UNIVERSITÀ DELLA CALABRIA

Tree Decomposition

Sebastiano A. Piccolo

Roadmap

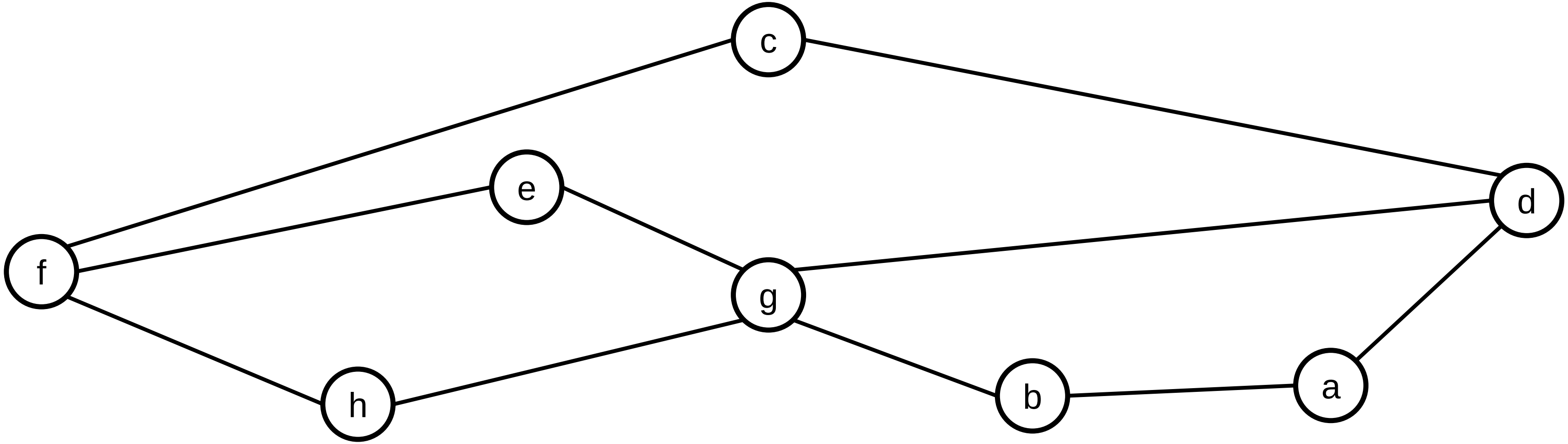
1. NP-Hard problems on trees
2. Treewidth and Tree Decomposition
3. Cops and robbers
4. Computing a Tree Decomposition

NP-Hard problems on trees

- NP-Hard problems on graphs, generally, cannot be solved exactly because of their prohibitive computational cost
  - Greedy Approximation
  - Heuristics: Simulated Annealing, Genetic Algorithm, ACO, …
  - Polynomial Time Approximation Scheme (PTAS)
- On trees, many NP-Hard problems can be solved exactly in polynomial time
  - Trees have a computationally convenient structure
    - Sub-trees are independent

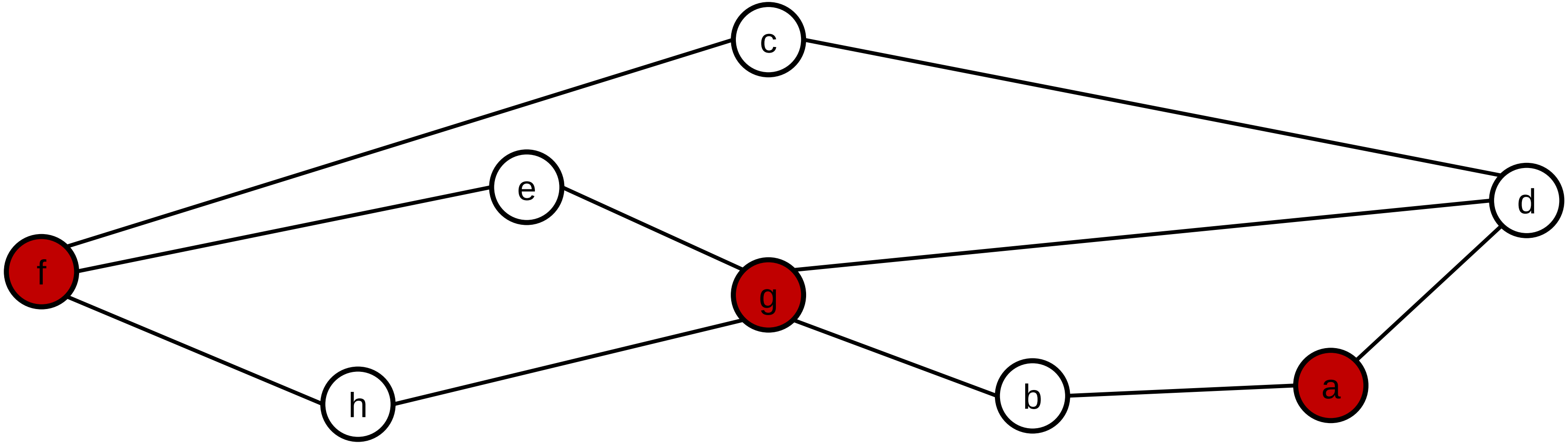
Maximum Independent Set (MIS)

Let $G = (V, E)$ be a graph, an independent set is a subset of nodes such that there is no edge between them: $S = \{C \subset V, (u, v) \notin E\ \forall u, v \in C\}$



Maximum Independent Set (MIS)

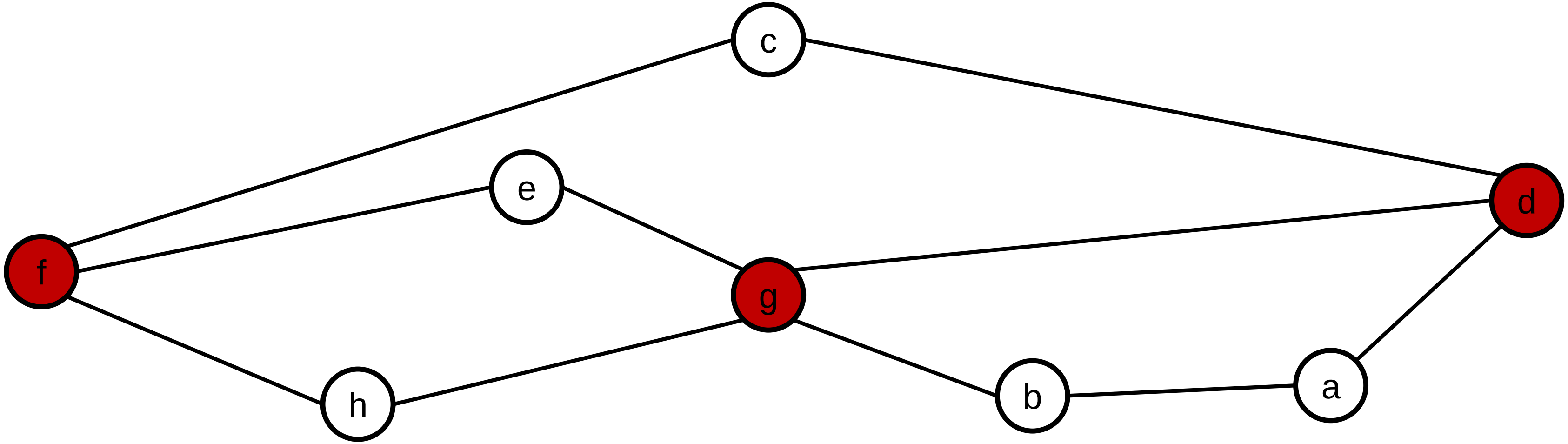
Let $G = (V, E)$ be a graph, an independent set is a subset of nodes such that there is no edge between them: $S = \{C \subset V, (u, v) \notin E \; \forall u, v \in C\}$



$S = \{f, g, a\}$ is an independent set

Maximum Independent Set (MIS)

Let $G = (V, E)$ be a graph, an independent set is a subset of nodes such that there is no edge between them: $S = \{C \subset V, (u, v) \notin E \ \forall u, v \in C\}$



$S = \{f, g, d\}$ is not an independent set

Maximum Independent Set (MIS)

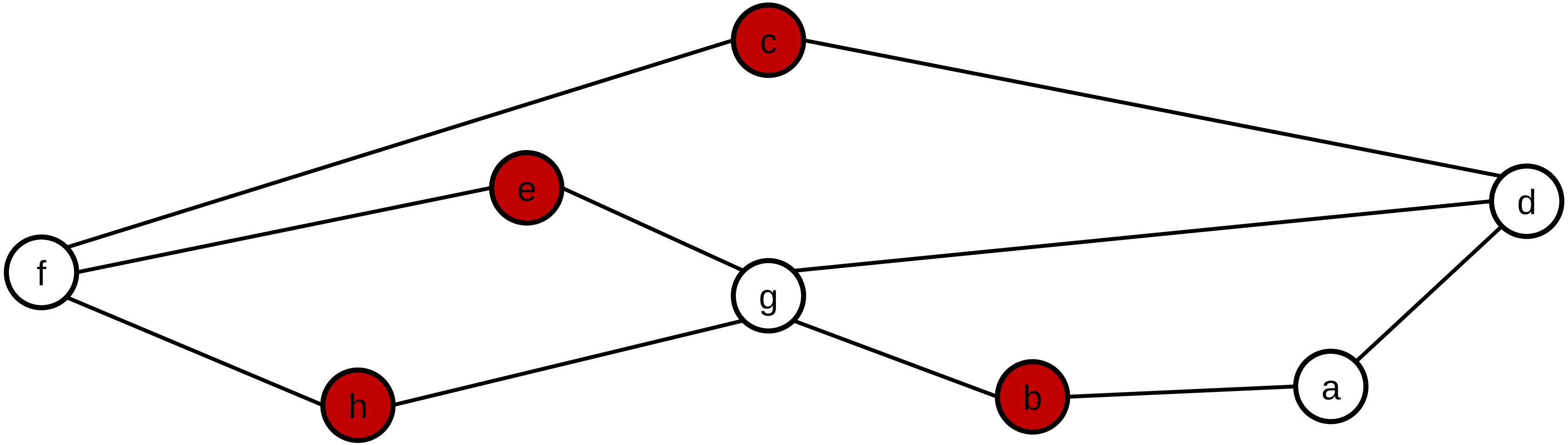
Let $G = (V, E)$ be a graph, an independent set is a subset of nodes such that there is no edge between them: $S = \{C \subset V, (u, v) \notin E \ \forall u, v \in C\}$

$$MIS = \underset{S}{\operatorname{argmax}} |S|, \qquad \forall S \in IS(G)$$

Maximum Independent Set (MIS)

