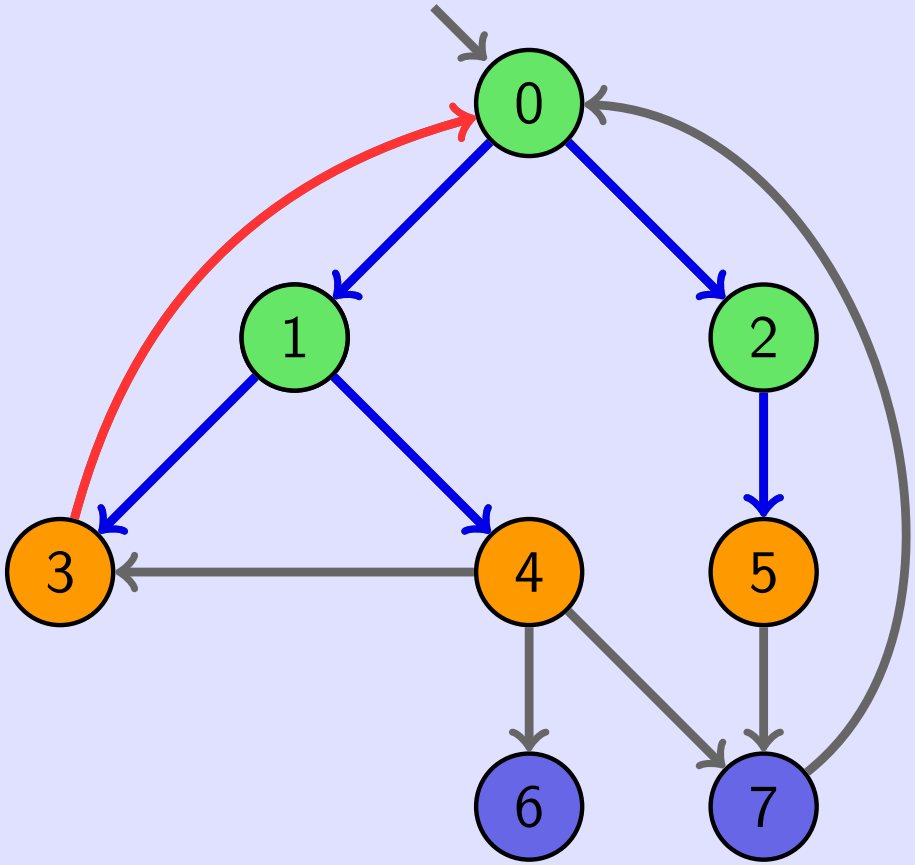
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

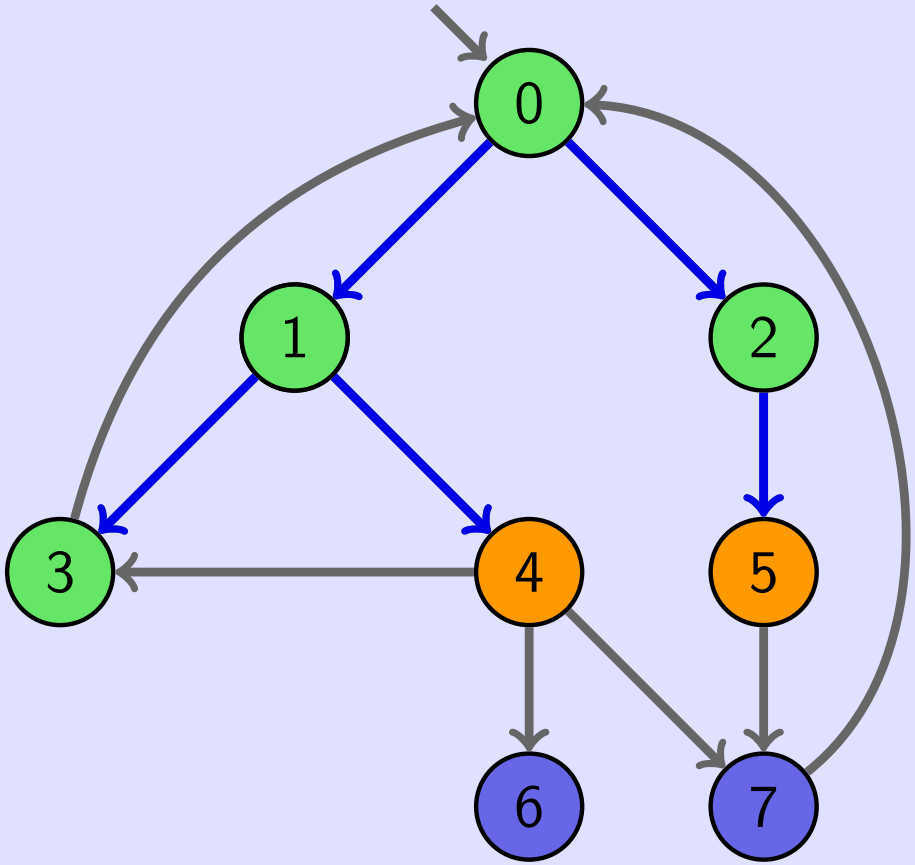
closed state

unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

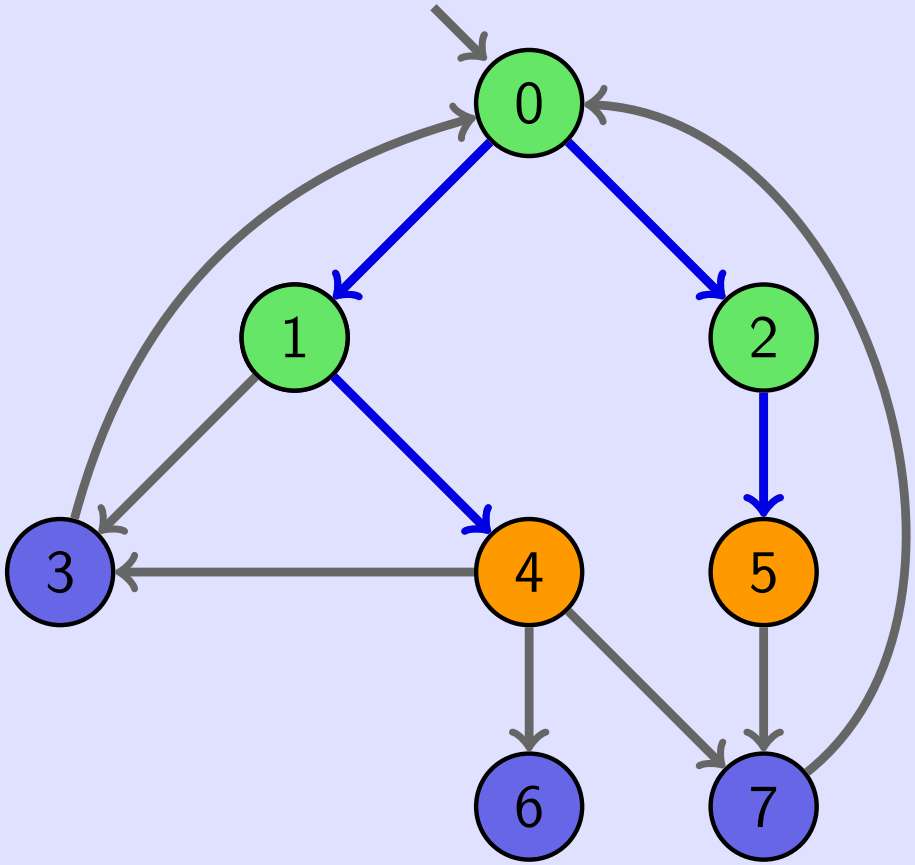
State caching for GSEA
Example



open state
closed state
unmet state
$\longrightarrow$ transition executed
$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

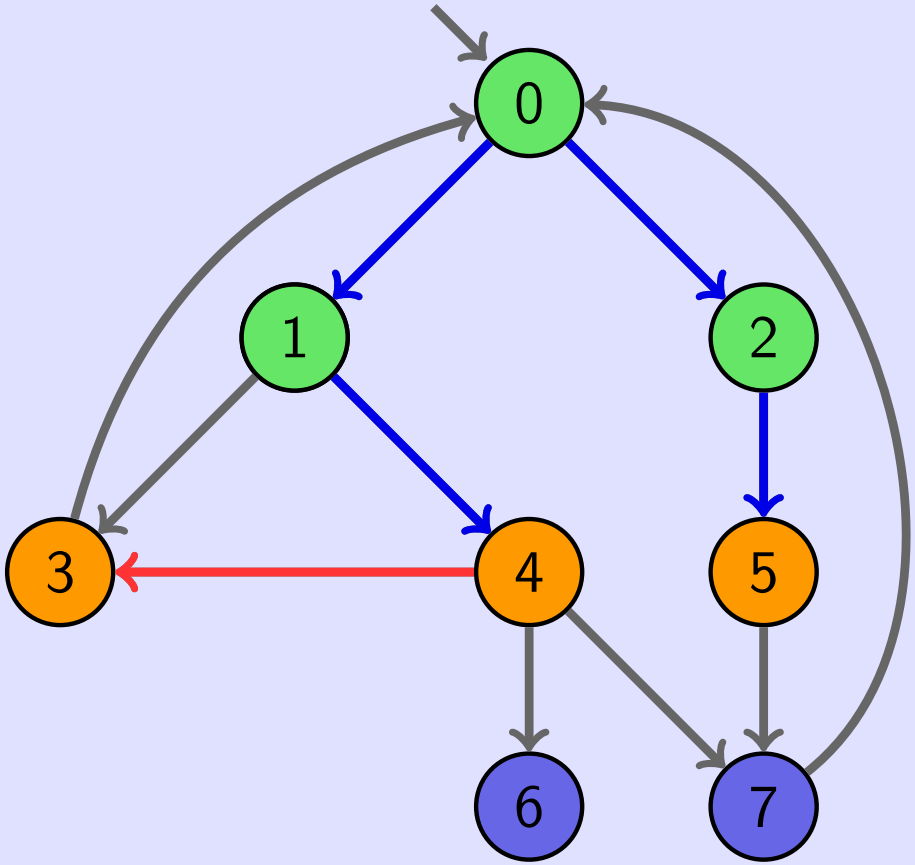
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

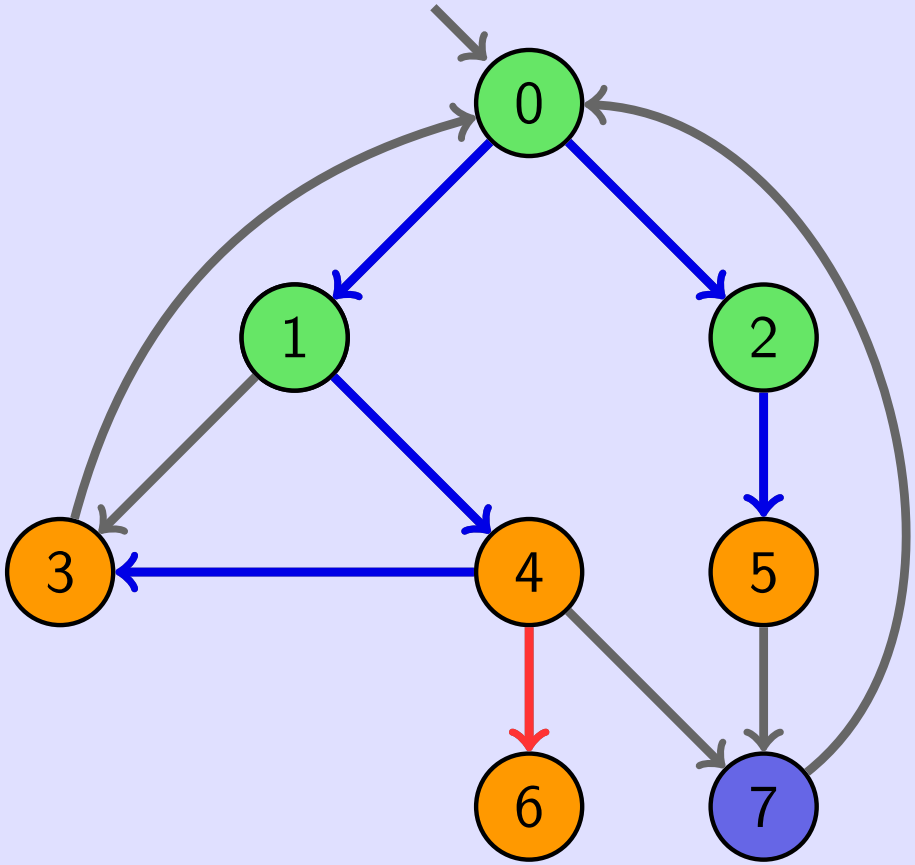
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

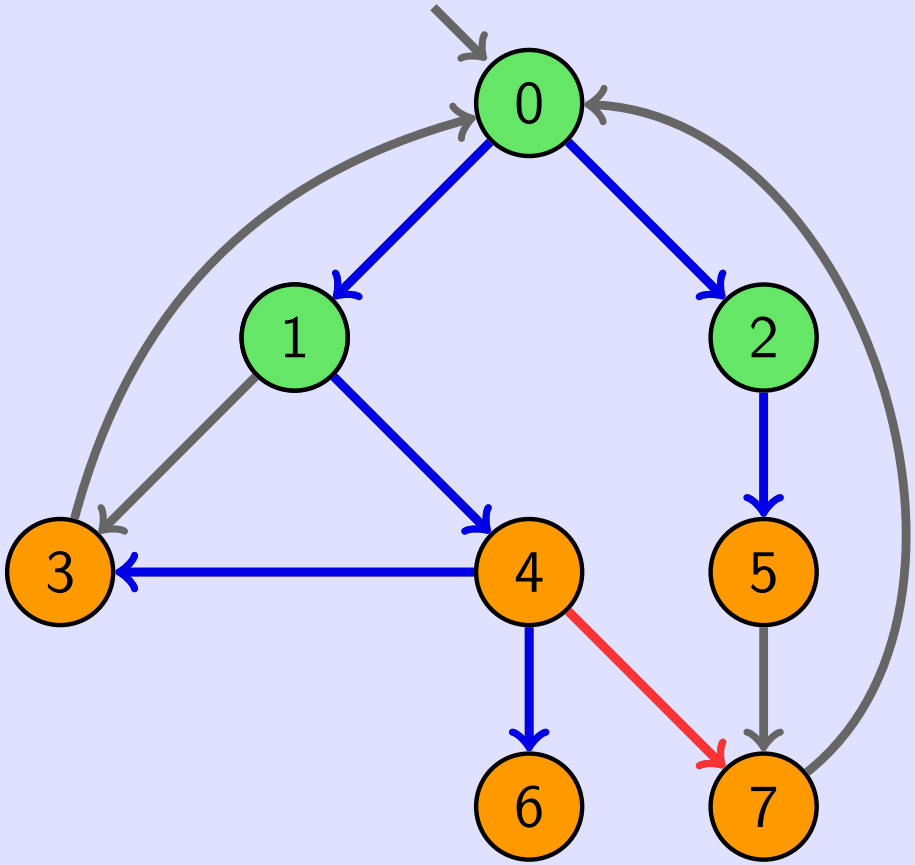
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

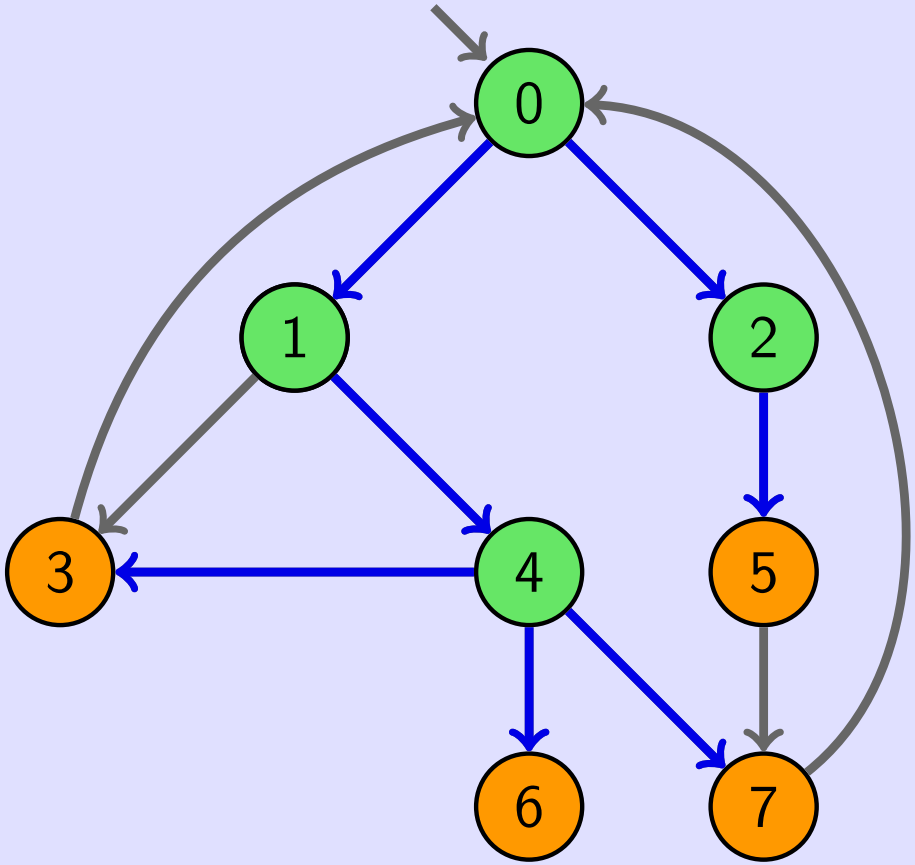
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

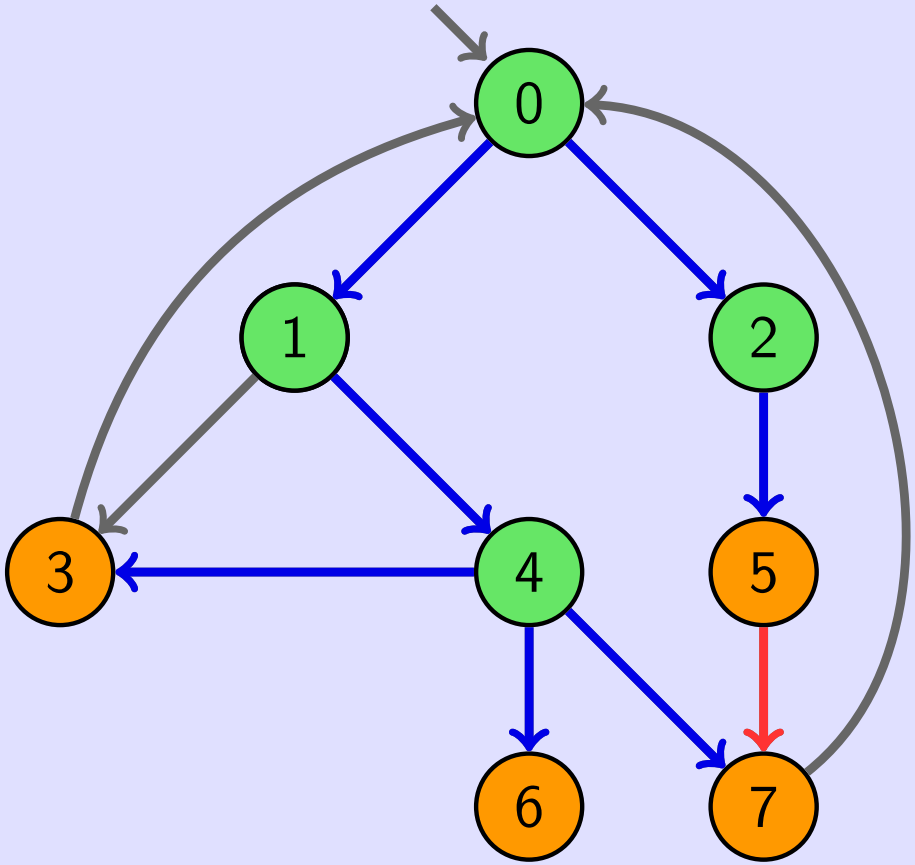
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

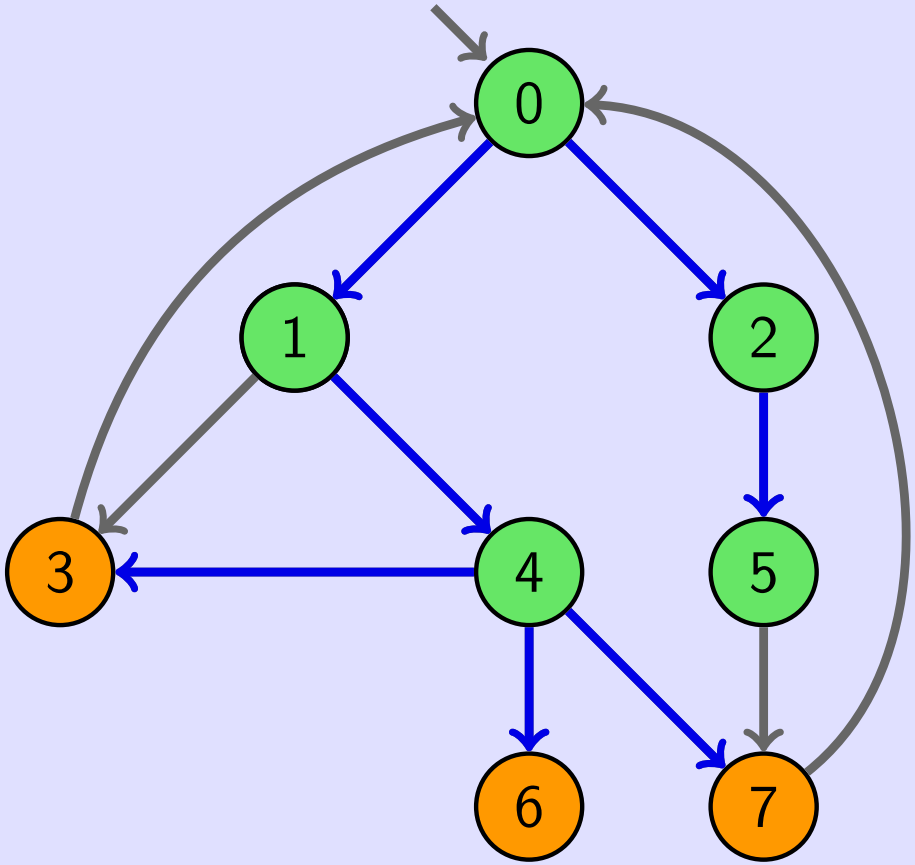
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

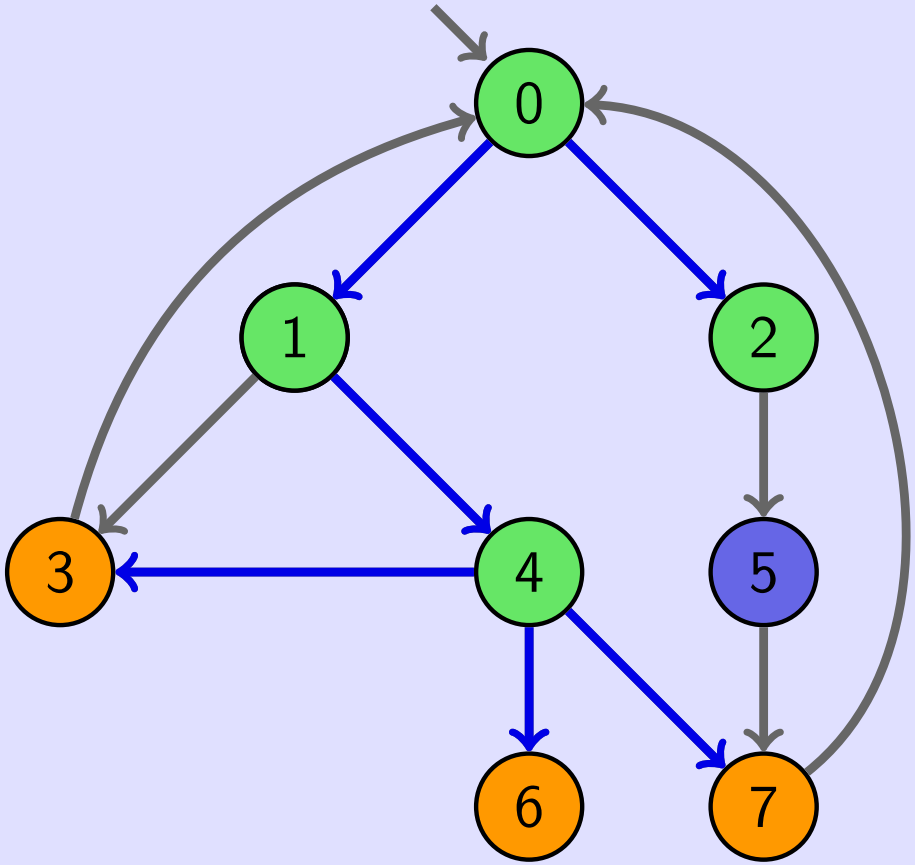
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

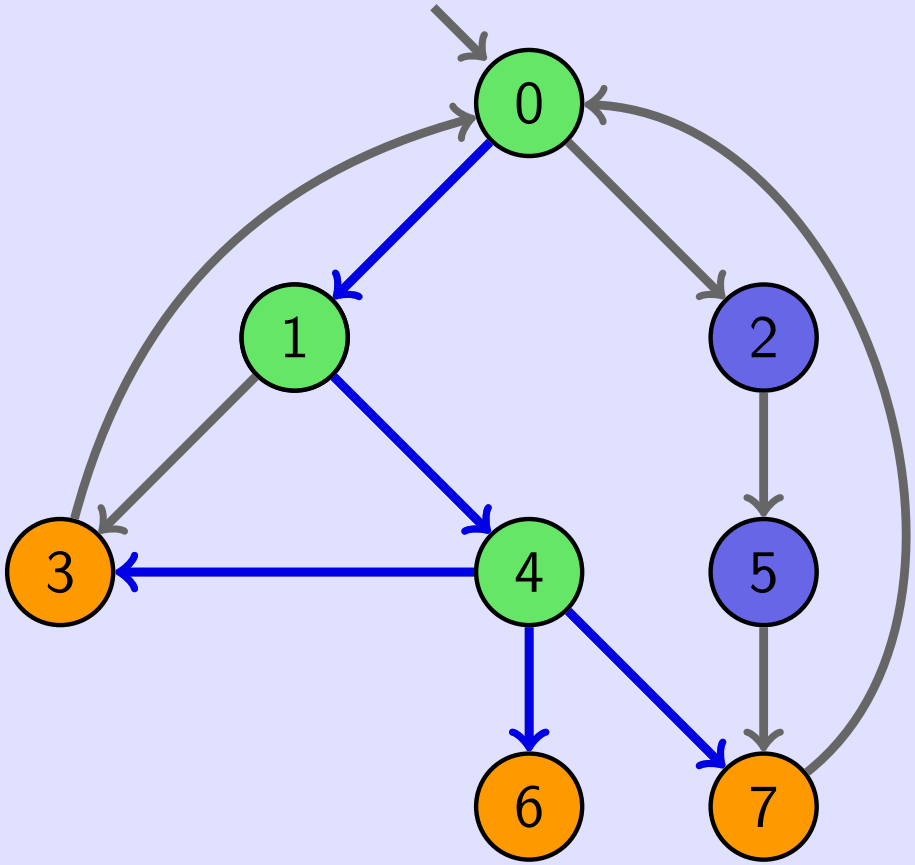
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

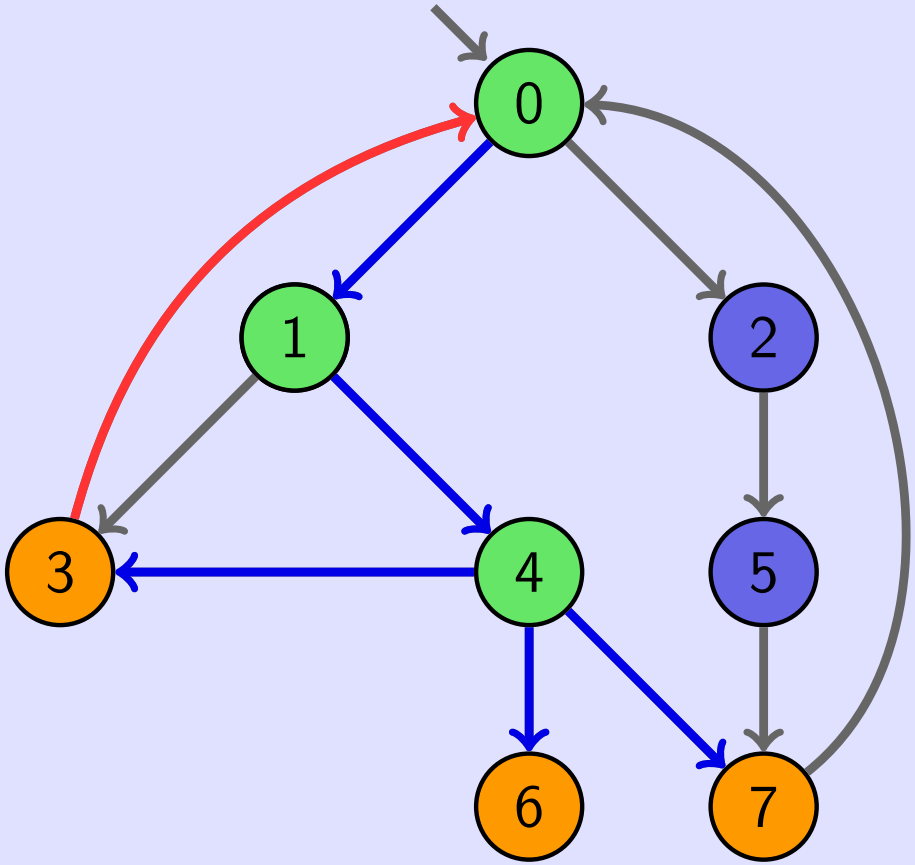
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

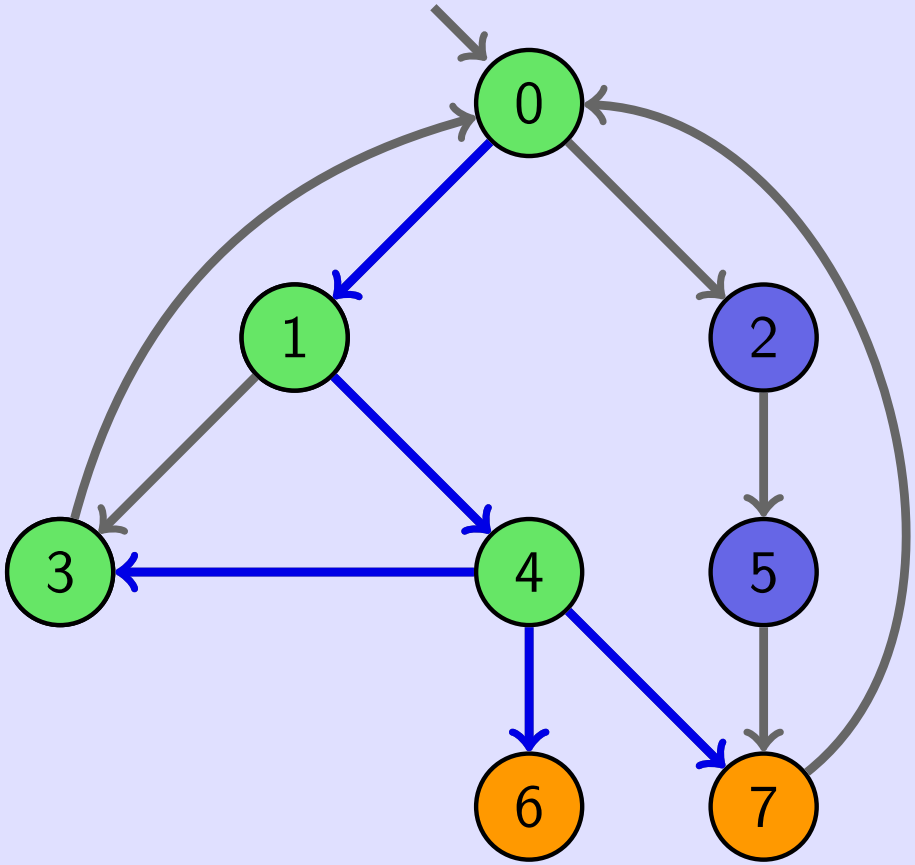
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

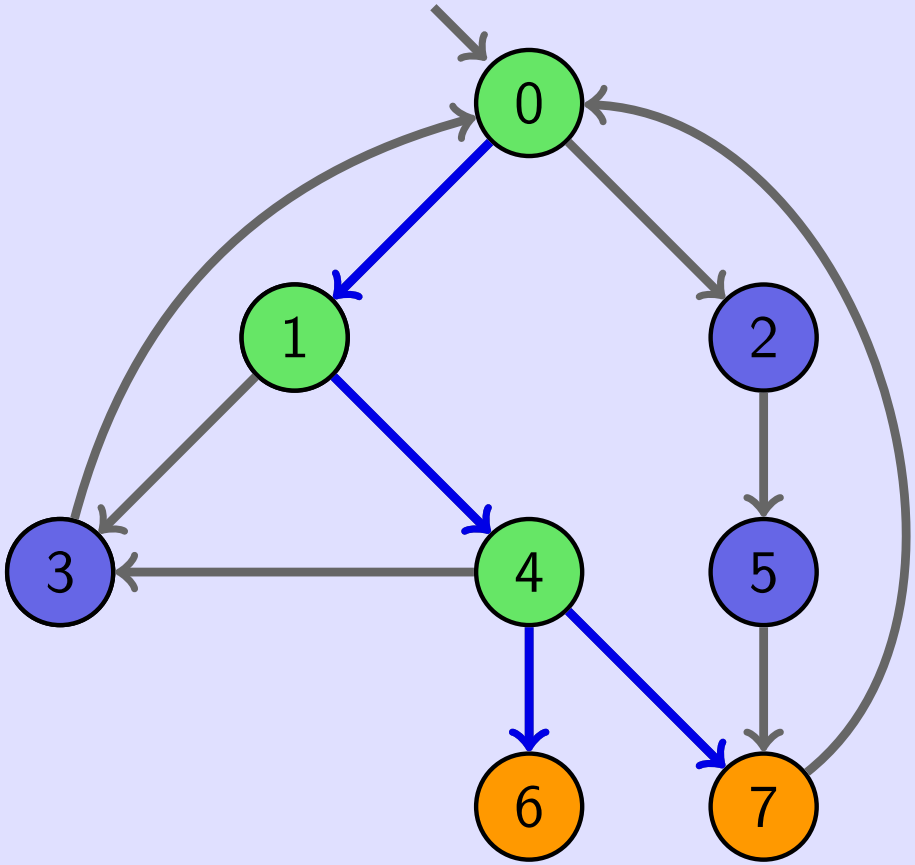
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

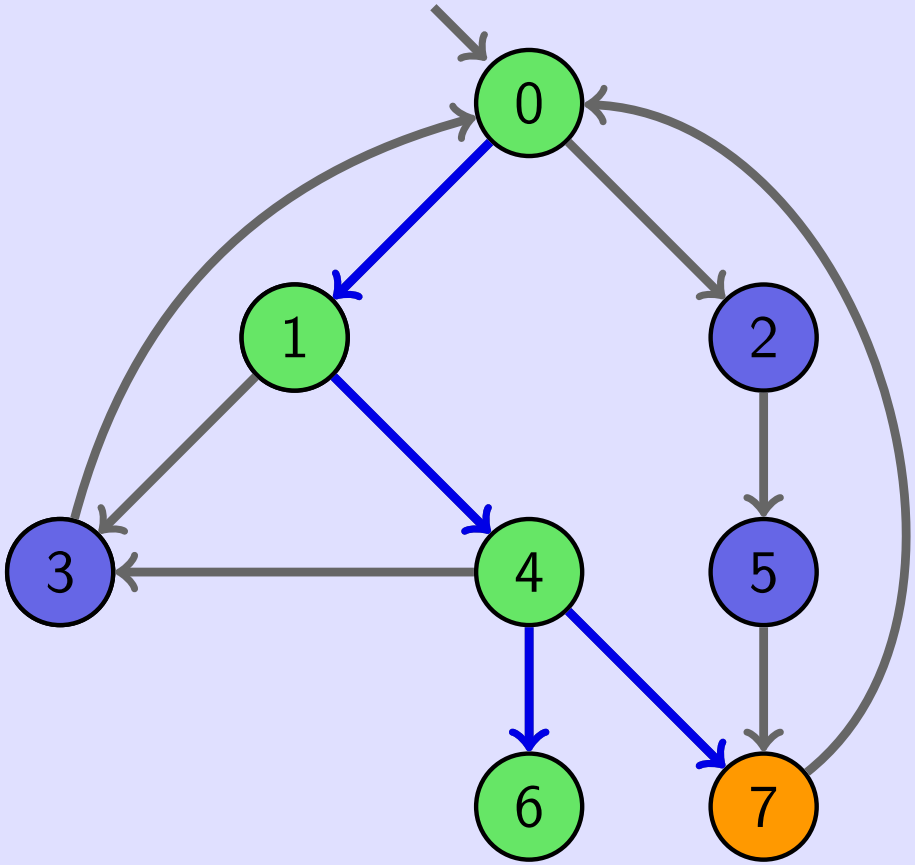
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

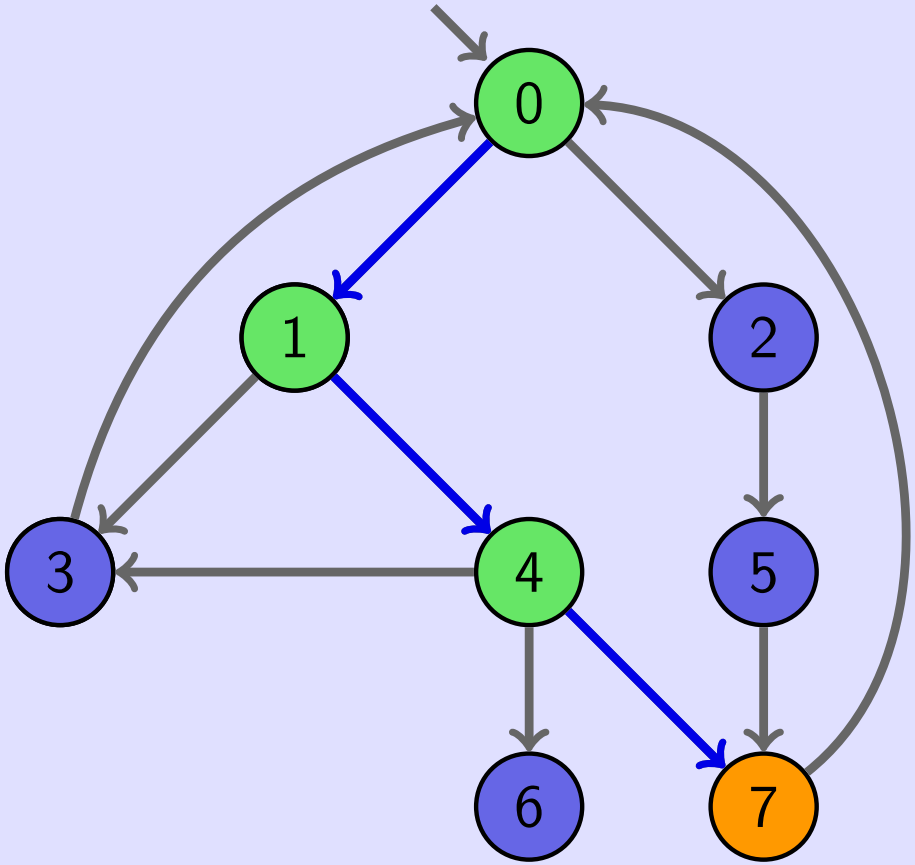
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