Let $G = (V, E)$ be a graph, an independent set is a subset of nodes such that there is no edge between them: $S = \{C \subset V, (u, v) \notin E \ \forall u, v \in C\}$

$$MIS = \underset{S}{\operatorname{argmax}} |S|, \qquad \forall S \in IS(G)$$

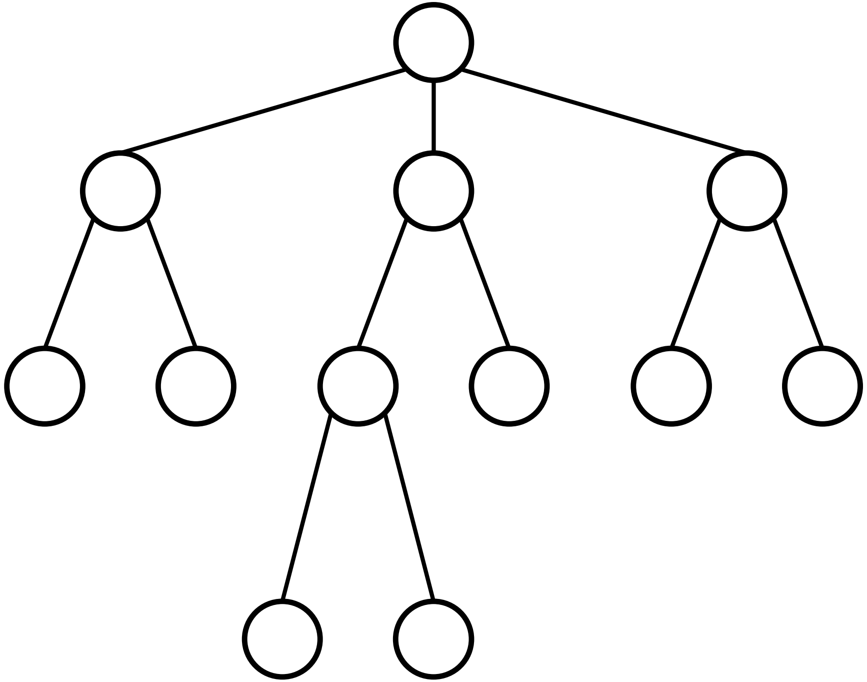


Maximum Independent Set (MIS)

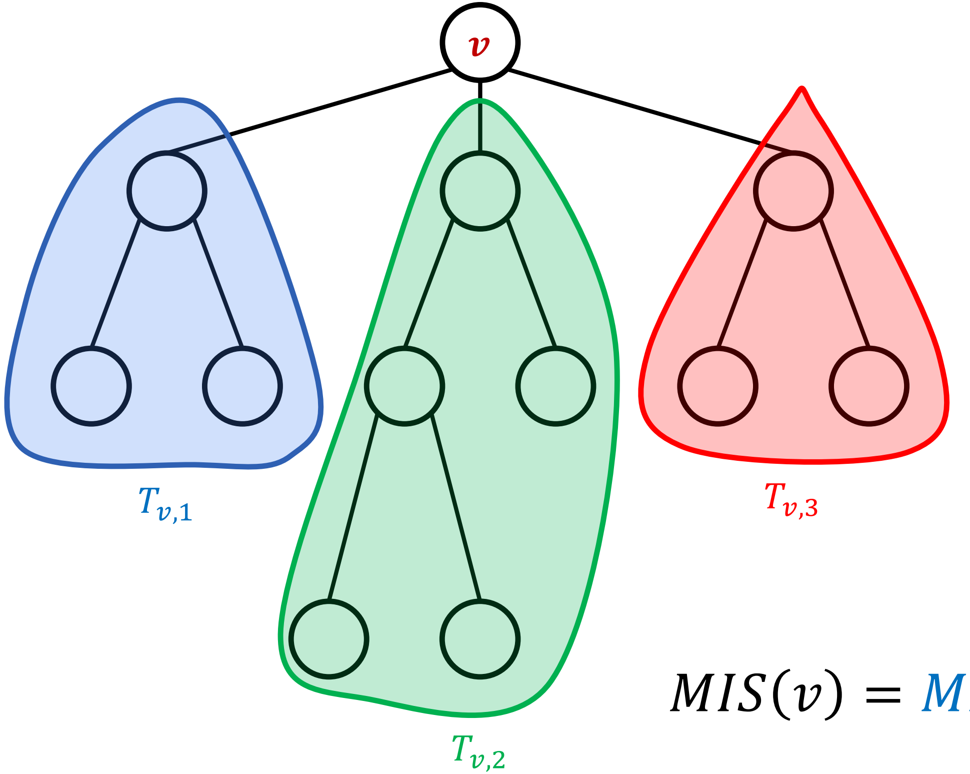
- MIS is NP-Complete
- Naive implementation requires $\mathcal{O}(n^2 2^n)$ computational time
  - "Fast" implementation in $\mathcal{O}(1.1996^n)$

- On trees, we can solve MIS in linear time!

Maximum Independent Set on Trees



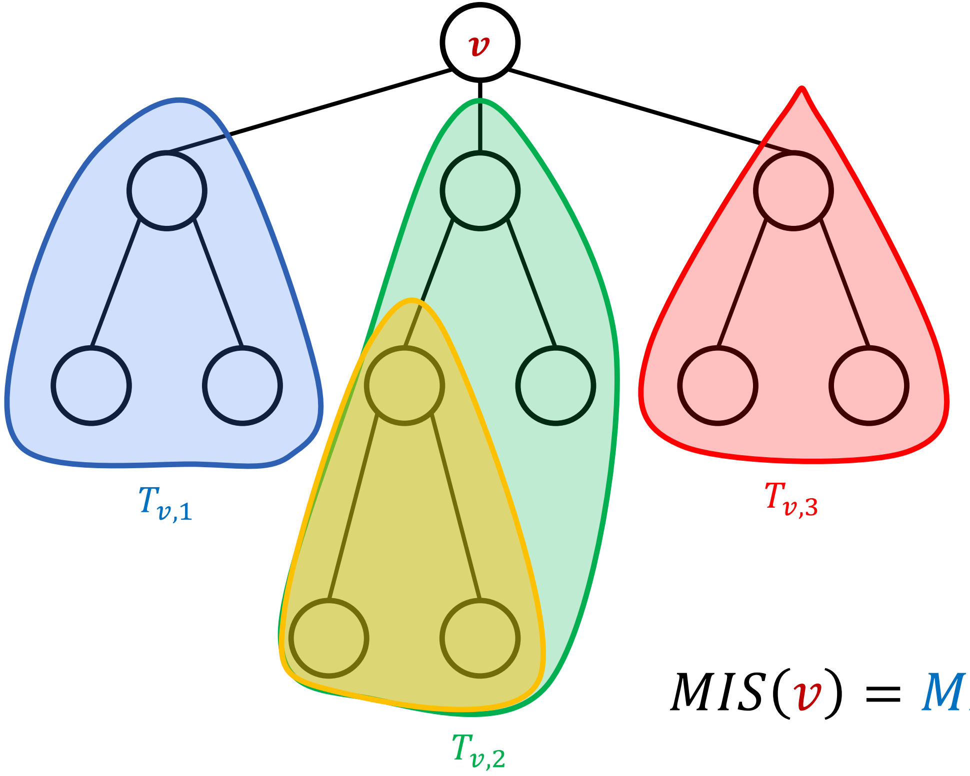
Maximum Independent Set on Trees



- Optimal substructure
  - optimal solution implies optimal solutions of subproblems

$$MIS(v) = MIS(T_{v,1}) + MIS(T_{v,2}) + MIS(T_{v,3}) \pm v$$

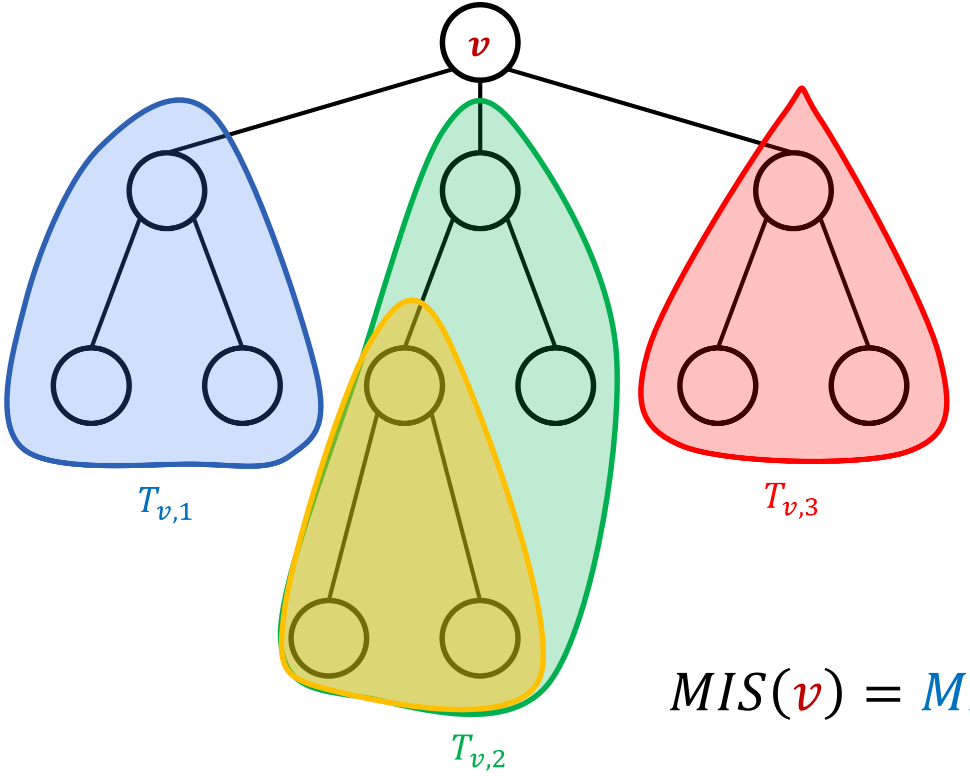
Maximum Independent Set on Trees



- Optimal substructure
  - optimal solution implies optimal solutions of subproblems
- Overlapping subproblems
  - subproblems are computed (reused) several times

$$MIS(v) = MIS(T_{v,1}) + MIS(T_{v,2}) + MIS(T_{v,3}) \pm v$$

Maximum Independent Set on Trees

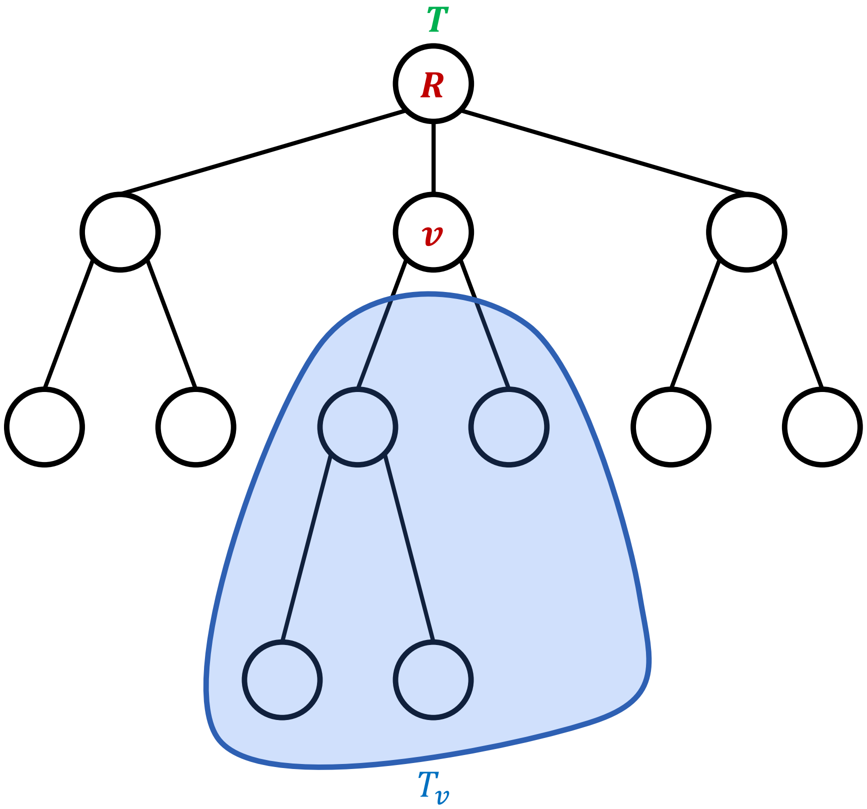


- Optimal substructure
  - optimal solution implies optimal solutions of subproblems
- Overlapping subproblems
  - subproblems are computed (reused) several times

- **Dynamic Programming**

$$MIS(v) = MIS(T_{v,1}) + MIS(T_{v,2}) + MIS(T_{v,3}) \pm v$$

Maximum Independent Set on Trees



For each vertex v we compute:

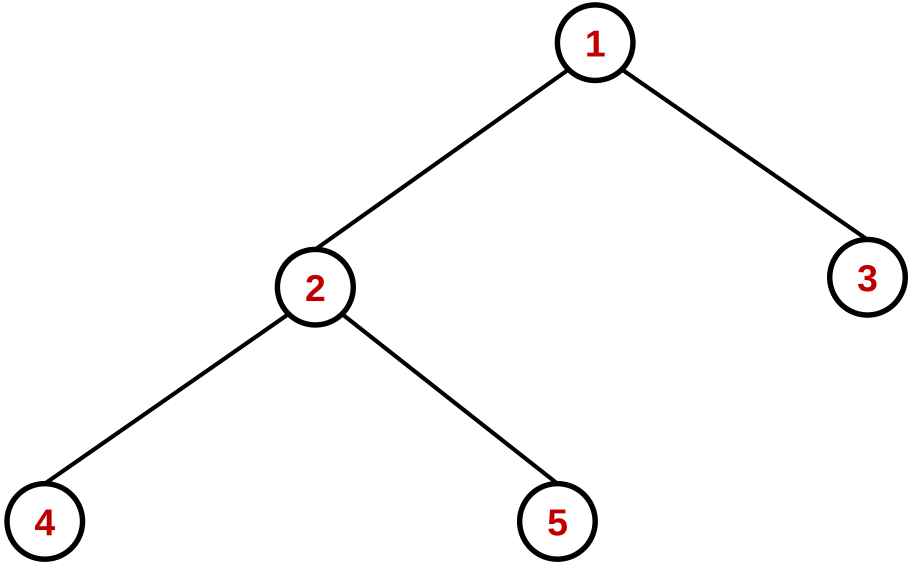
- $M^+[v] = |MIS(T_v) \cup \{v\}|$
- $M^-[v] = |MIS(T_v) \setminus \{v\}|$

For a vertex v with children $w_1, \dots, w_d$

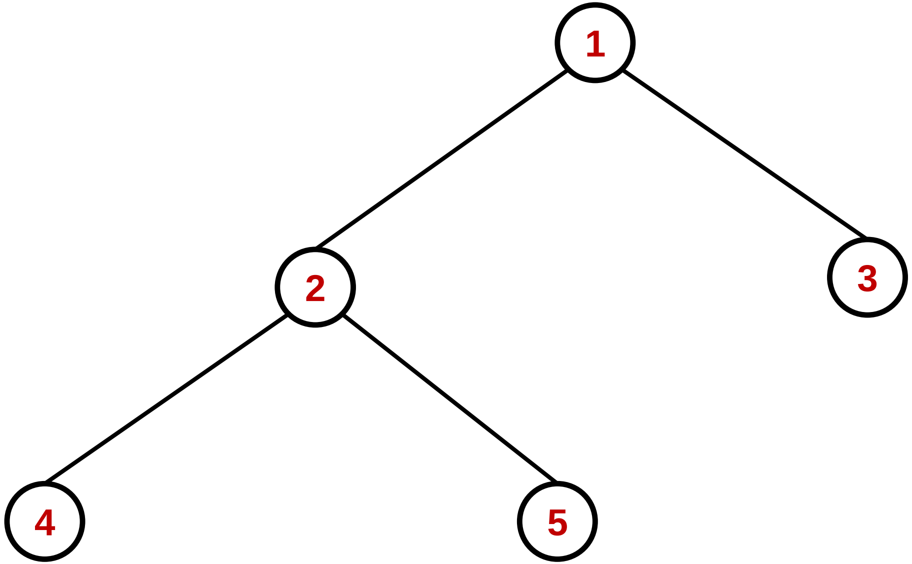
- $M^+[v] = 1 + \sum_{i=1}^{d} M^-[w_i]$
- $M^-[v] = \sum_{i=1}^{d} \max\{M^+[w_i], M^-[w_i]\}$