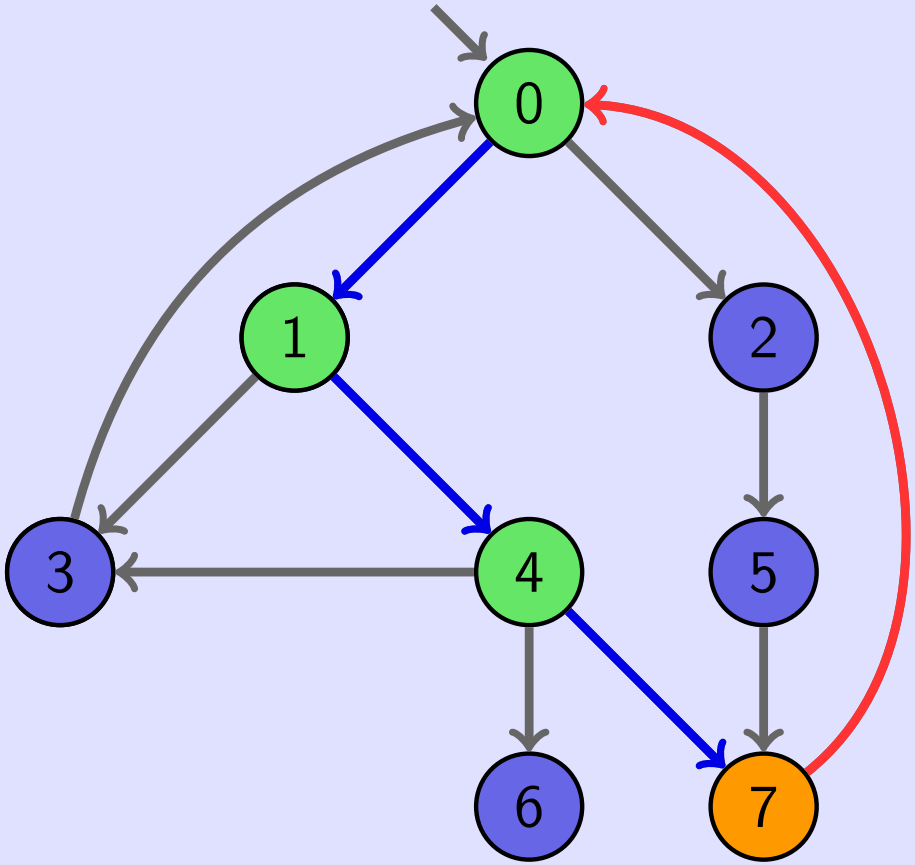
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

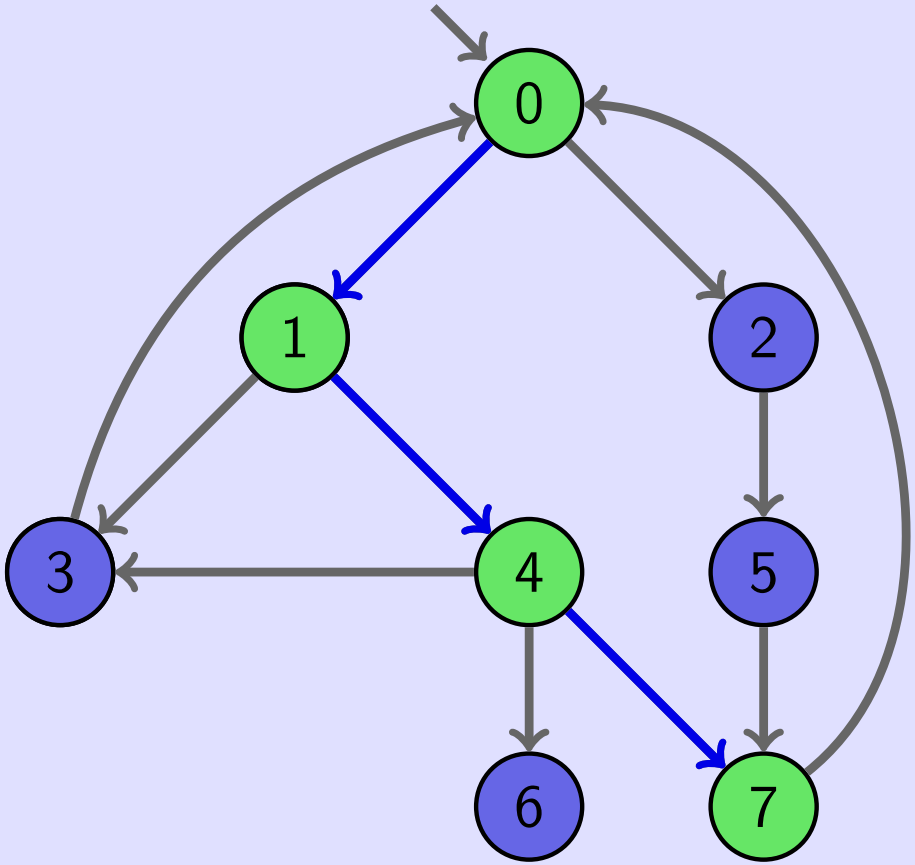
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

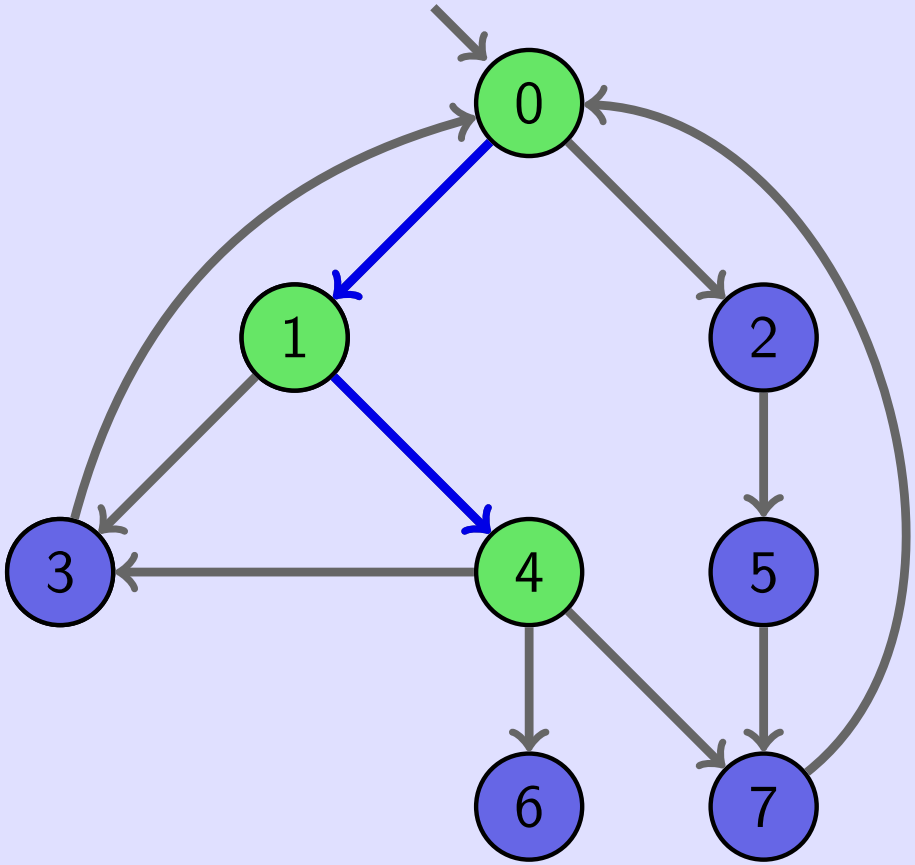
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

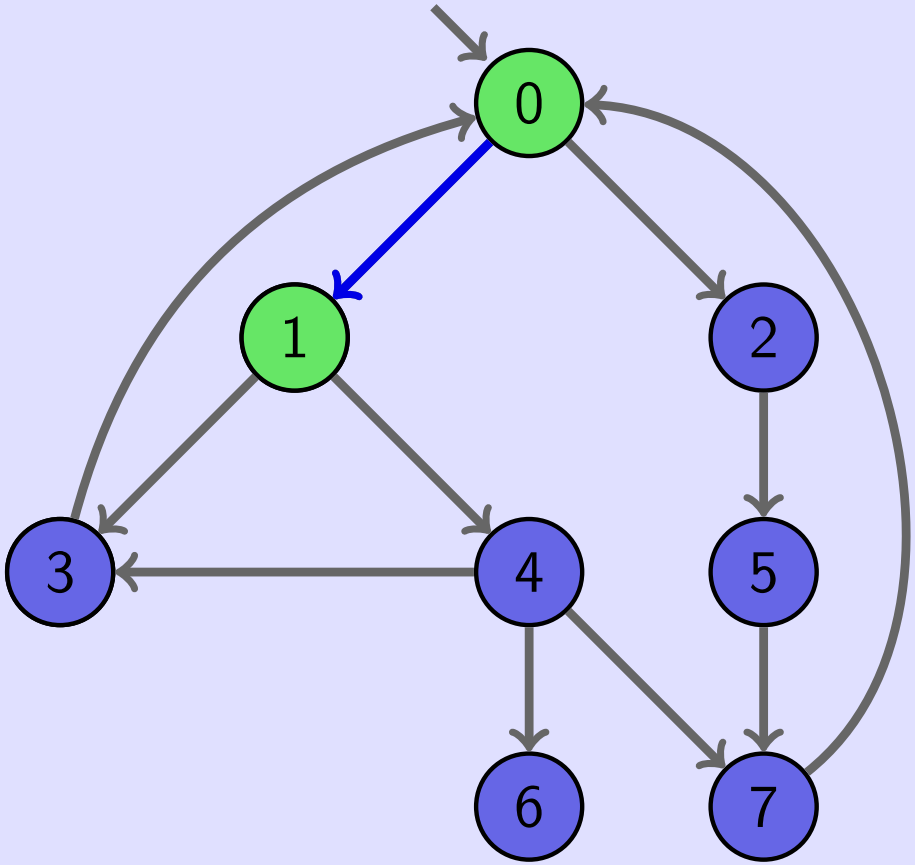
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

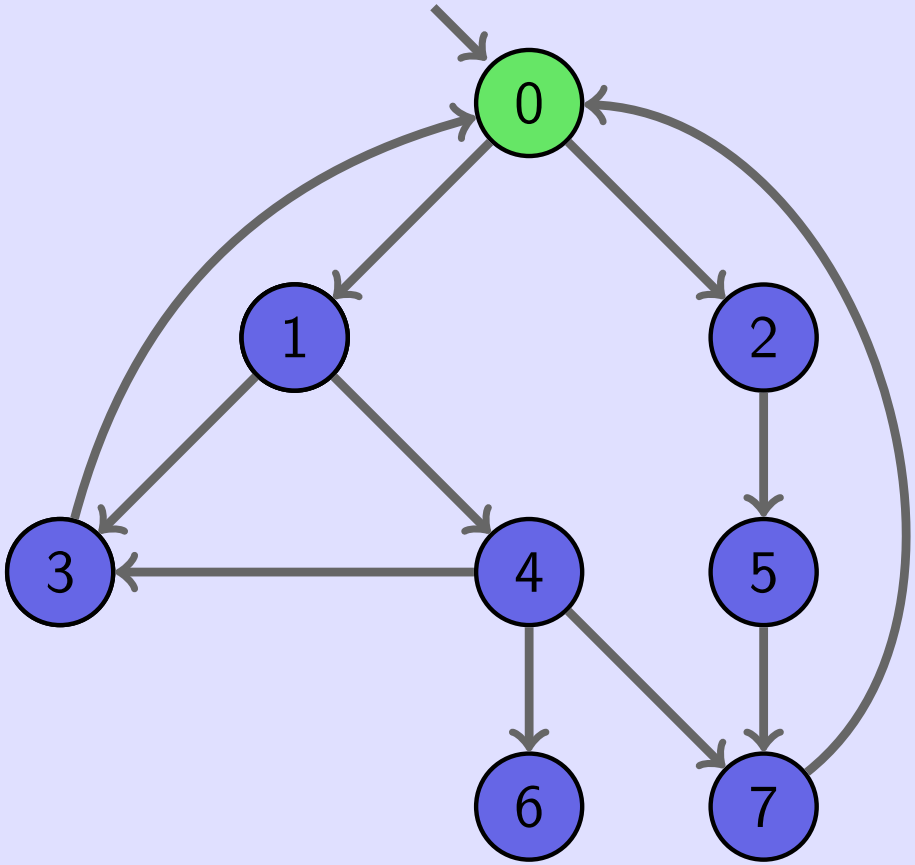
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

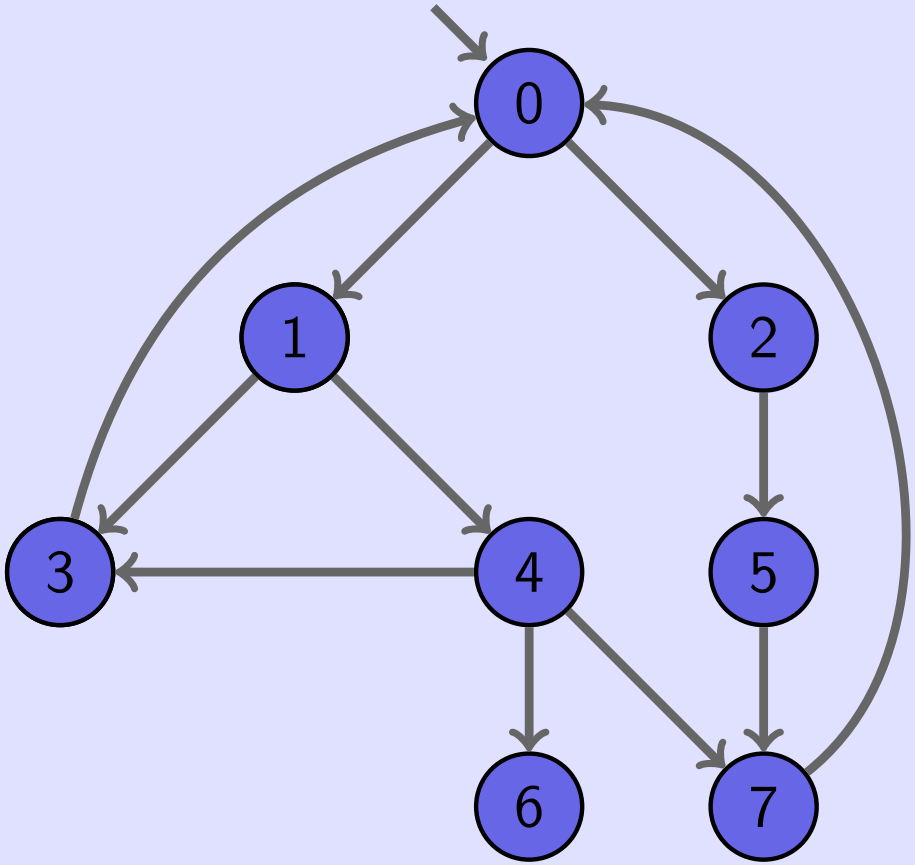
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Example