$$MIS(T) = \max\{M^+[R], M^-[R]\}$$

Maximum Independent Set on Trees

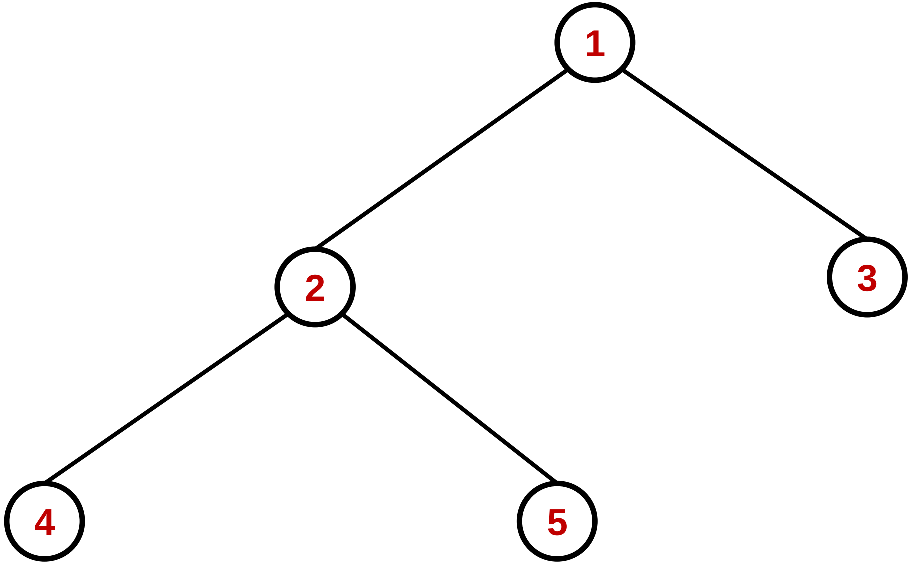


Maximum Independent Set on Trees



$M^{+}[4] = 1$
$M^{-}[4] = 0$

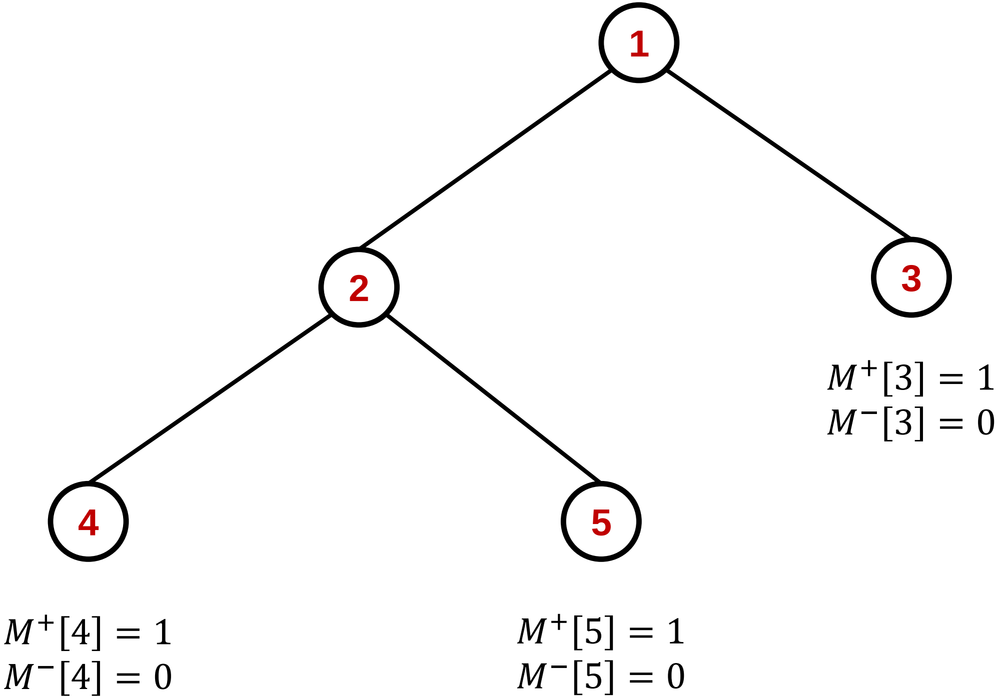
Maximum Independent Set on Trees



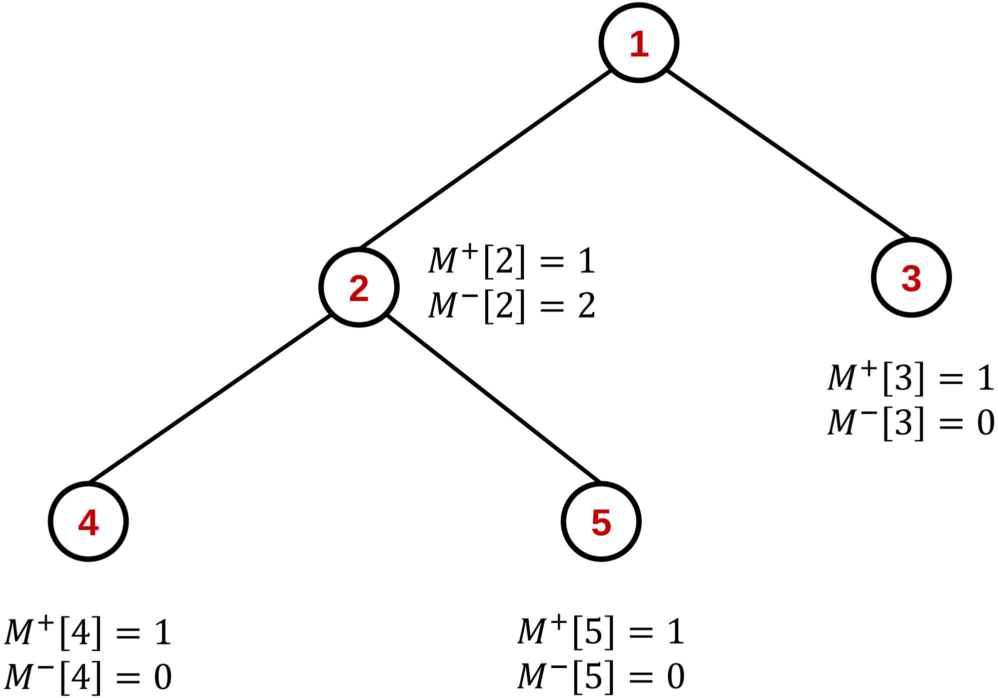
$M^+[4] = 1$
$M^-[4] = 0$

$M^+[5] = 1$
$M^-[5] = 0$

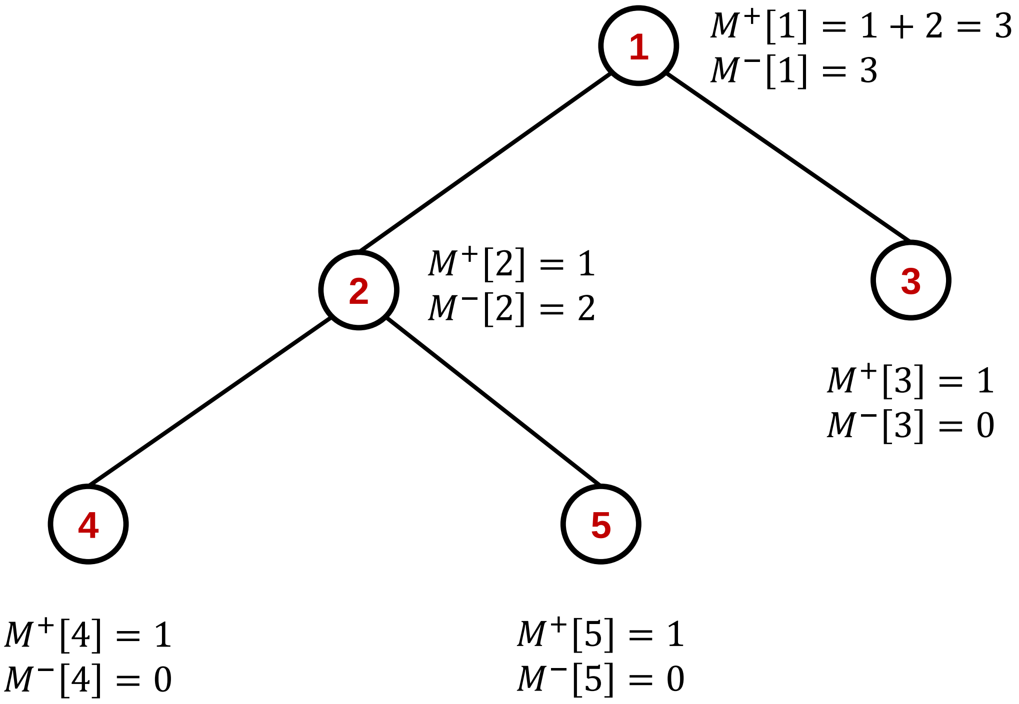
Maximum Independent Set on Trees



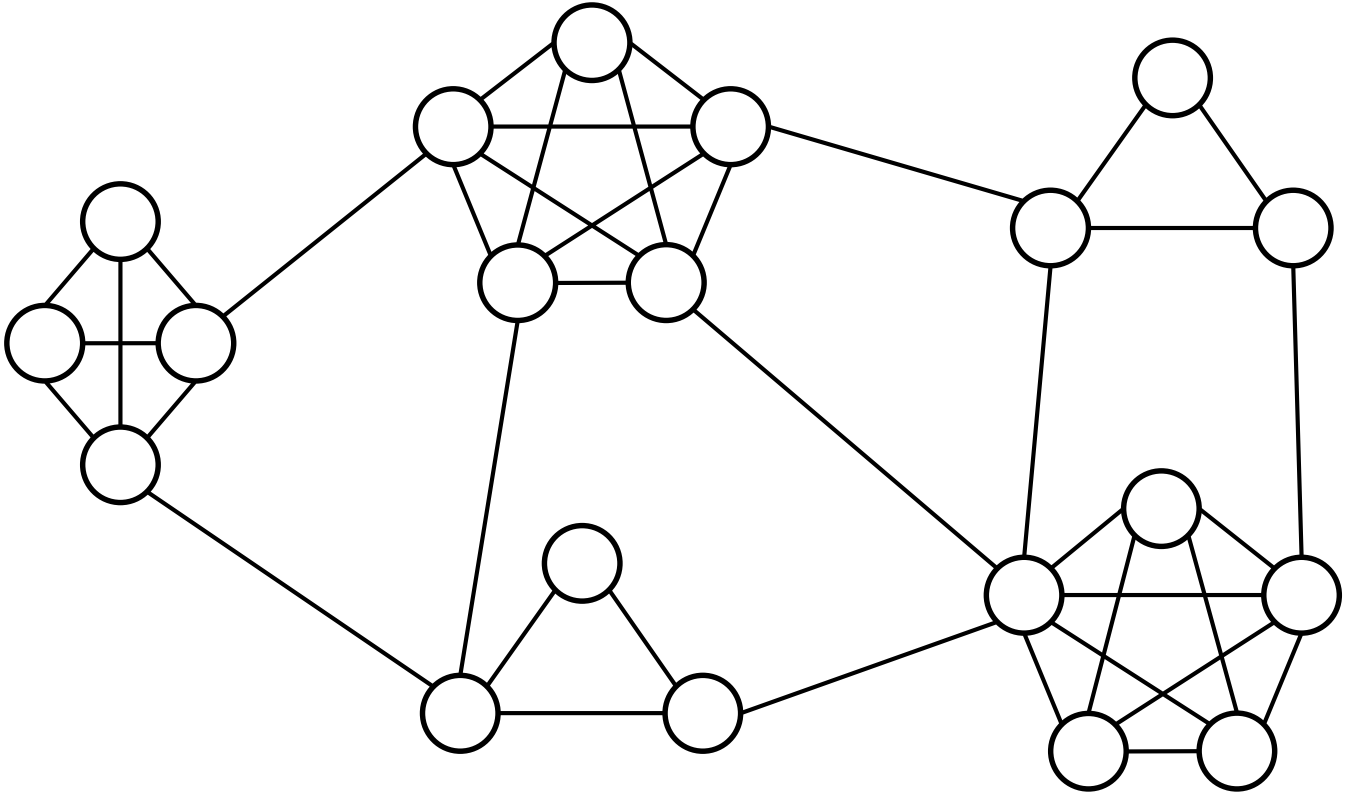
Maximum Independent Set on Trees



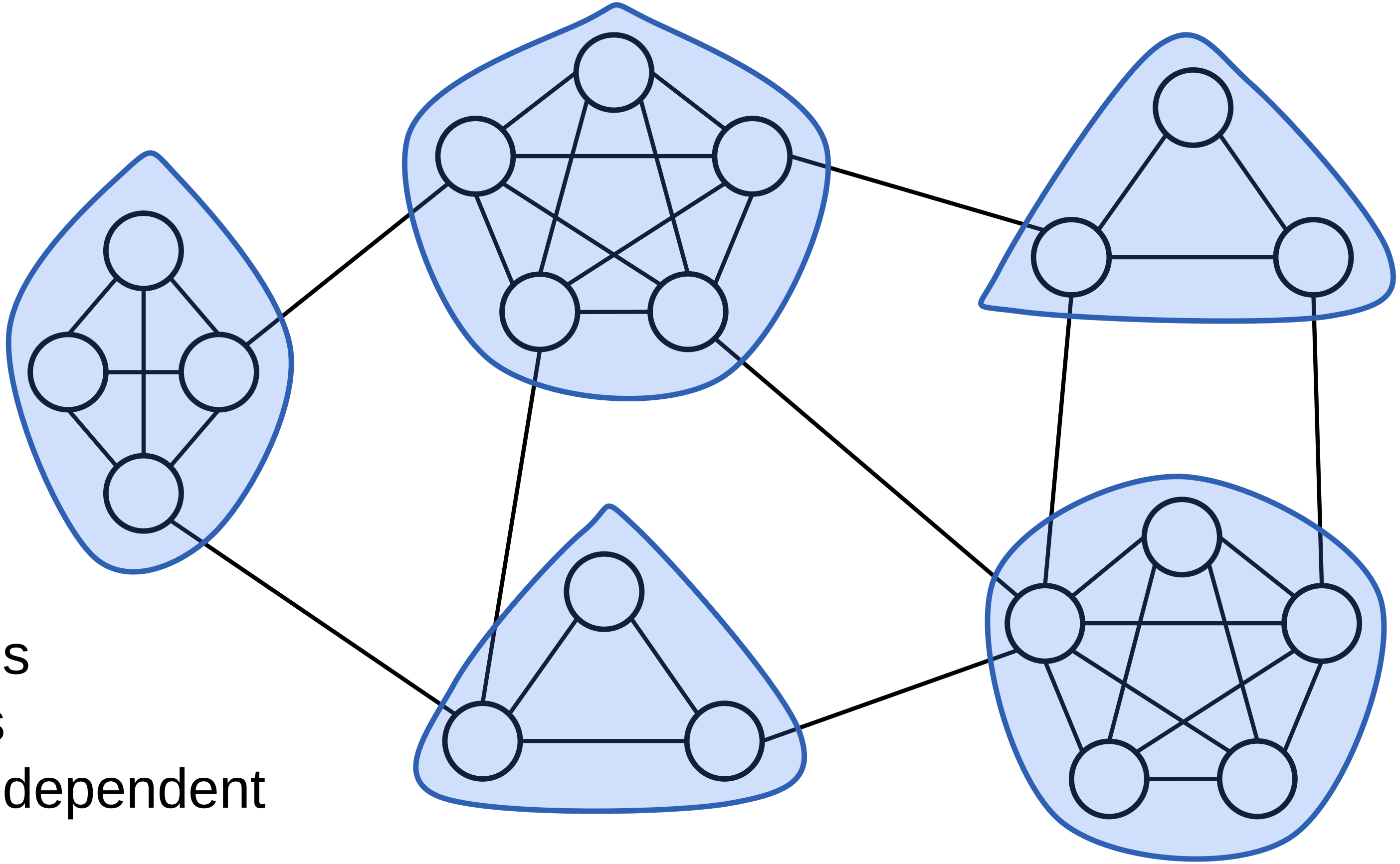