open state

closed state

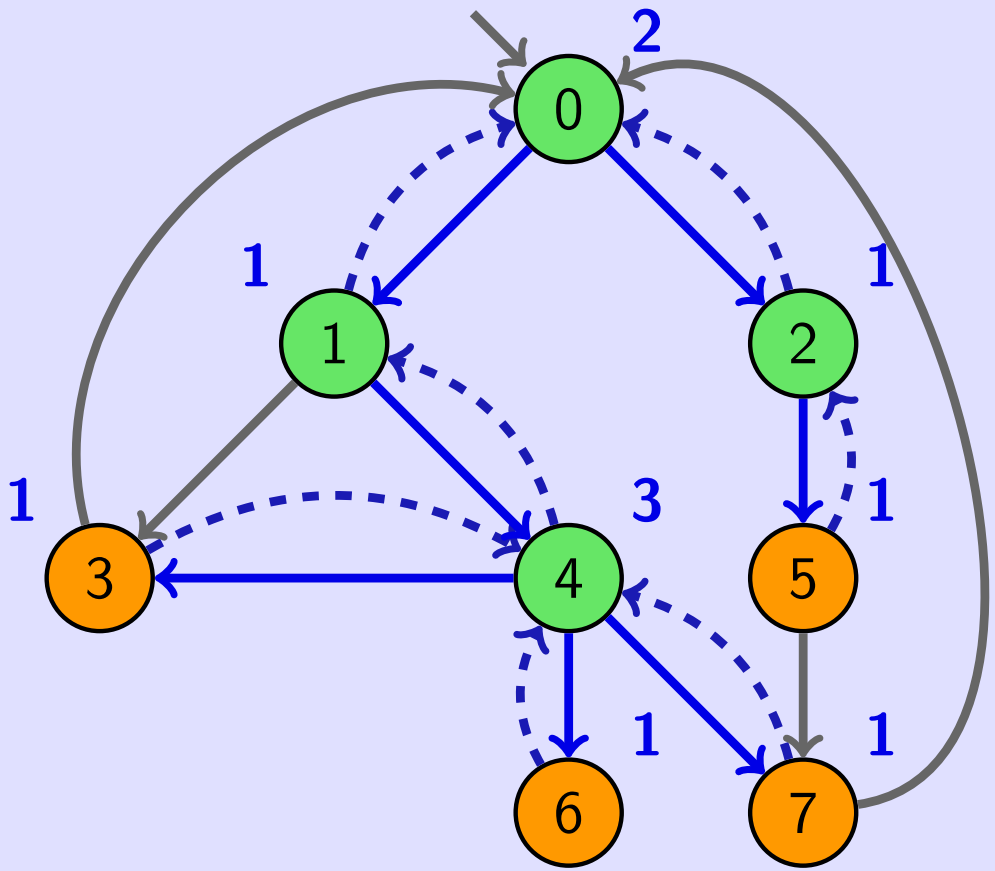
unmet state

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Maintaining the TD-tree



open states

closed states

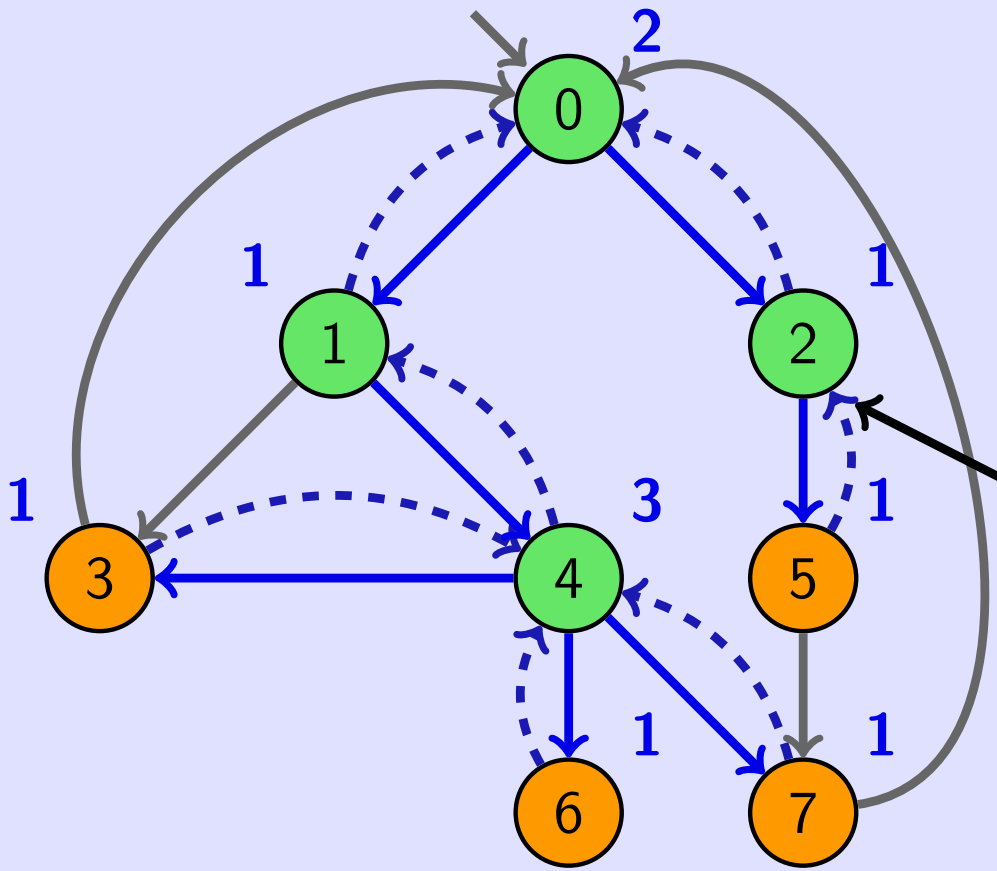
unmet states

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Maintaining the TD-tree



open states

closed states

unmet states

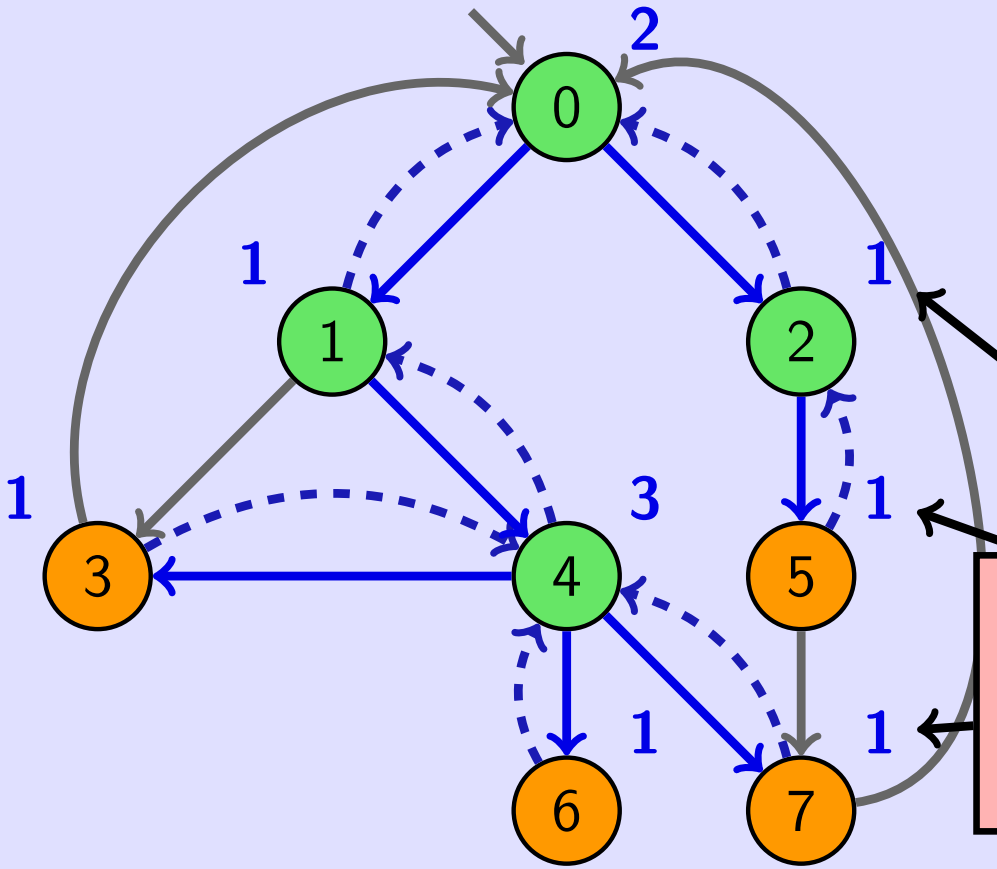
$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

Pointer to the father in the TD-tree (4 bytes)

State caching for GSEA

Maintaining the TD-tree



open states

closed states

unmet states

$\longrightarrow$ transition executed

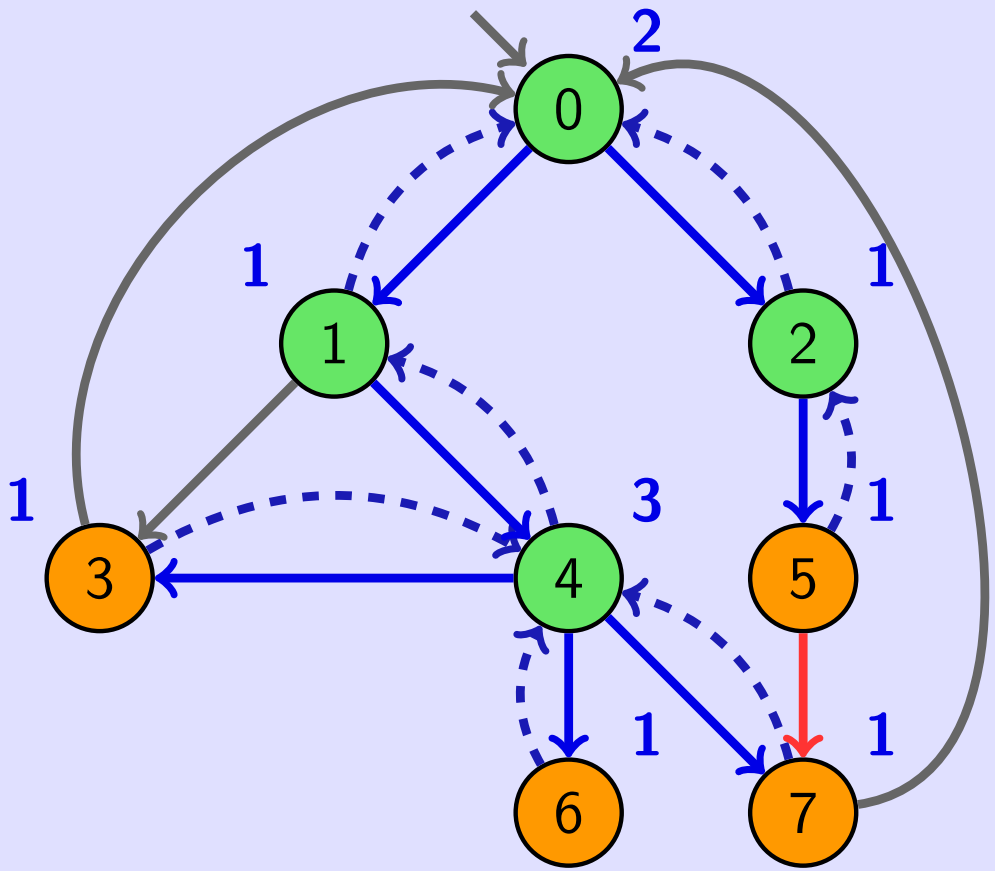
$\longrightarrow$ transition of the TD-tree

Ref. counter equals to

- # of children in the TD-tree
- +1 if it is an open state

State caching for GSEA

Maintaining the TD-tree



open states

closed states

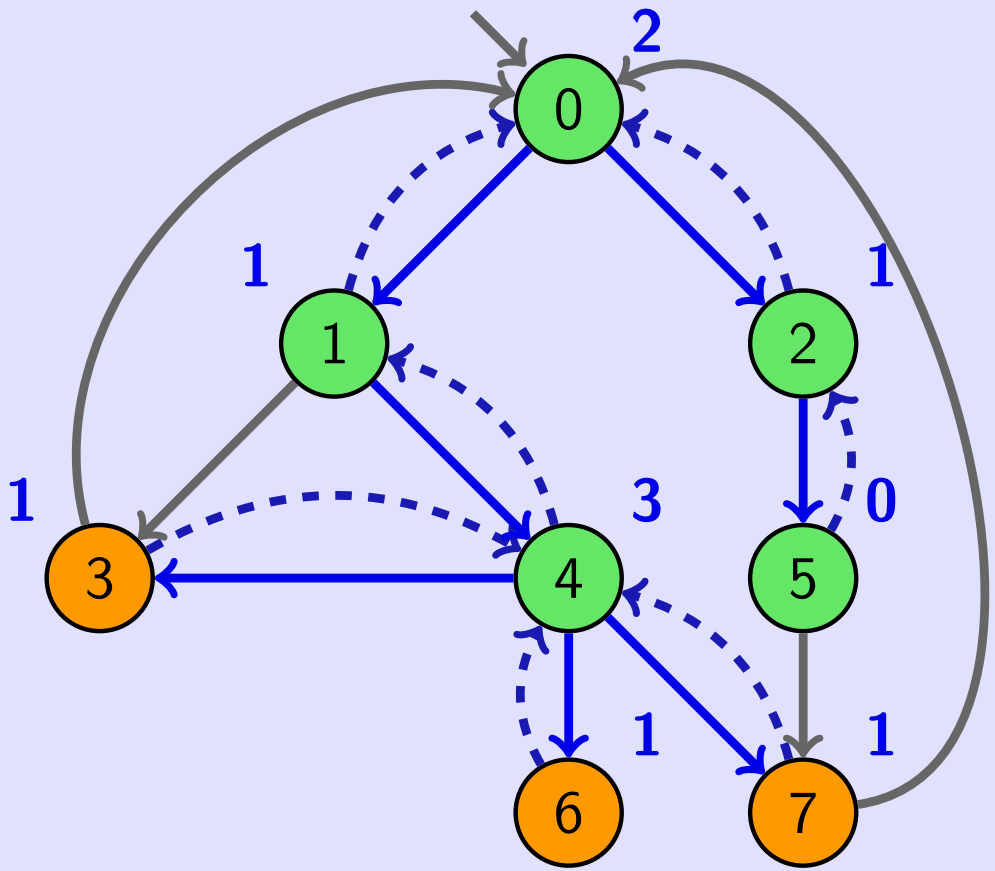
unmet states

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Maintaining the TD-tree



open states

closed states

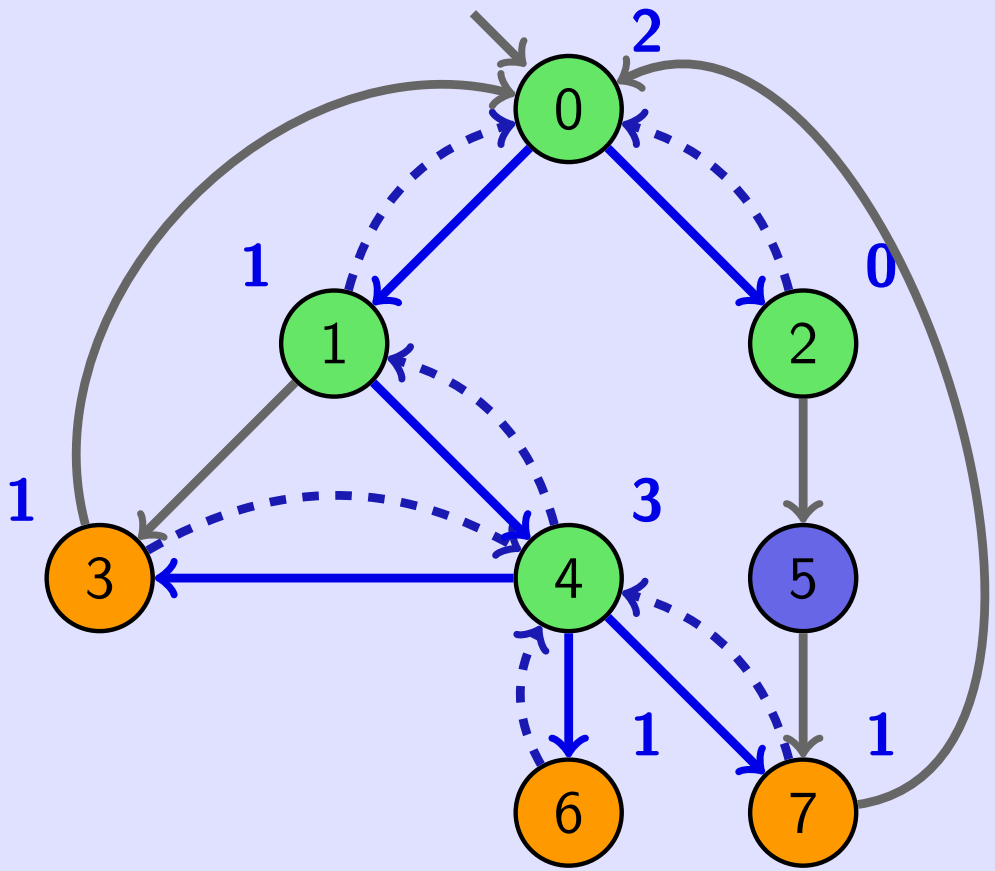
unmet states

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Maintaining the TD-tree



open states

closed states

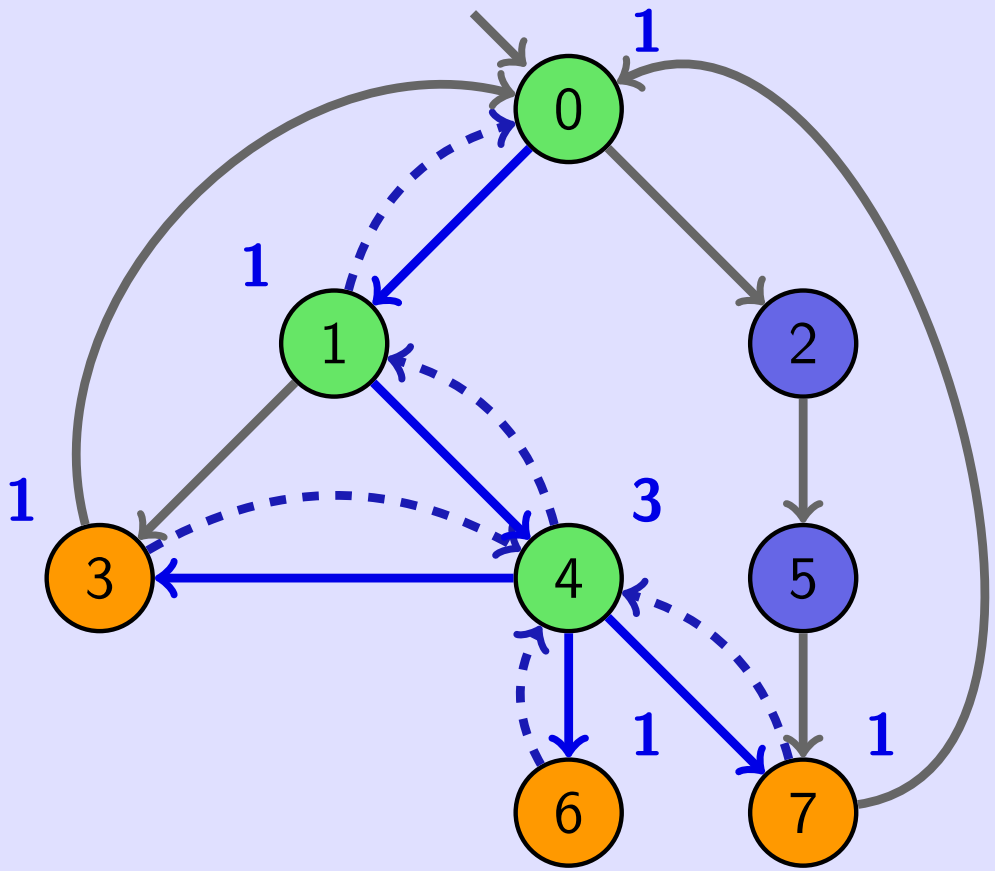
unmet states

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

State caching for GSEA

Maintaining the TD-tree



open states

closed states

unmet states

$\longrightarrow$ transition executed

$\longrightarrow$ transition of the TD-tree

Experiments
State caching + DFS and BFS

- algorithm implemented in ASAP
- input models: 135 DVE instances from the BEEM[1] database
- algorithms used: DFS, BFS (and combination of both, cf. paper)
- 60 different cache replacement strategies (selection of states based on in-degree, out-degree, distance from the initial state, . . .)
- a run can end in three different states:
  - success
  - out of memory: the cache is too small to store the TD-tree
  - out of time: explored more than $5 \cdot |S|$ ($|S|$ = state space size)
- cache size: $\{ 5\% \cdot |S|, 10\% \cdot |S|, 15\% \cdot |S|, \ldots \}$ until a successful run could be found with at least one cache replacement strategy
- used partial order reduction to avoid useless interleavings
- more than 1,000,000 runs performed

[1] `http://anna.fi.muni.cz/models/`

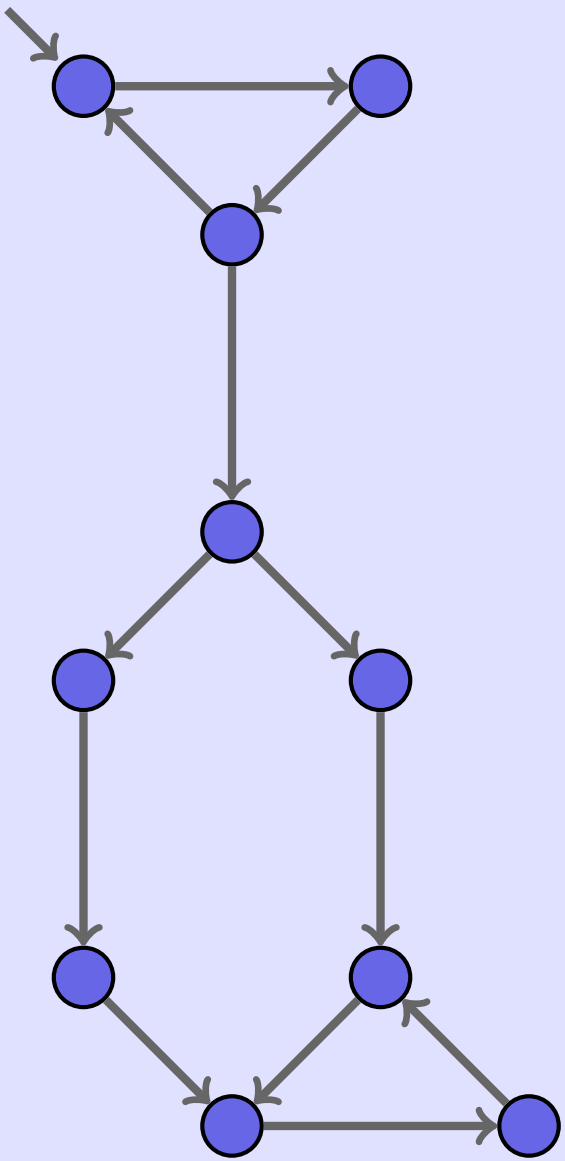
Experiments

State caching + DFS and BFS

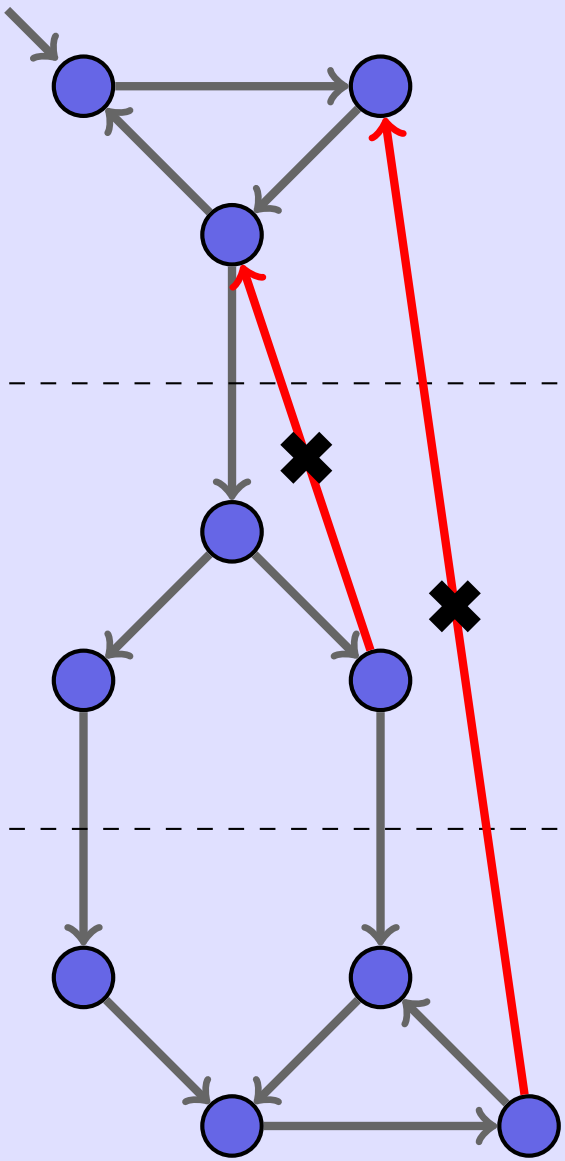
Model	States	DFS		BFS	
		S	V	S	V
brp.3	996,627	5%	235%	15%	112%
bopdp.2	25,685	15%	215%	30%	149%
extinction.3	751,930	10%	185%	10%	100%
gear.2	16,689	15%	106%	5%	102%
firewire_link.7	399,598	5%	327%	25%	101%
firewire_tree.4	169,992	10%	337%	10%	100%
needham.3	206,925	5%	412%	30%	100%
plc.2	130,777	5%	100%	30%	100%
rether.3	305,334	25%	152%	15%	111%
synapse.6	625,175	5%	148%	30%	111%

- S: Stored states, i.e., cache size (as a % of the state space size)
- V: Visited states (as a % of the state space size)
- 5%: data for the best successful run (ascending C, then ascending V)