Maximum Independent Set on Trees



What is the MIS on this graph?

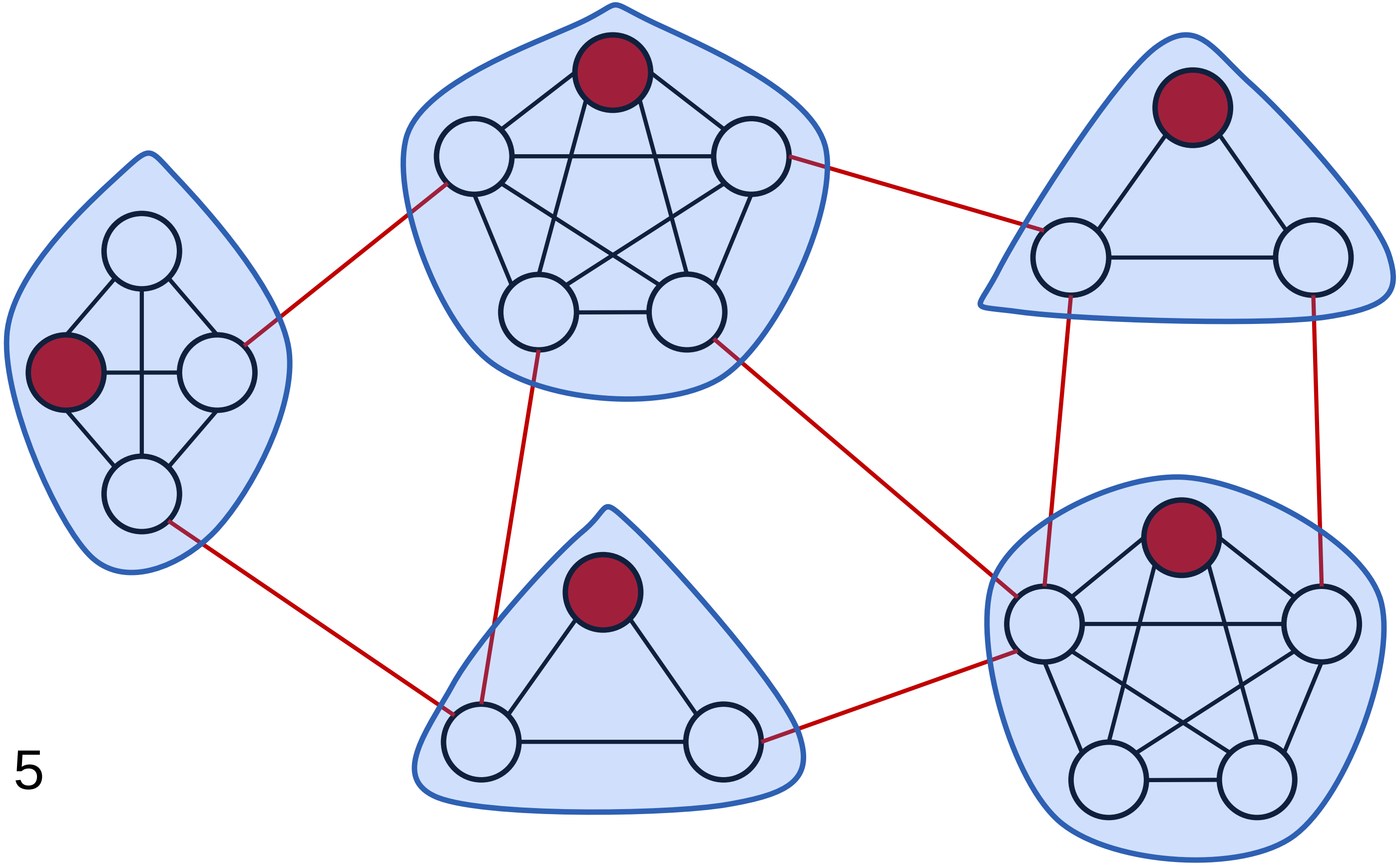


What is the MIS on this graph?



5 Modules
5 Cliques
Almost independent

What is the MIS on this graph?

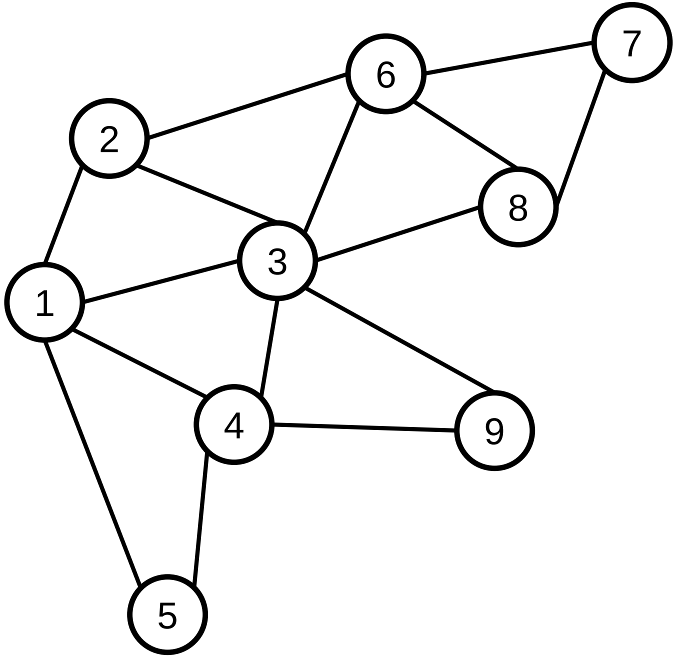


MIS = 5

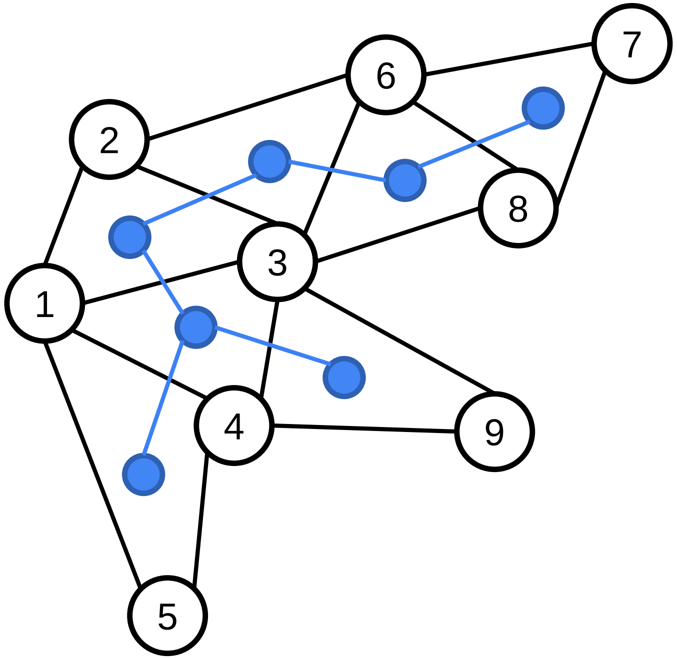
Tree decomposition: intuition

- Representing a graph as a tree T
  - Nodes of T are small *modules*, called bags
  - Bags form subproblems
- We can apply dynamic programming on a tree decomposition

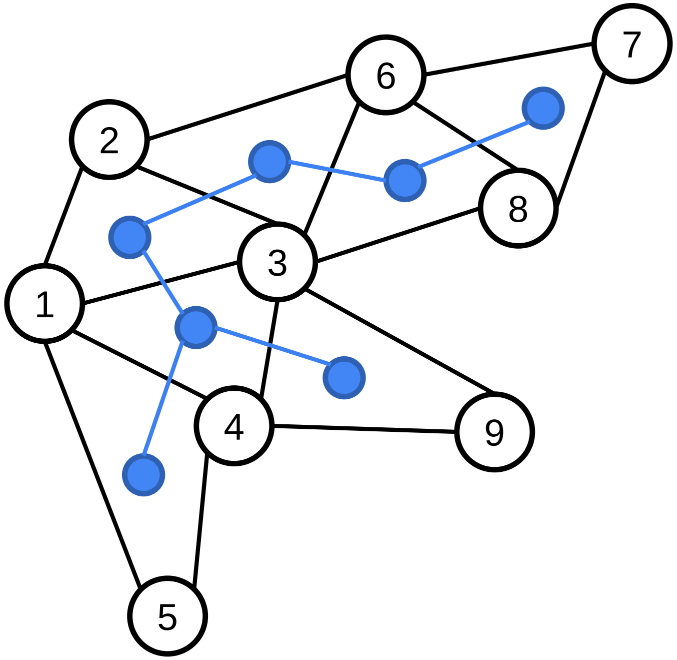
Tree decomposition: intuition

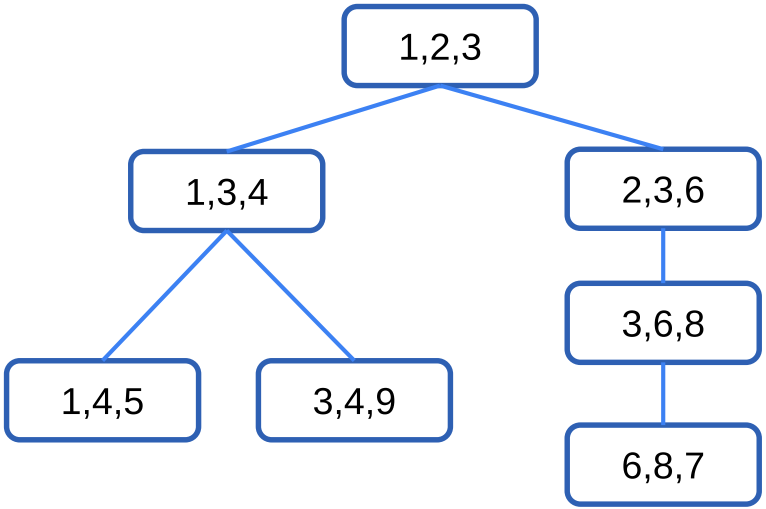


Tree decomposition: intuition

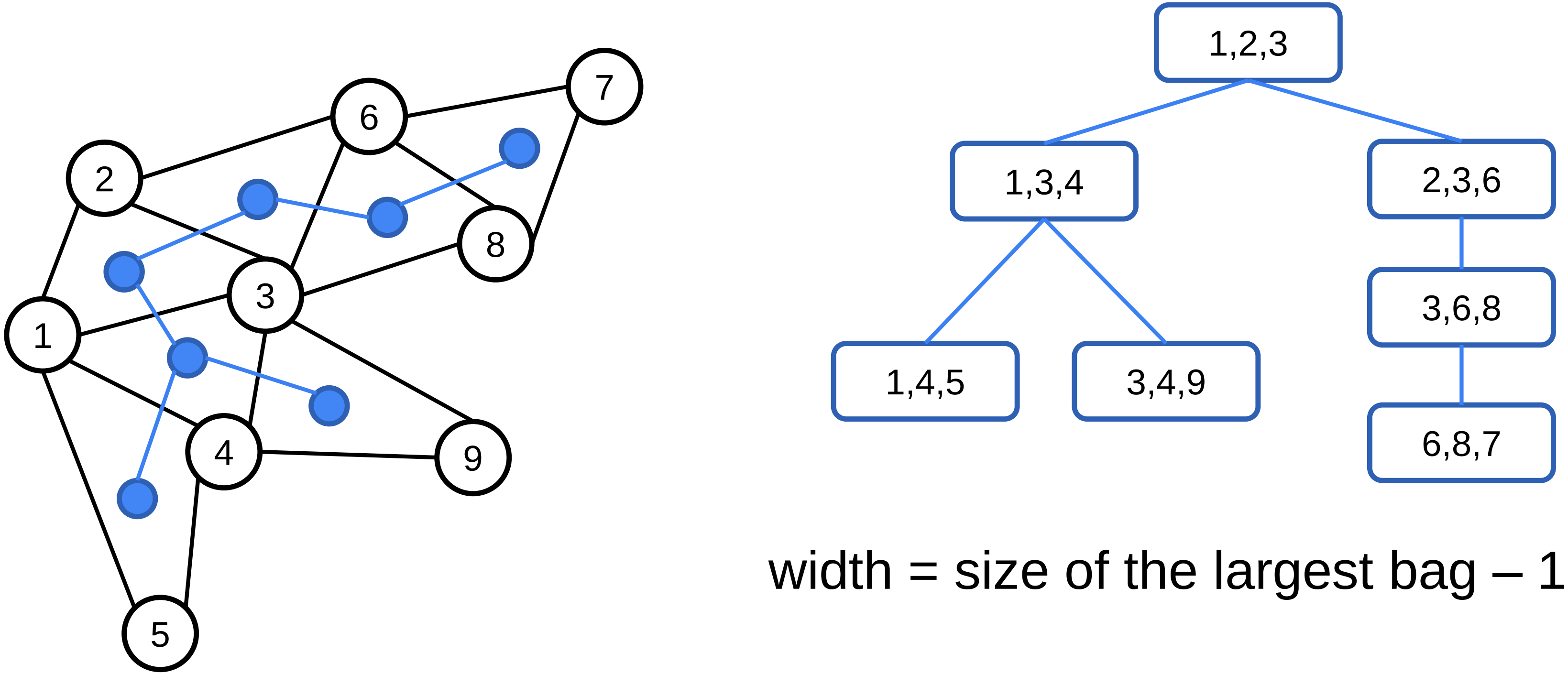


Tree decomposition: intuition





Tree decomposition: intuition



width = size of the largest bag – 1

Tree decomposition: definition

A tree decomposition of $G = (V, E)$ is a tree T of bags X such that:

- if $(u, v) \in E$ then u and v are together in some bag
- $\forall v \in V$ the bags containing v are connected in T

A graph can admit many tree decompositions

Treewidth

The treewidth is the smallest possible width among all the tree decompositions admitted by a graph

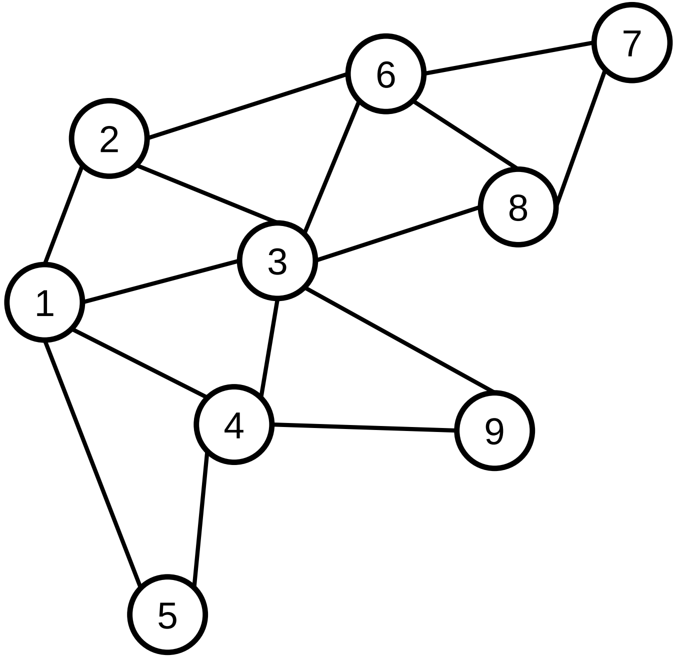
- tw(G) = 1 iff. G is a forest
- tw(G) = 2 iff. G is a series-parallel graph
- Deleting edges from G does not increase the treewidth
- Contracting edges does not increse treewidth
- Any clique in G must be in a bag