Sweep-line reduction



Sweep-line reduction



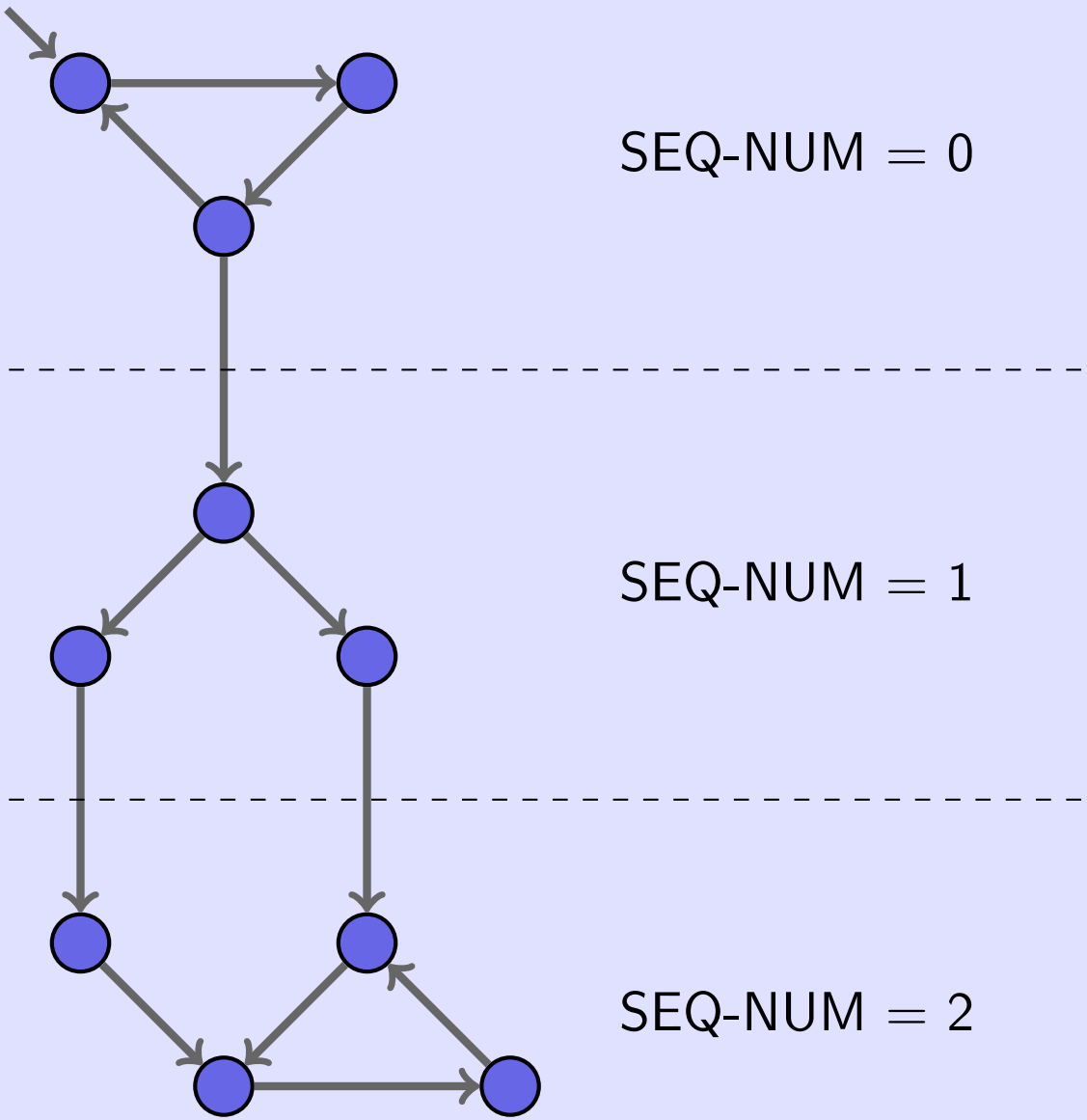
SEQ-NUM = 0

SEQ-NUM = 1

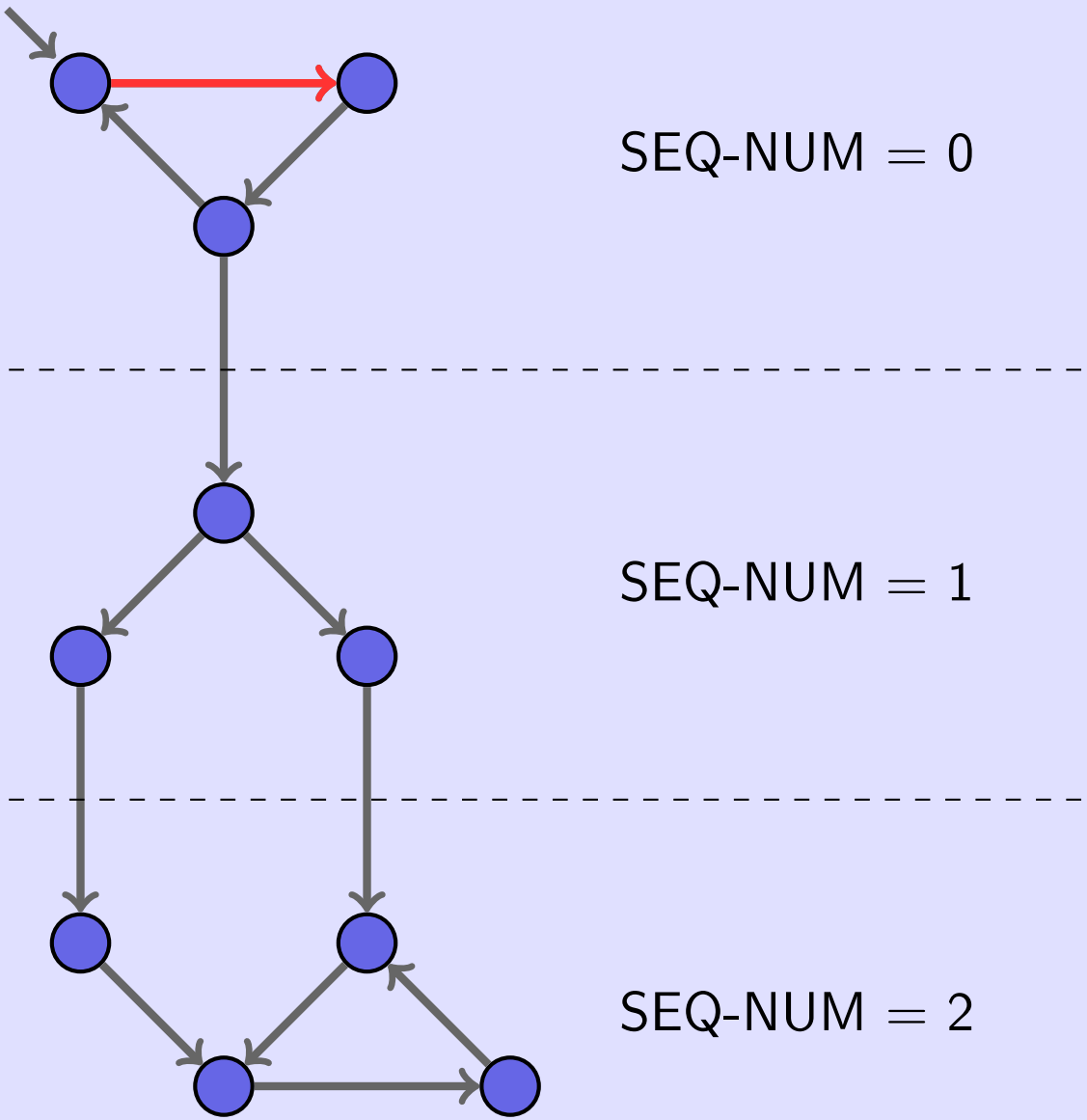
SEQ-NUM = 2

User defined
Progress Mapping
used to evaluate the
progression of states

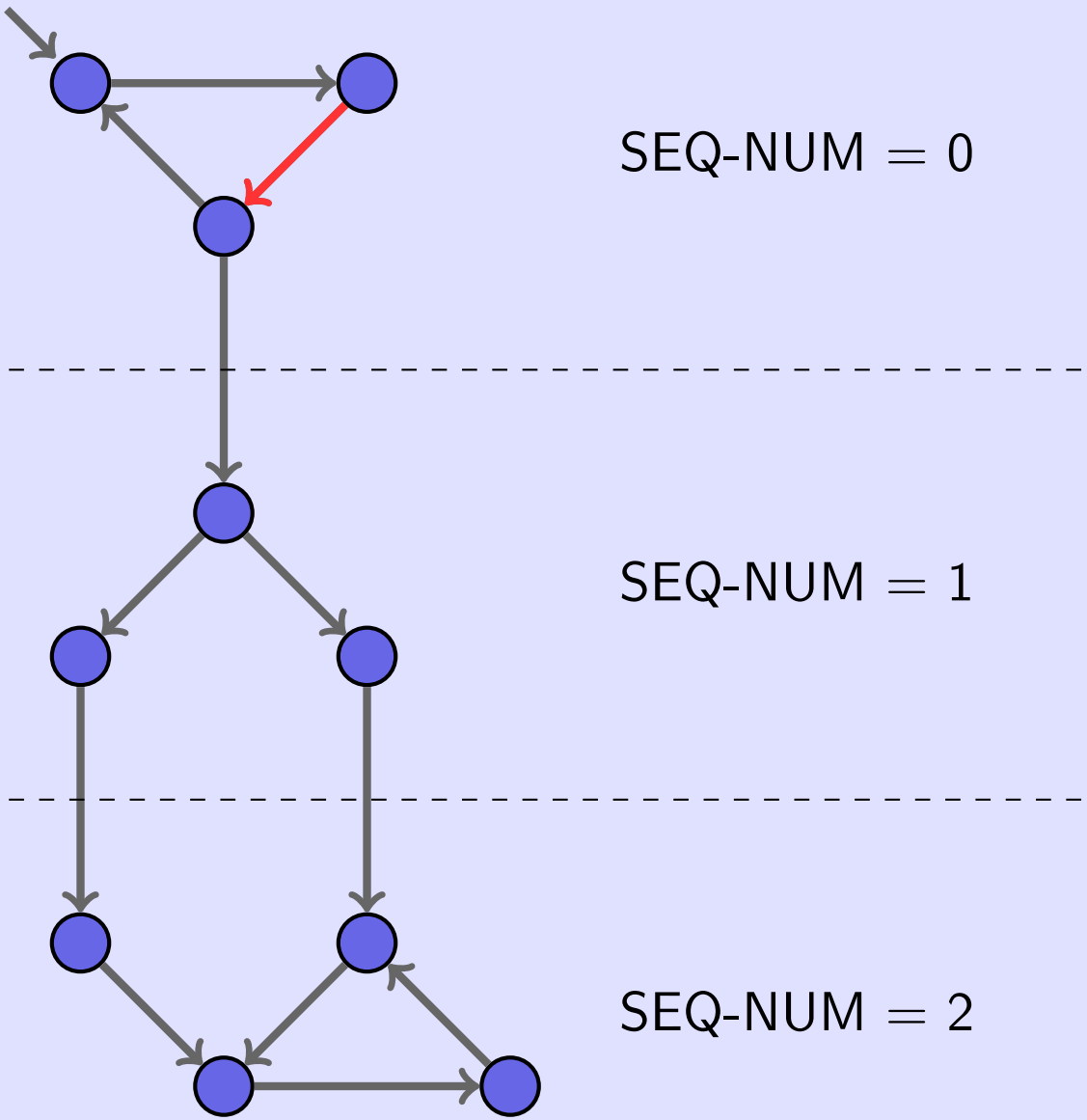
Sweep-line reduction



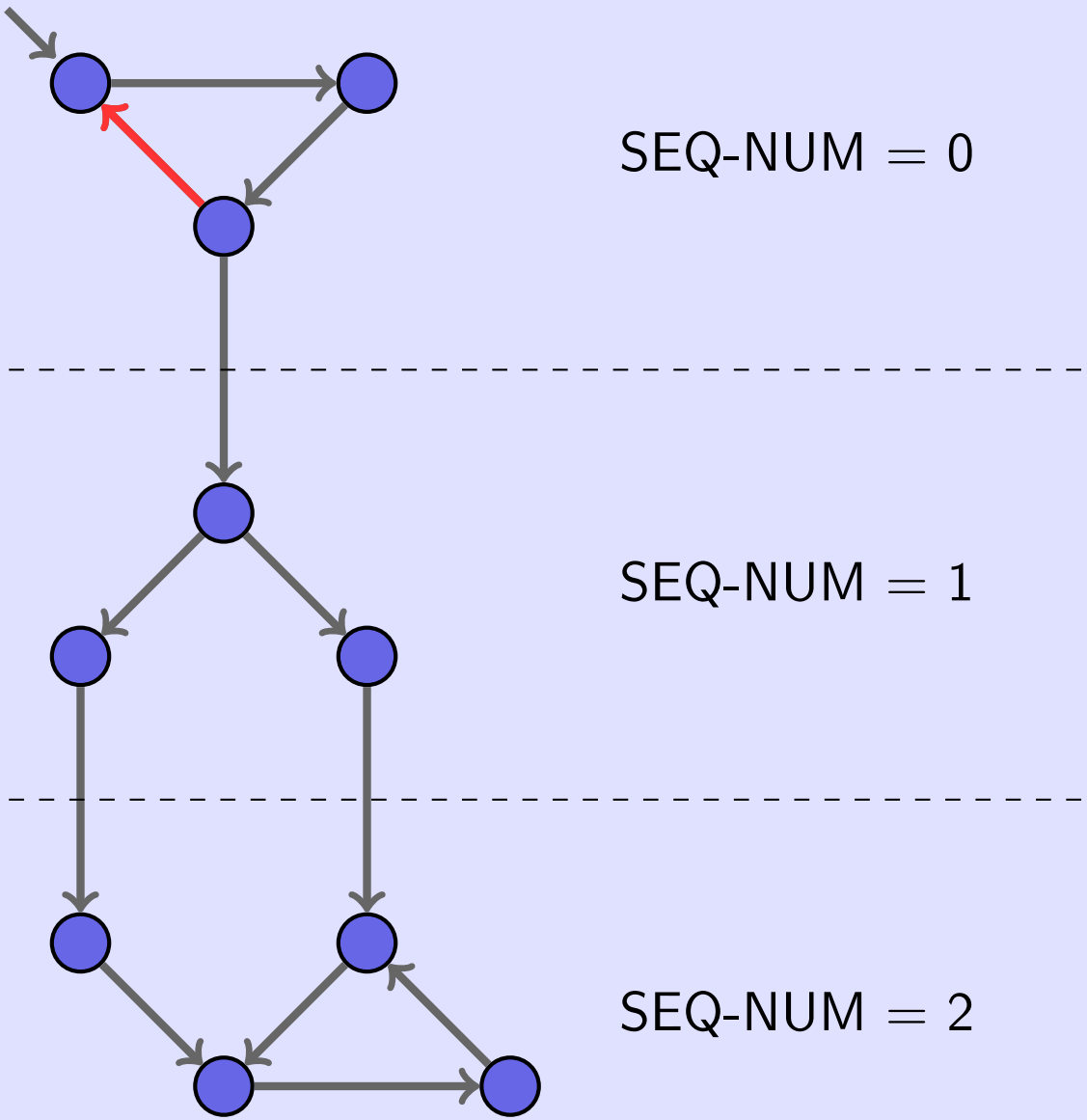
Sweep-line reduction



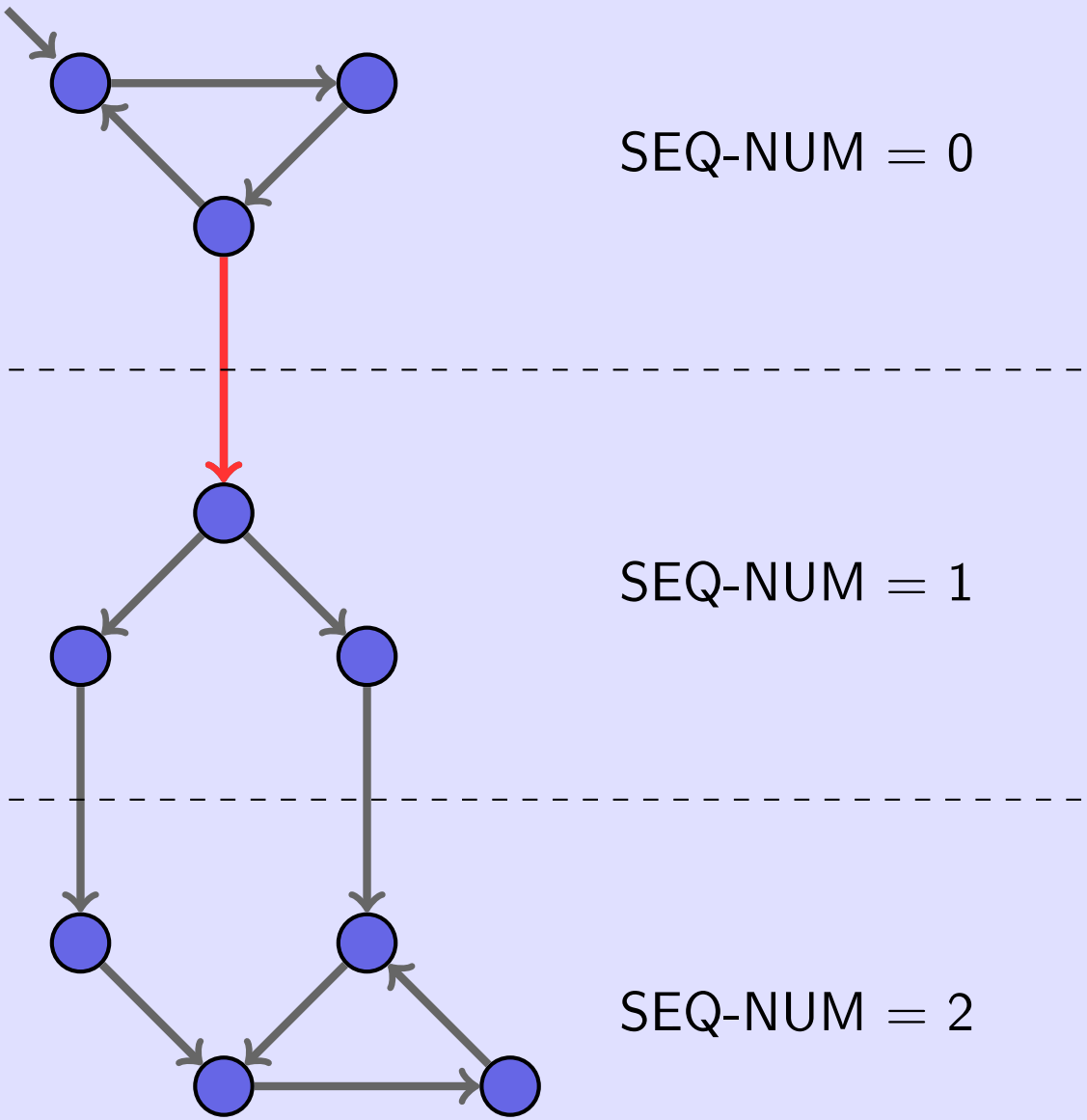
Sweep-line reduction



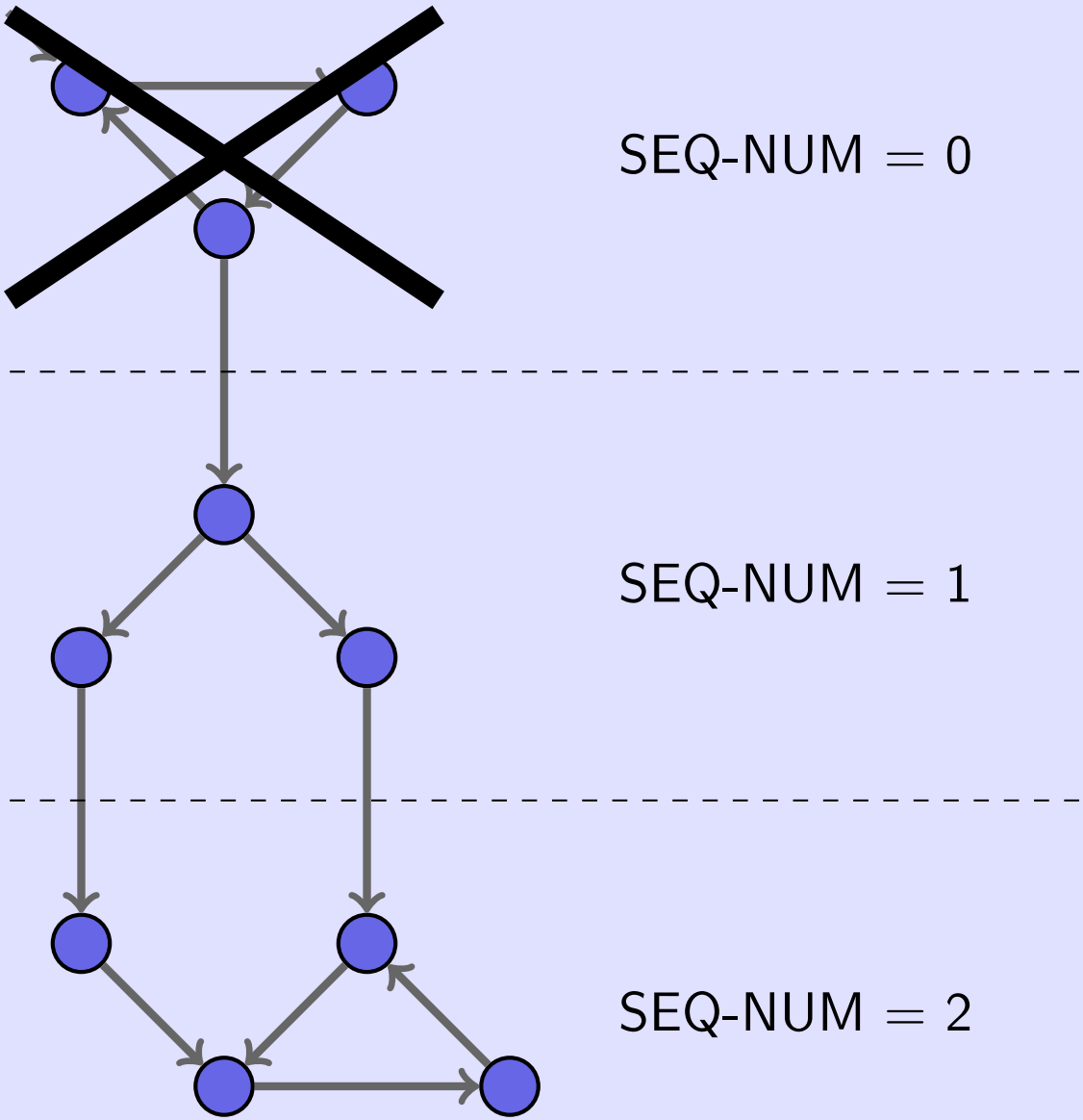
Sweep-line reduction



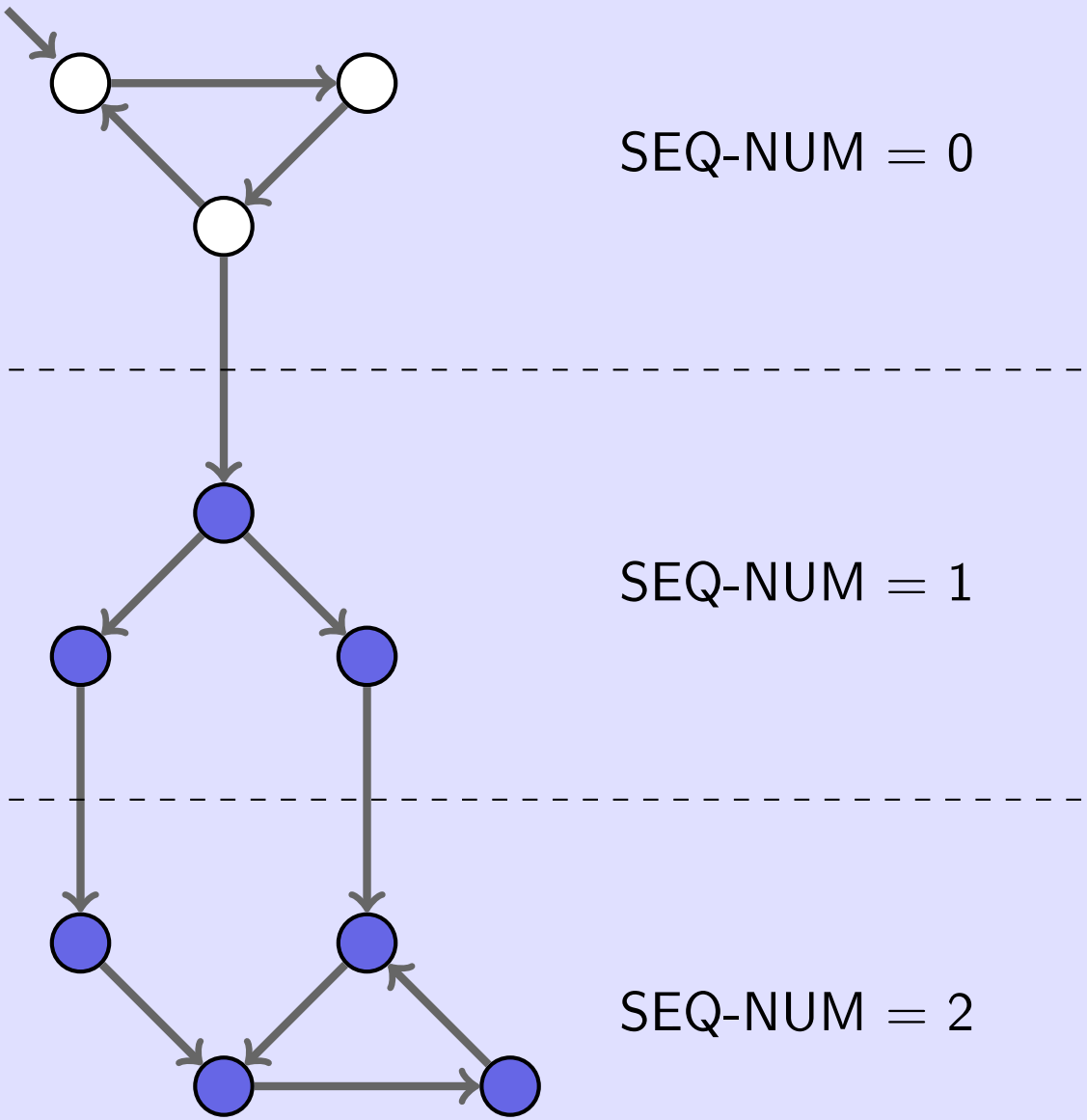
Sweep-line reduction



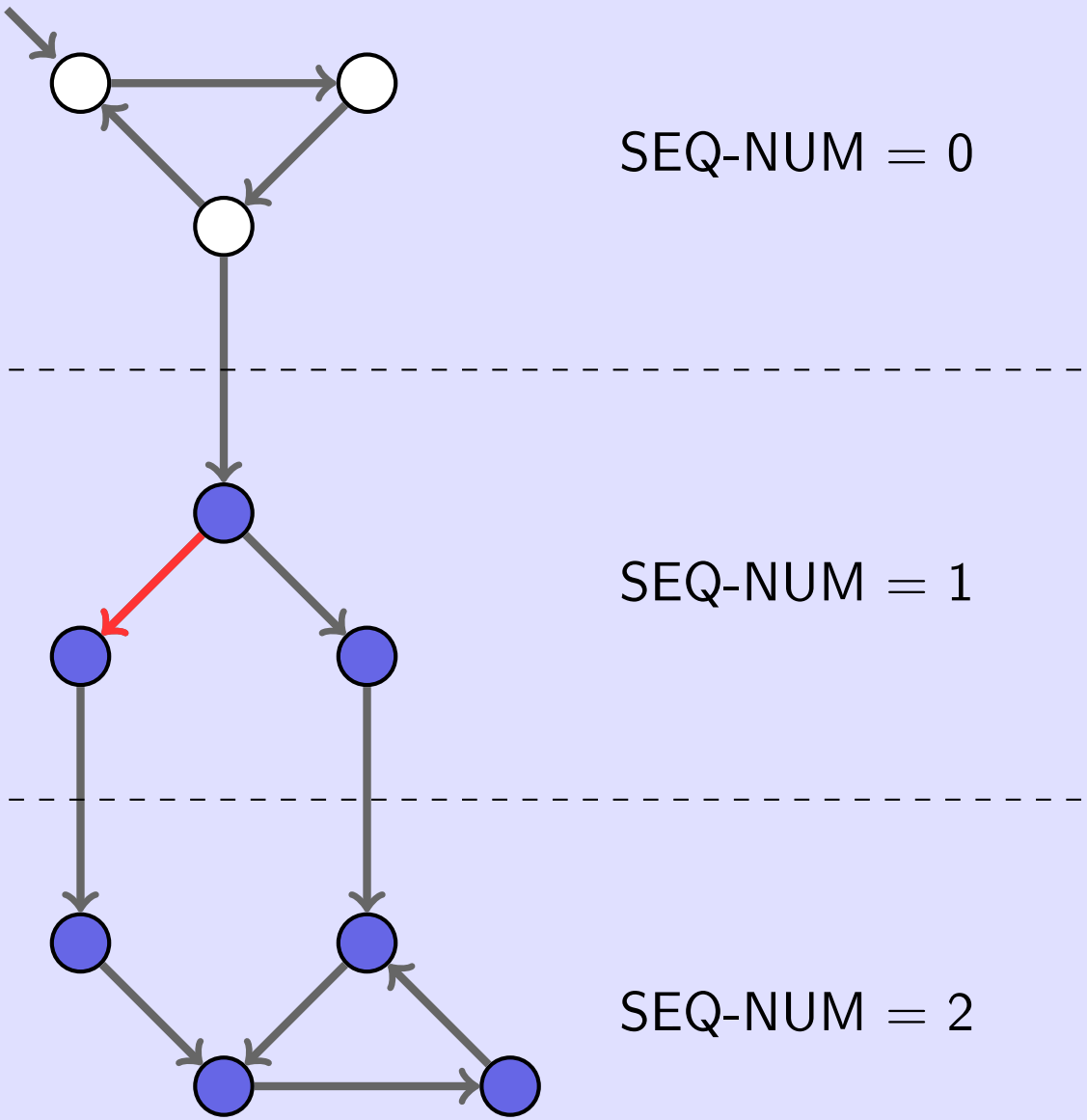
Sweep-line reduction



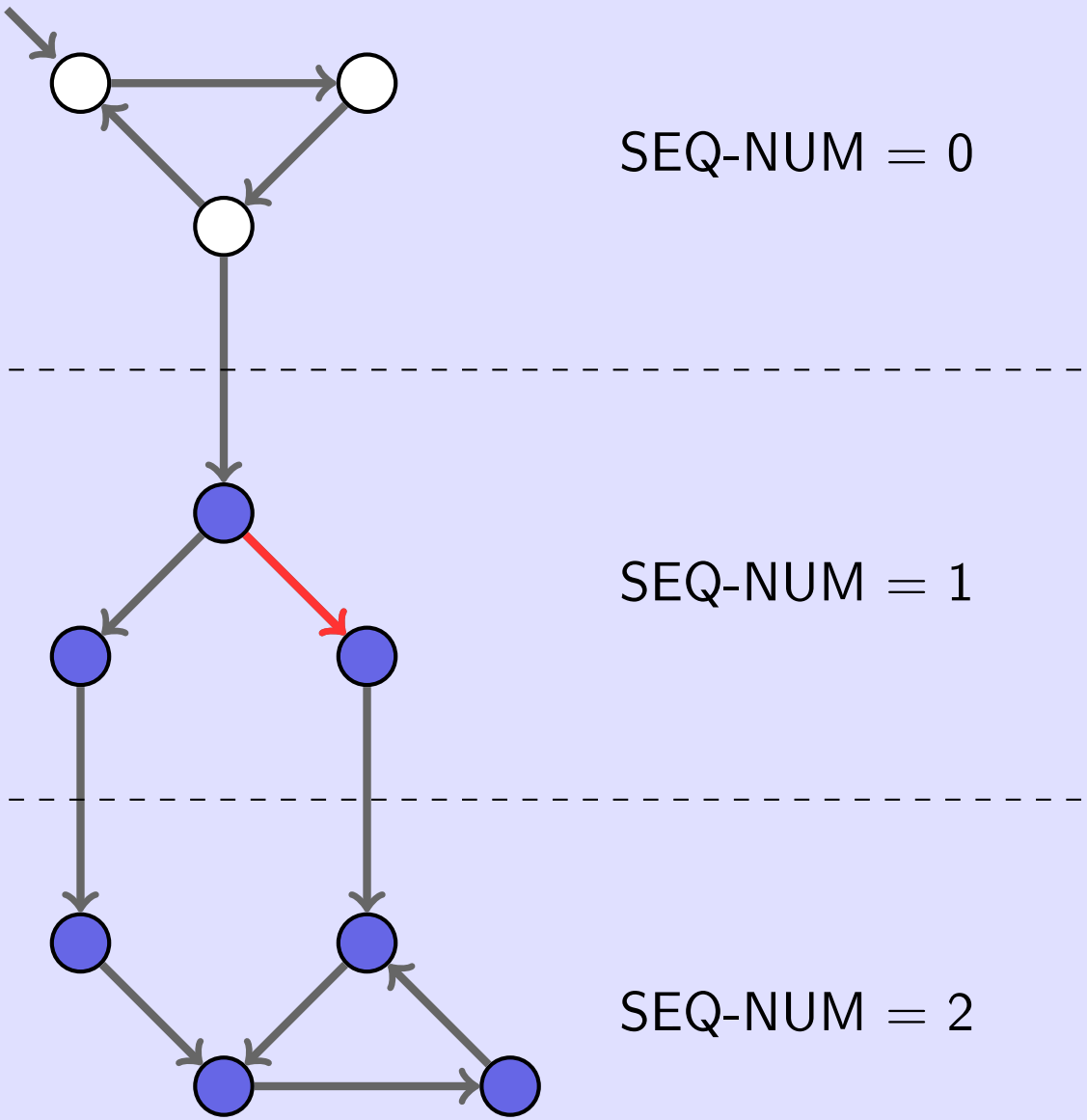
Sweep-line reduction



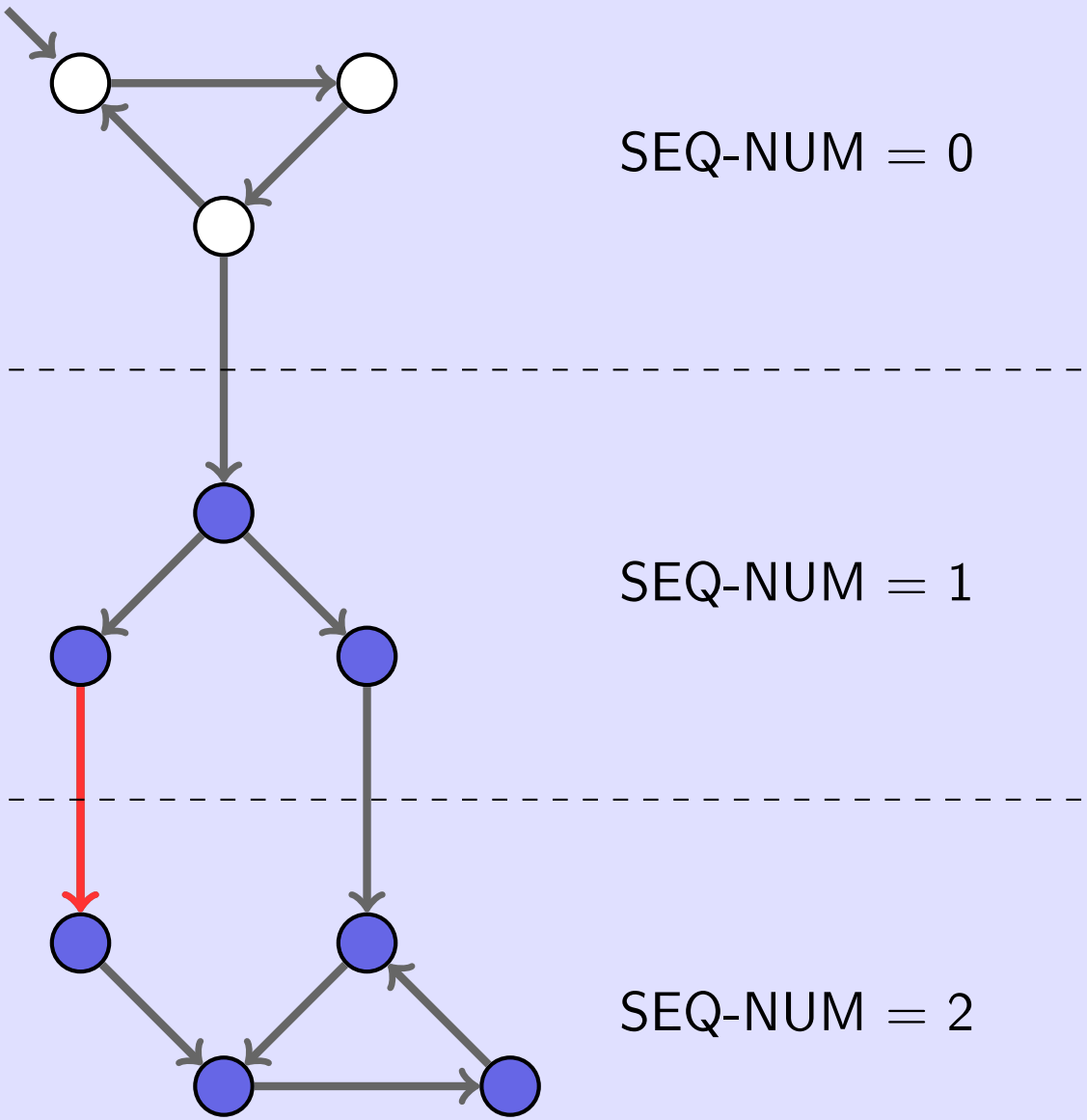
Sweep-line reduction



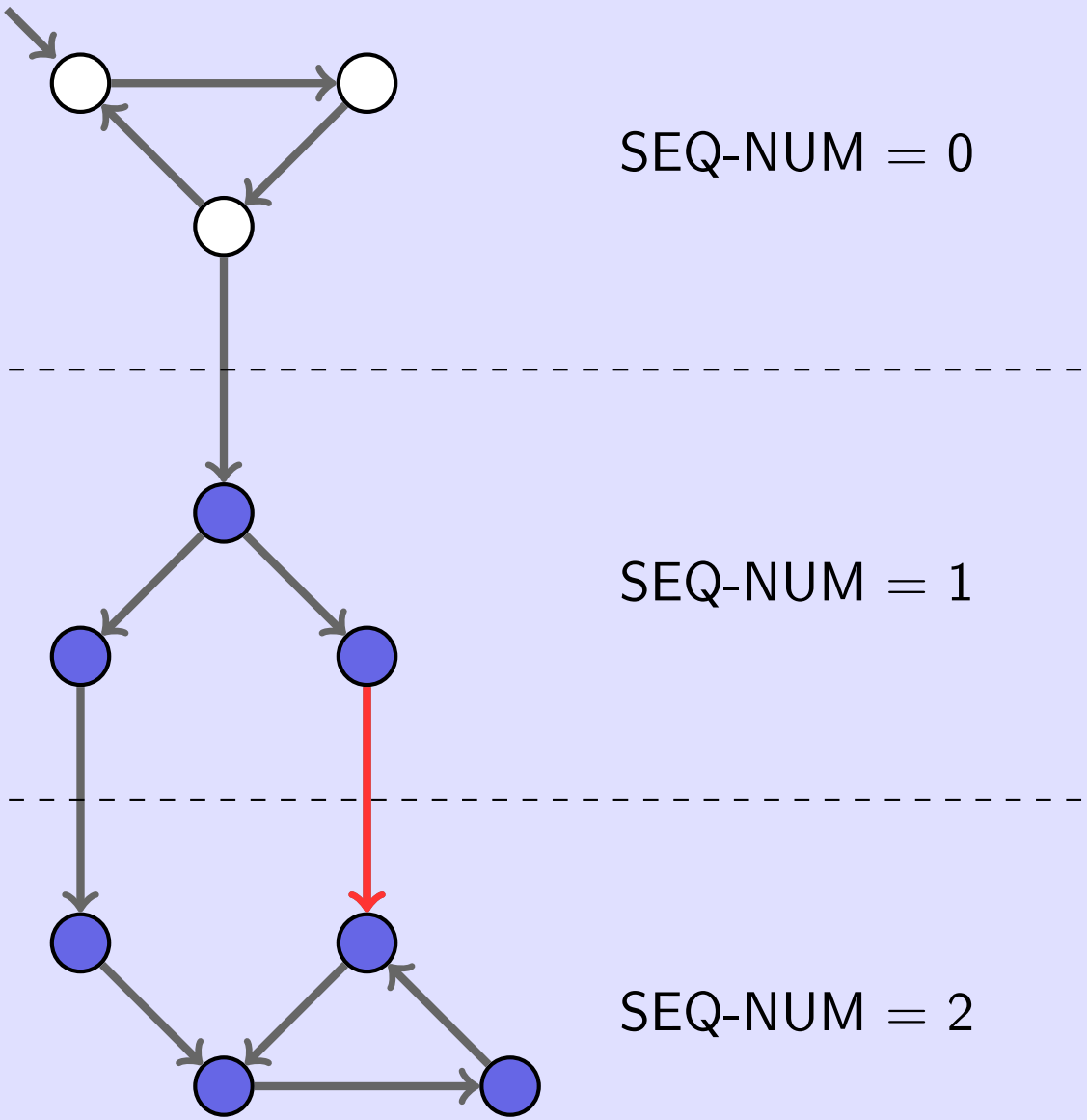
Sweep-line reduction



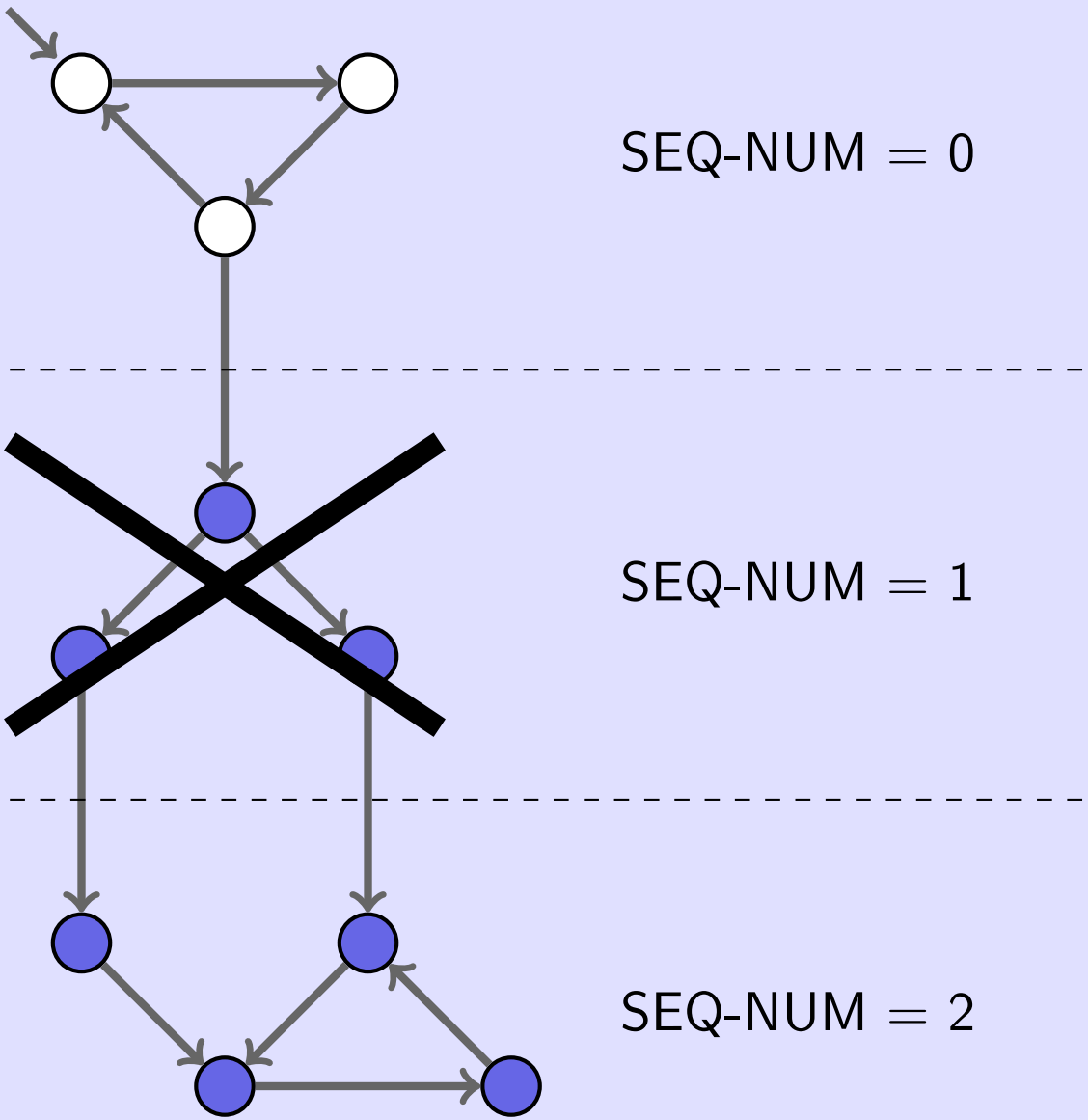
Sweep-line reduction



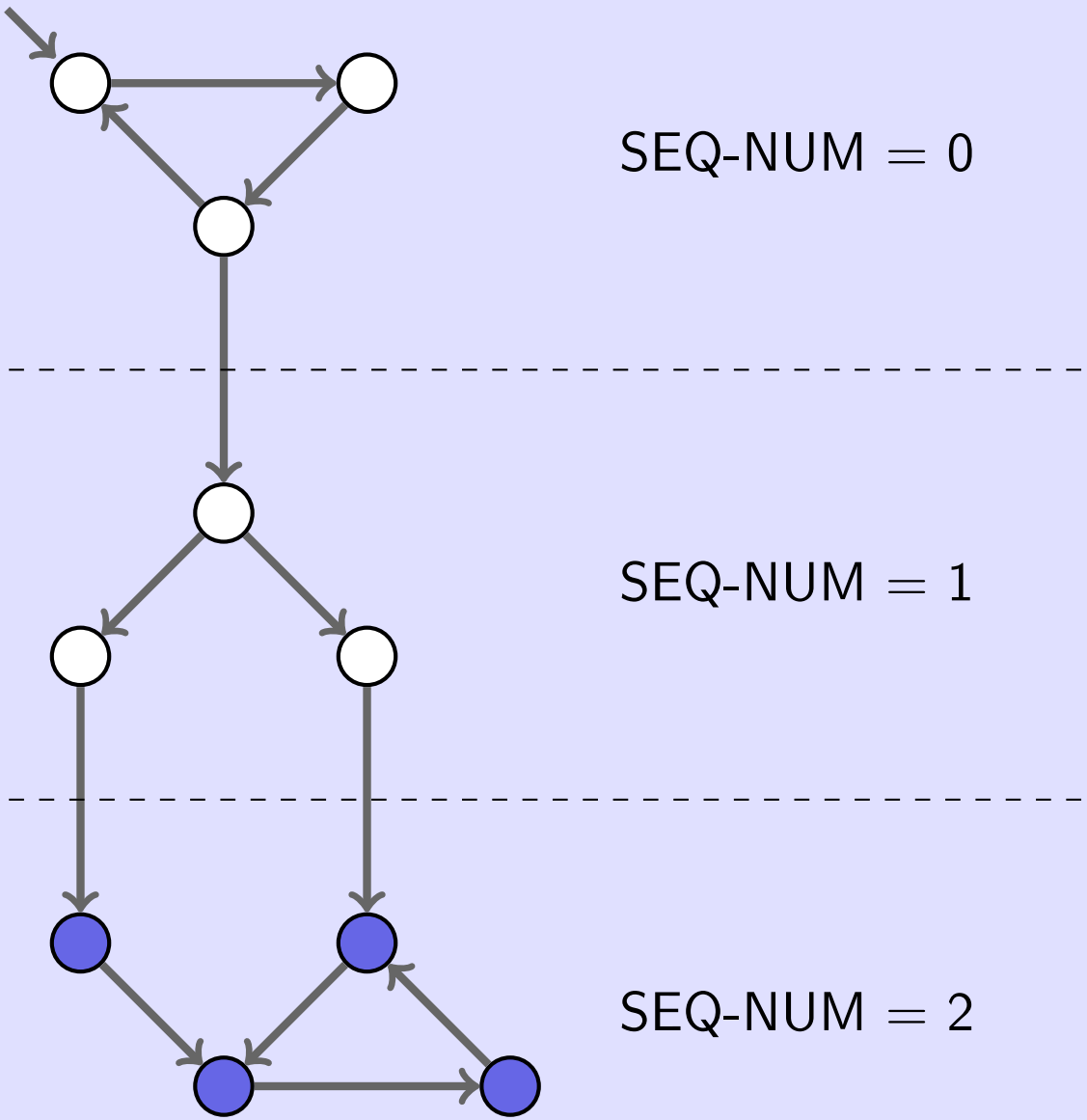
Sweep-line reduction



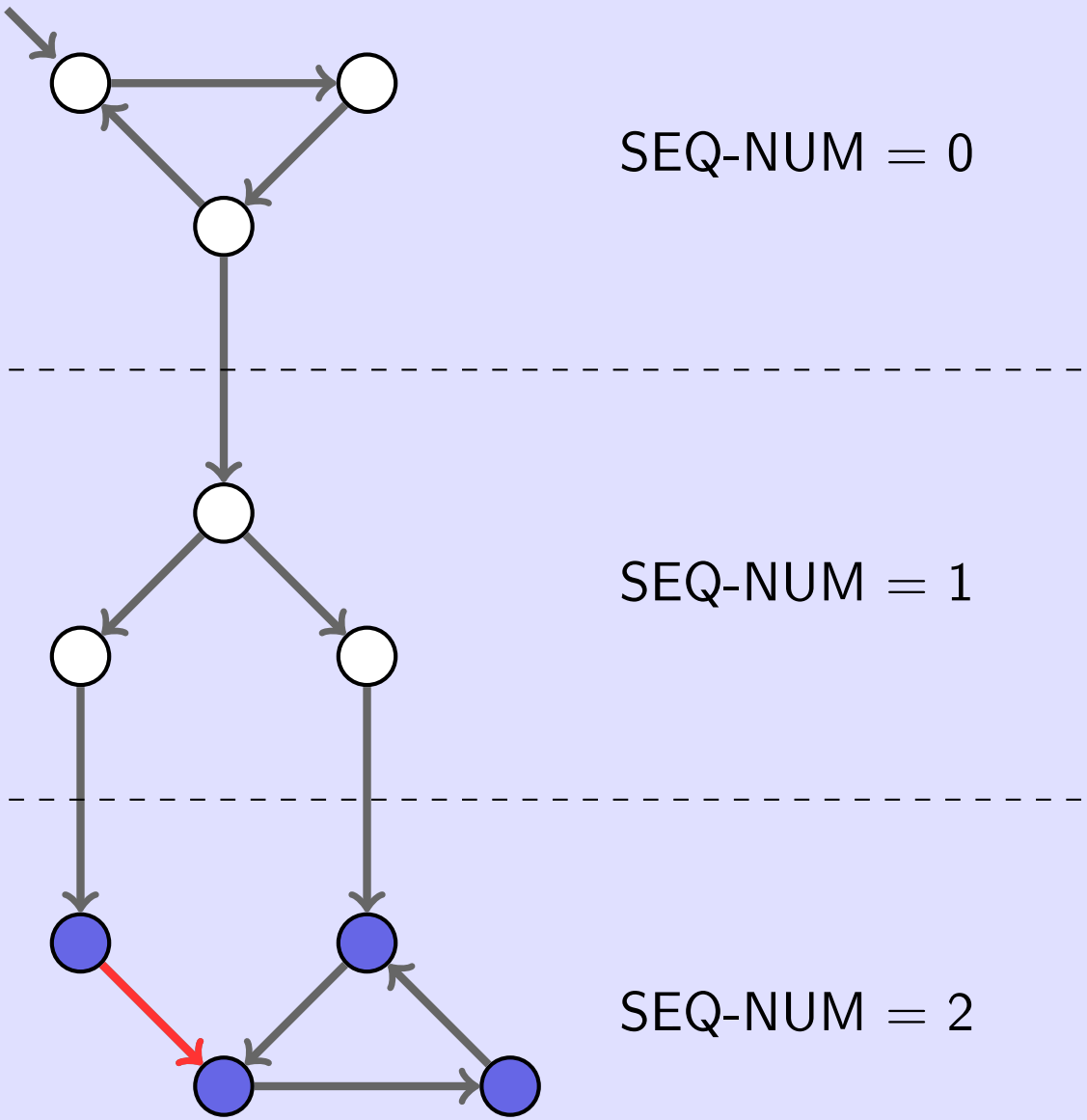
Sweep-line reduction



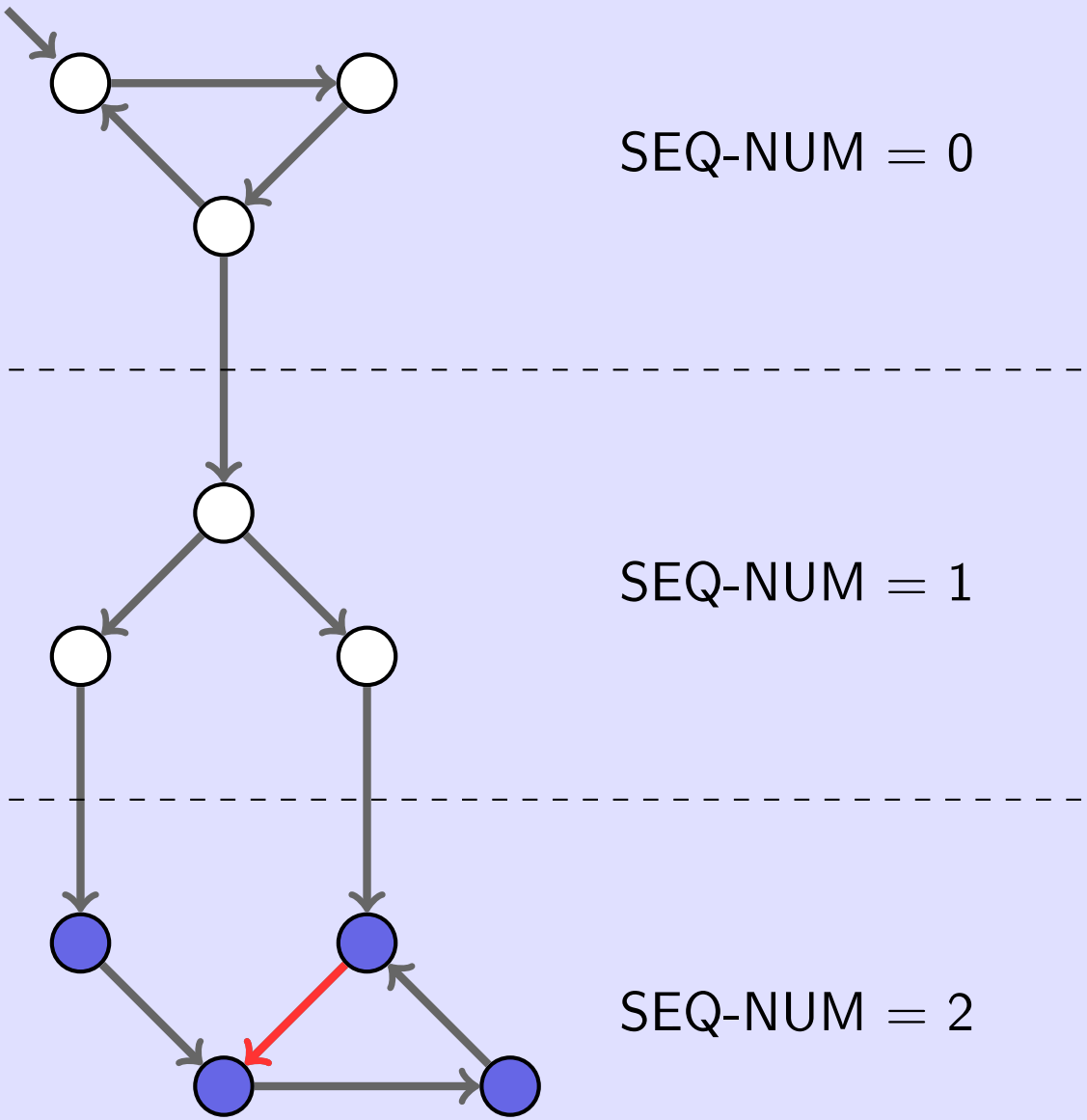
Sweep-line reduction



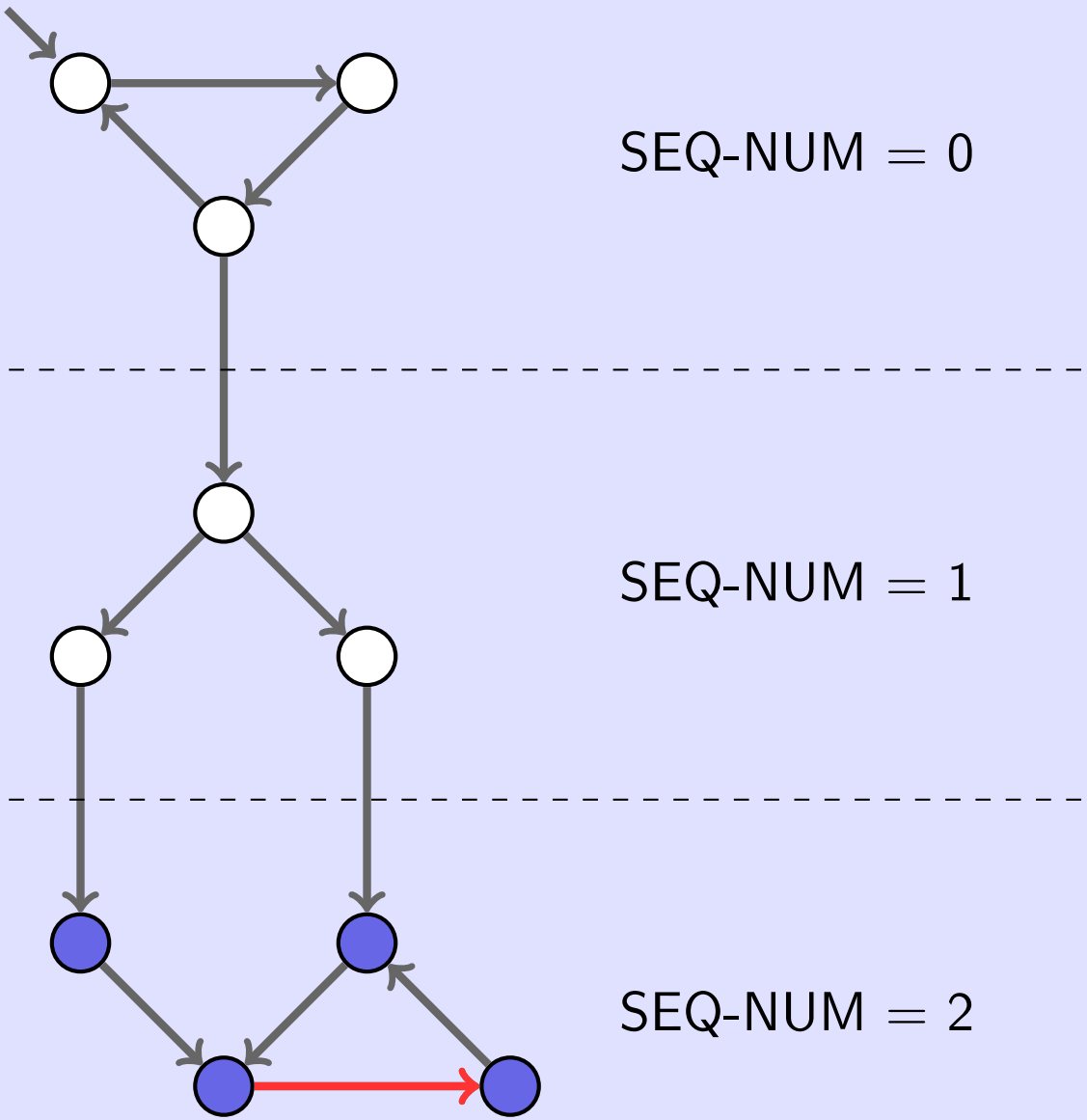
Sweep-line reduction



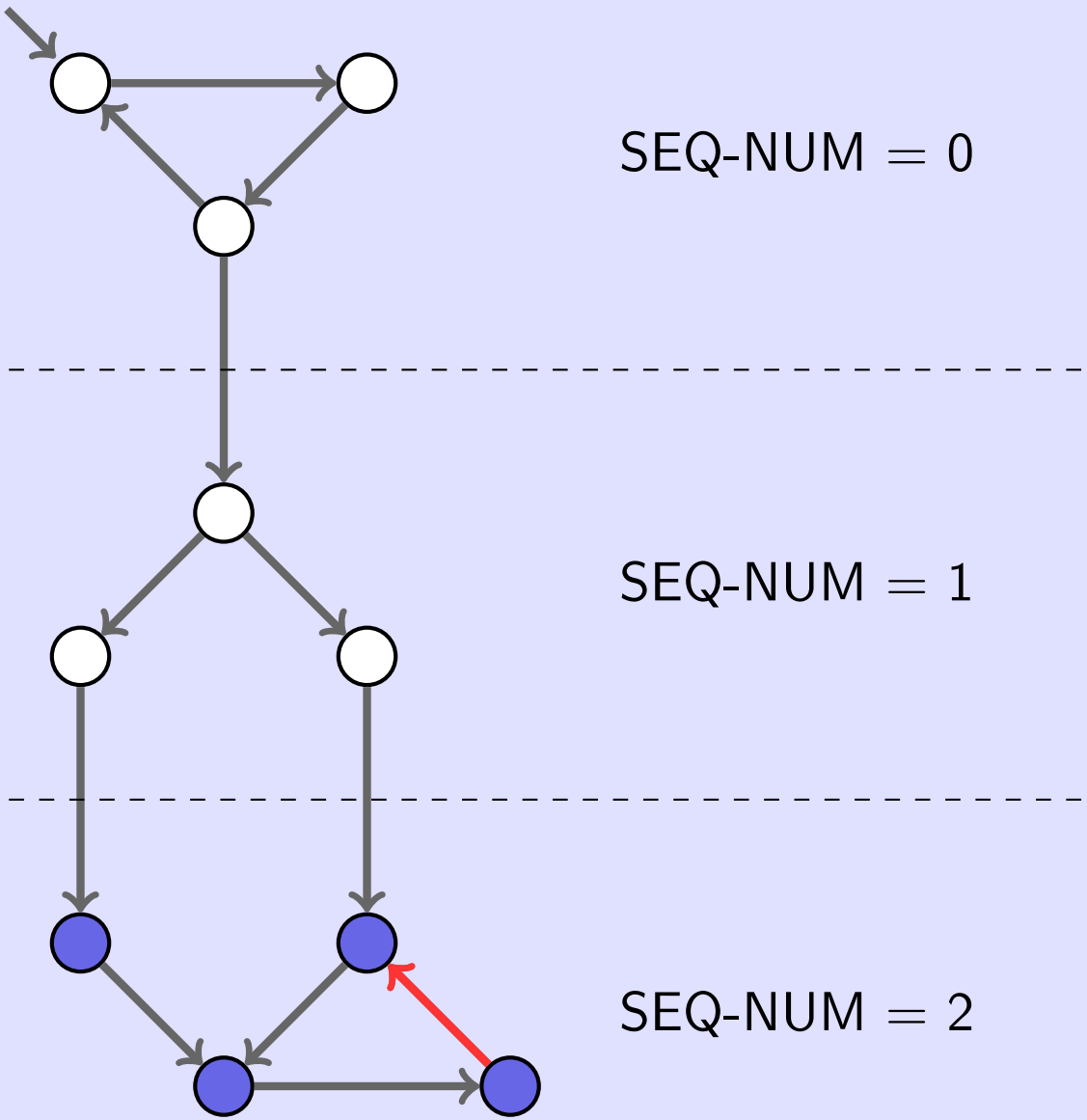
Sweep-line reduction



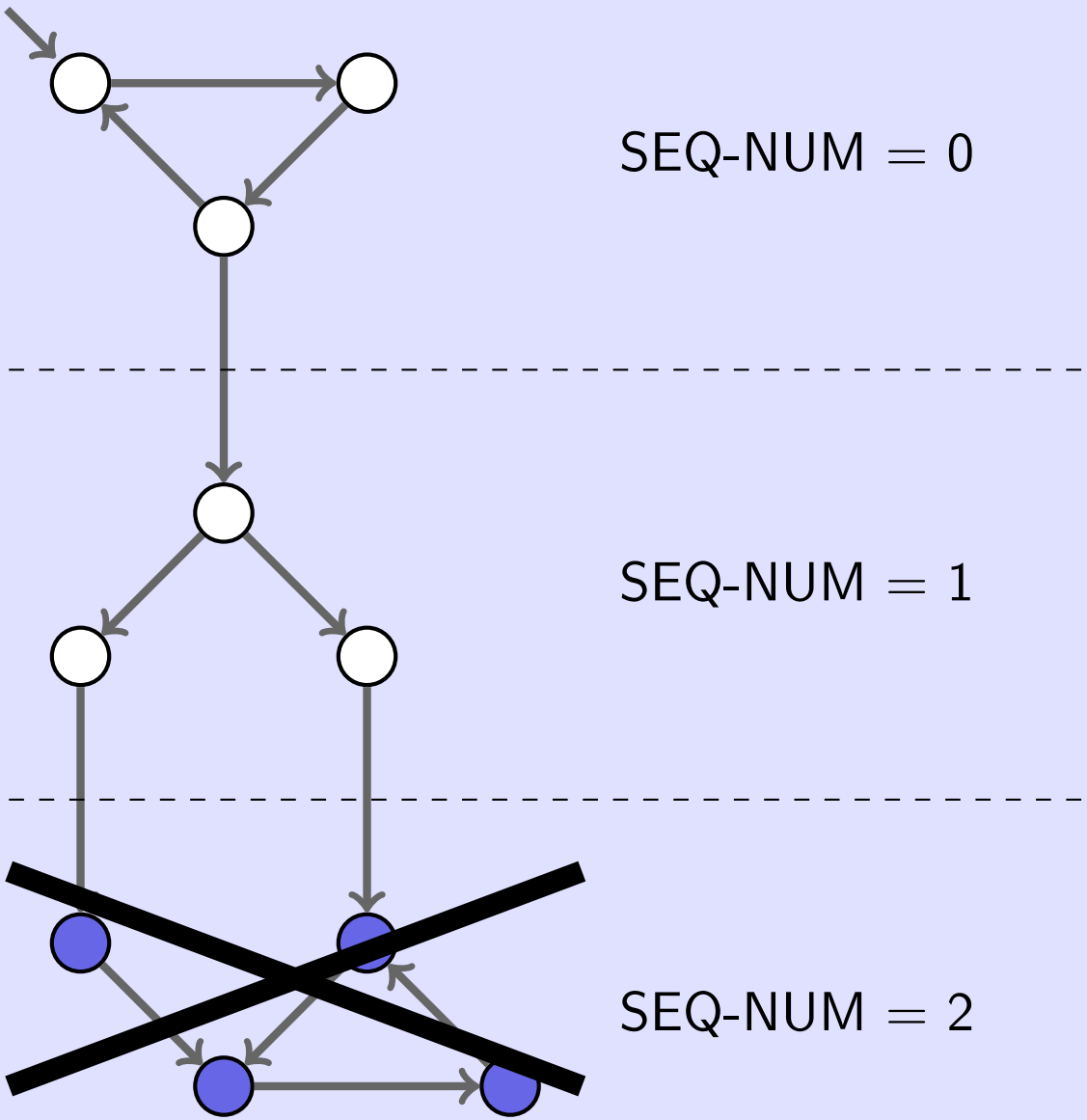
Sweep-line reduction



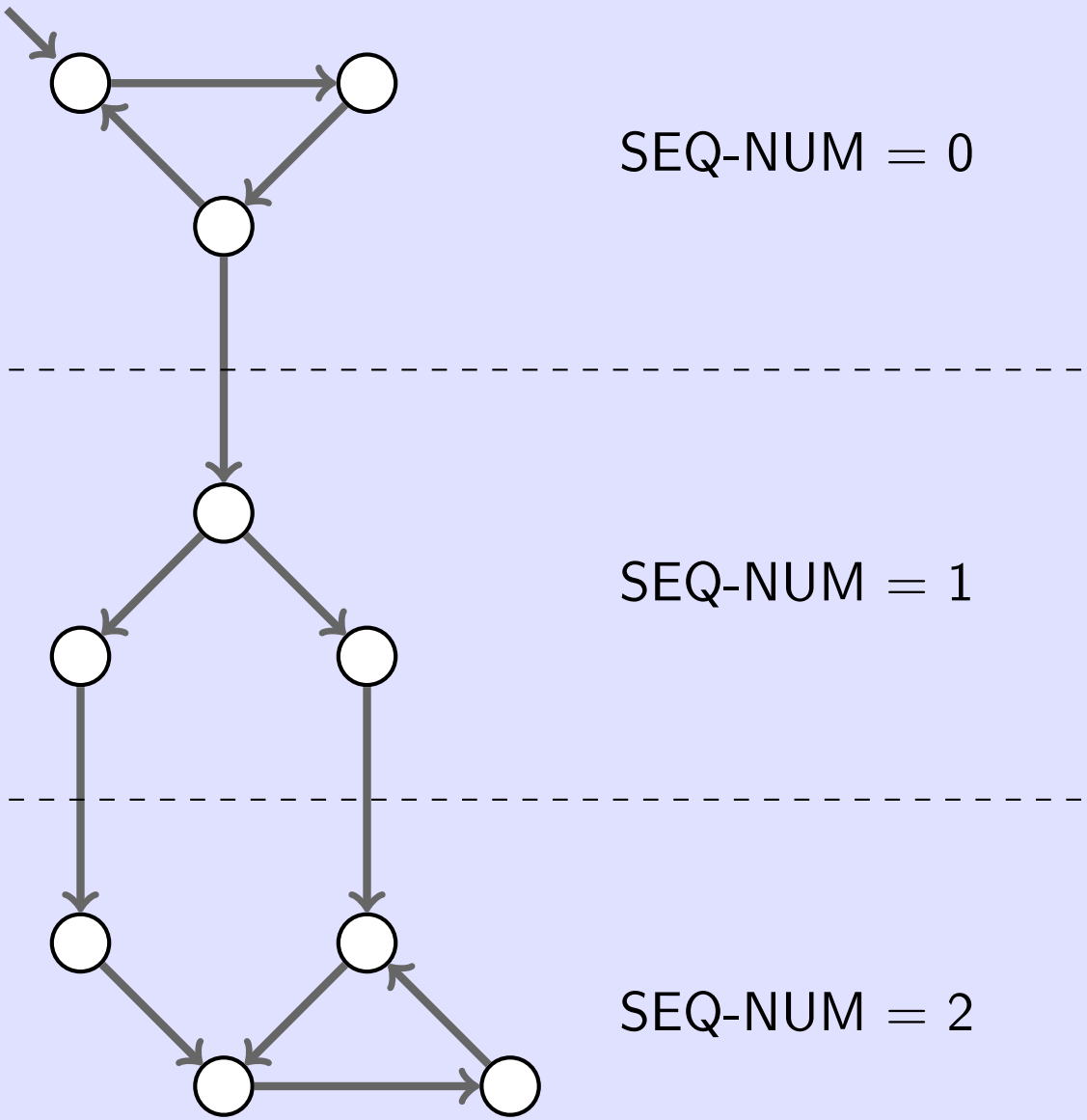
Sweep-line reduction



Sweep-line reduction



Sweep-line reduction



State caching + sweep-line reduction