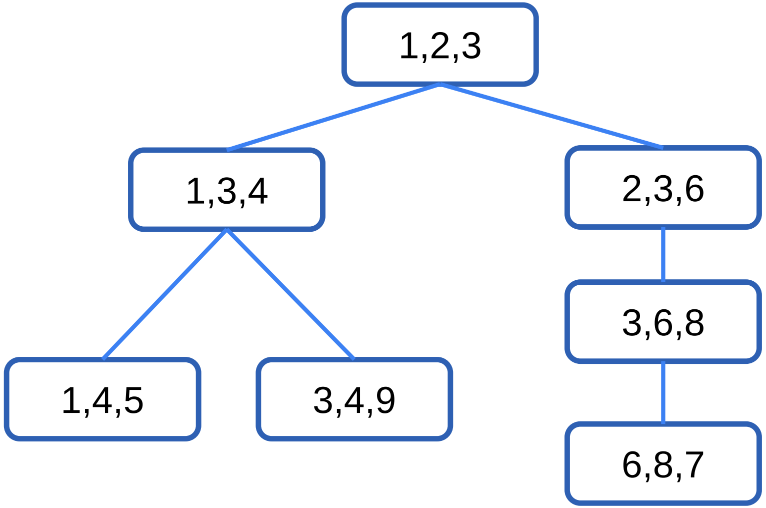
Cops and robbers

- One robber: very fast, can move on the graph
- k cops: assumed to be in helicopters (can jump through nodes)
- In order to win, cops need to corner the robber (blocking all the escape routes) and land on the same node in which the robber is

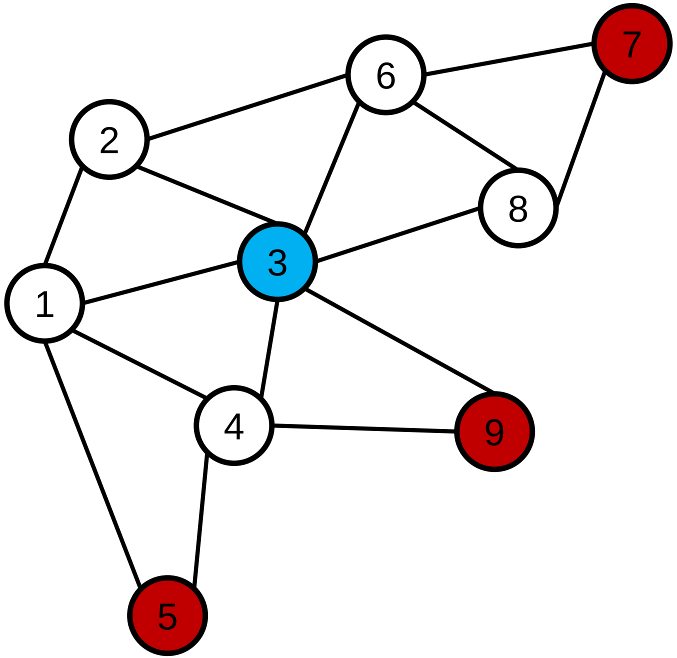
- Theorem: $tw(g) \leq k \Leftrightarrow k + 1$ cops can win the game
- Strategy given by the tree decomposition