- Motivation: the sweep-line reduction is not helpful as long as we work on states with the same progress value (i.e., with the same SEQ-NUM on our example)
- state caching can help us for that
- Algorithm:
  - use the sweep-line algorithm
  - each time we start to visit states with a new progress value Ψ
    - perform a local search on states with progress value Ψ using state caching

$\Rightarrow$ no need to store all the states with the same progress value at the same time

- easy to combine: the sweep-line algorithm is also an instance of GSEA (where open set = priority queue)
- take care that the state-caching reduction does not delete persistent states needed by the sweep-line algorithm to terminate

Experiments

State caching (SC) + sweep-line (SL)

- we used the same 135 DVE instances as in the first experiment
- for each model, we generated its state space and then automatically derived from it 6 progress mappings identified by a level ($0 \rightarrow 5$)
- a higher level means
  - a more precise progress mapping
  - more different progress values
  - smaller classes of states with the same progress value
  - a better memory reduction
- Example:

level 0 $\rightarrow$ SEQ-NUM of sender
level 1 $\rightarrow$ (SEQ-NUM of sender, SEQ-NUM of receiver)
level 2 $\rightarrow$ (SEQ-NUM of sender, SEQ-NUM of receiver, variable i of sender)
...

Experiments

State caching (SC) + sweep-line (SL)

Model	States	Level	SL		SL + SC	
			S	V	S	V
bopdp.2	25,685	3	52.4%	231%	20.4%	498%
		4	26.8%	242%	14.9%	480%
		5	13.7%	259%	12.7%	482%
brp.3	996,627	2	13.3%	145%	4.9%	203%
		3	10.6%	126%	9.7%	154%
		4	10.0%	122%	9.9%	120%
		5	8.1%	124%	8.2%	119%
pgm_protocol.3	195,015	3	2.4%	144%	2.1%	129%
		4	3.2%	110%	3.1%	110%
		5	9.0%	110%	9.0%	108%
synapse.6	625,175	3	26.6%	100%	7.4%	477%
		4	27.4%	316%	17.2%	477%
		5	18.1%	150%	17.4%	136%

Conclusion

- introduction of a generalization of state caching to GSEA
- ☺ usable with DFS, BFS, Best-First search, randomized search,...
- ☺ compatible with various techniques
  - chain reduction
  - distributed verification
  - sweep-line method
  - state reconstruction based reduction
- ☺ modeling language independent
- ☹ slightly increase memory requirements (5 bytes per state)

Conclusion
Experimental evaluation

- DFS + SC
  - ☺ good memory reduction
  - ☹ frequent time explosions
  - ☹ very sensitive on the caching strategy
- BFS + SC
  - ☹ memory reduction is not as good as with DFS
  - ☺ no time explosion (+30–40% on the average)
  - ☺ caching strategy has less impact
- SL + SC
  - ☺ can further enhance the sweep-line reduction by a factor of 2–3
  - ☺ no need to provide a very fine-tuned progress mapping

Perspectives

- experiment with other formalisms, especially CPNs
- experiment with the combination with other reduction techniques (e.g., distributed search)