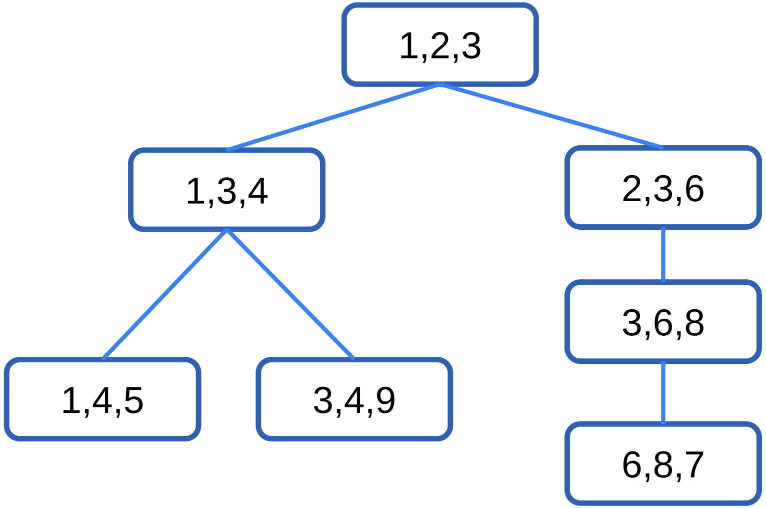
Cops and robbers



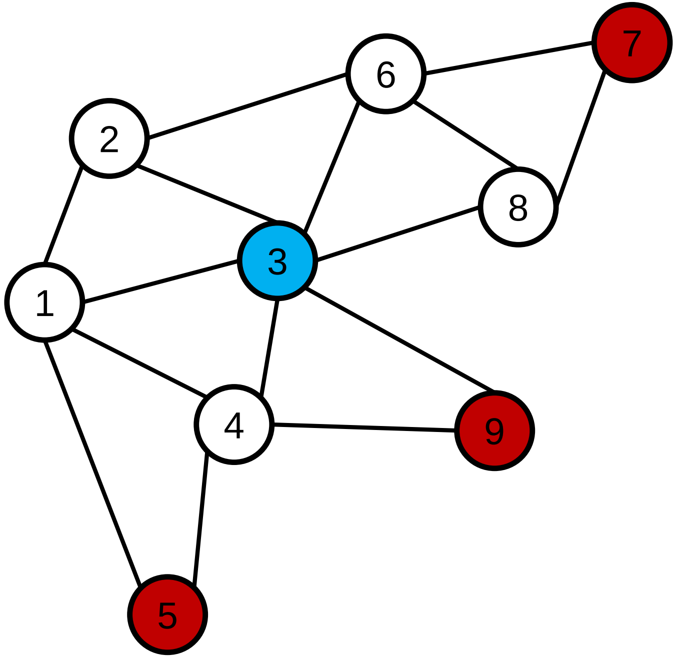


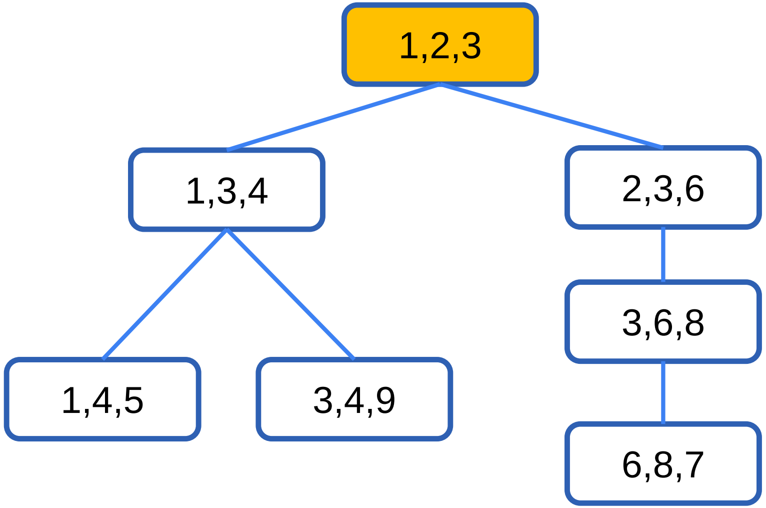
Cops and robbers



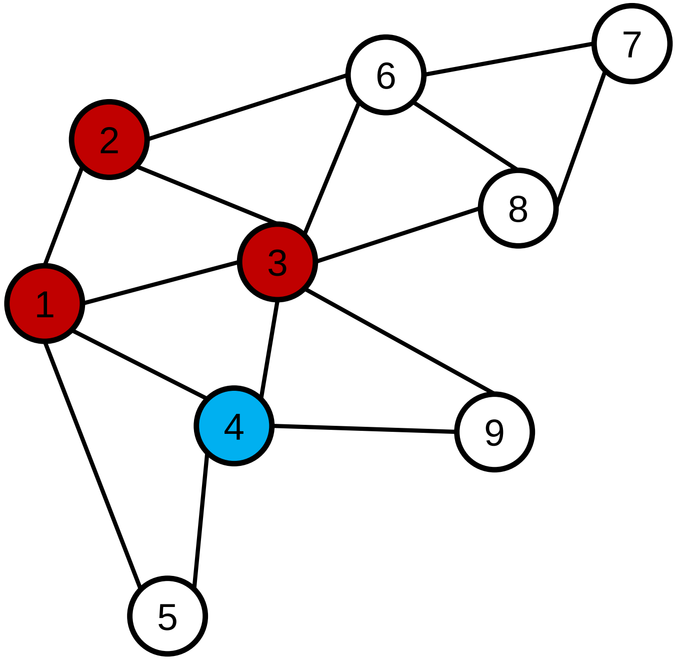


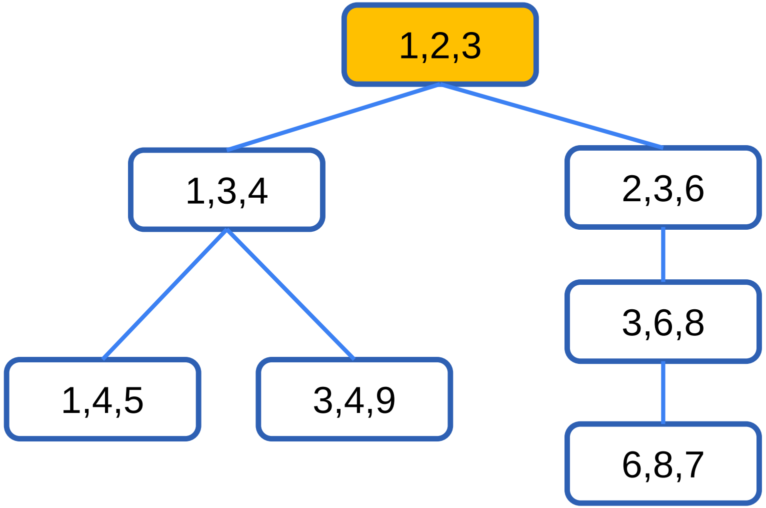
Cops and robbers



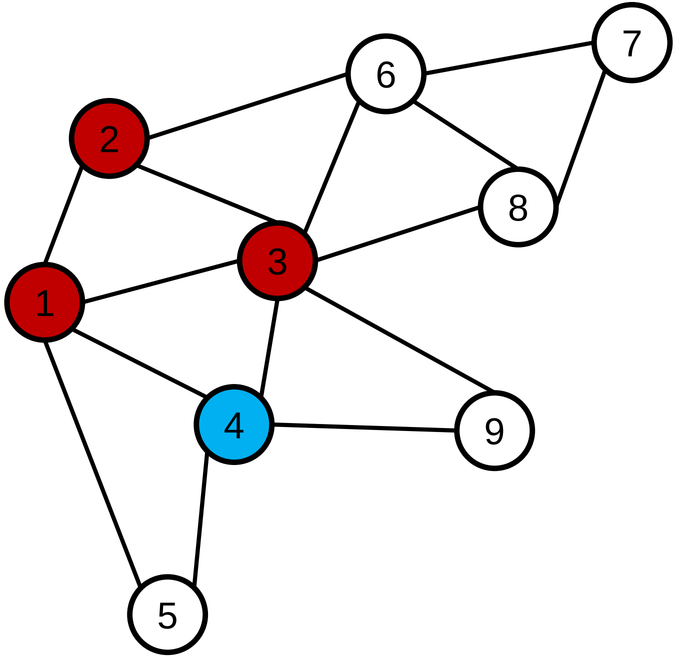


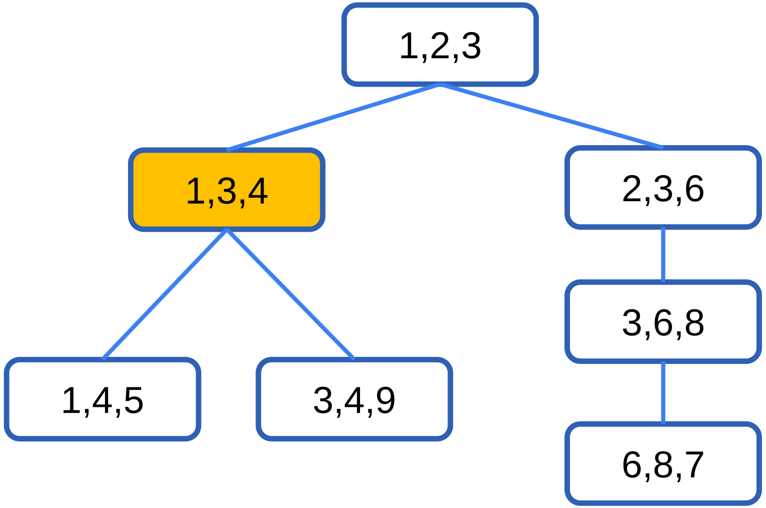
Cops and robbers



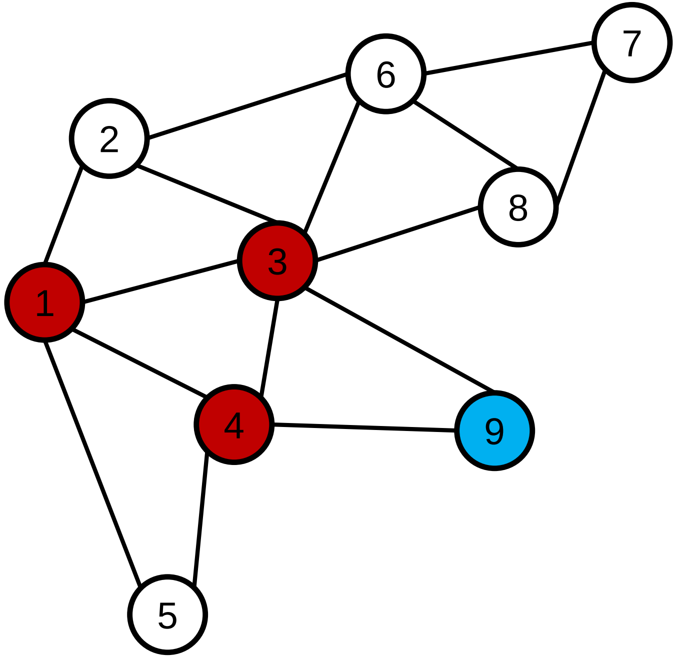


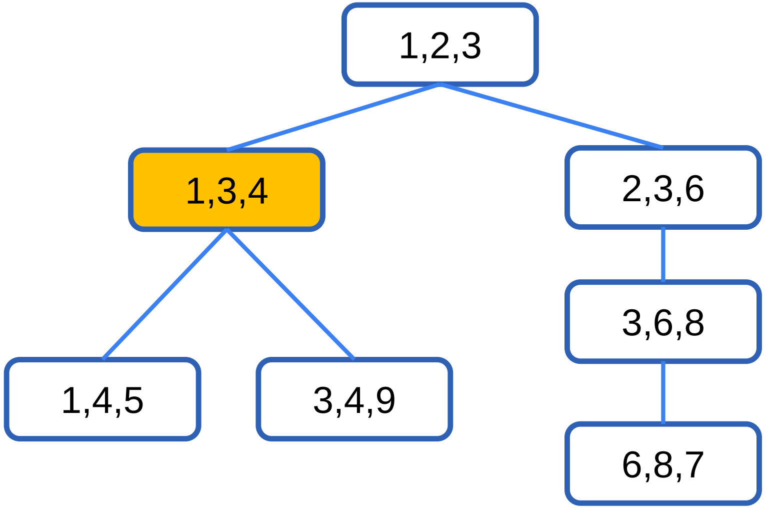
Cops and robbers



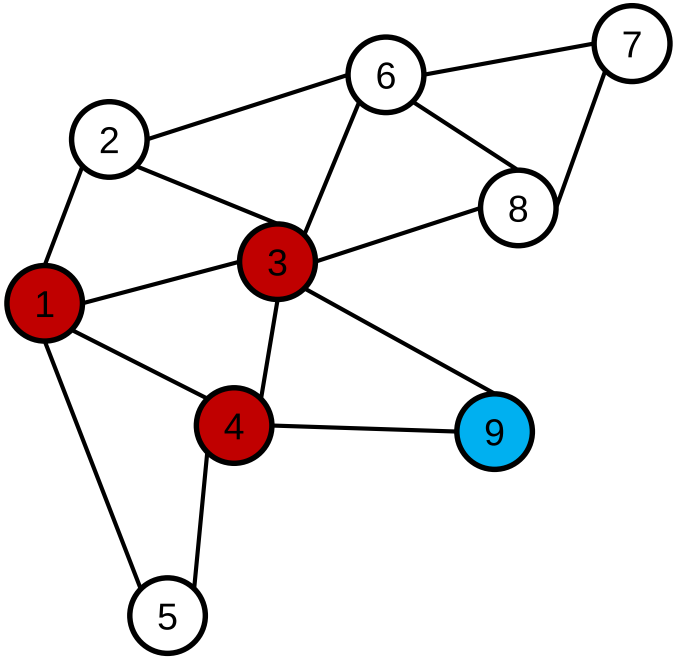


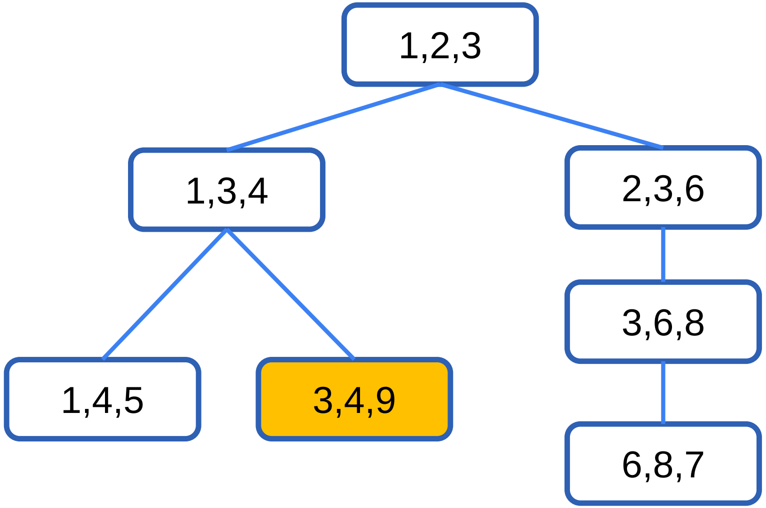
Cops and robbers



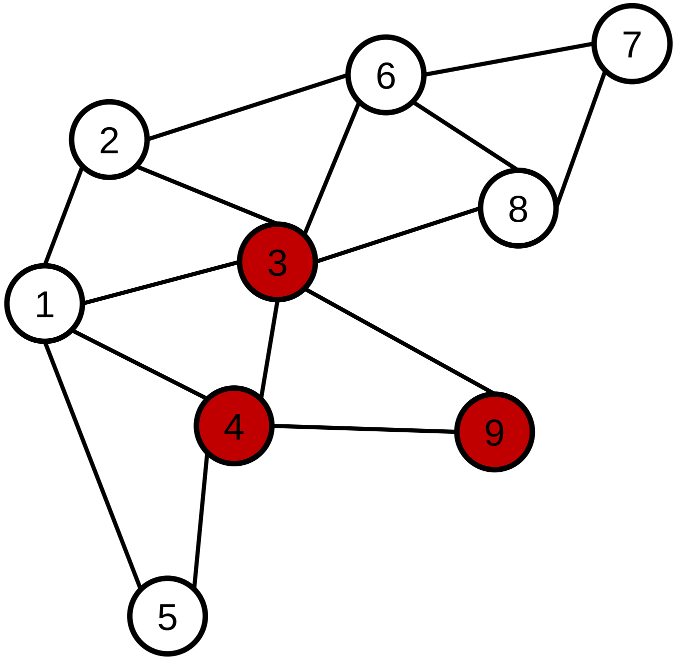


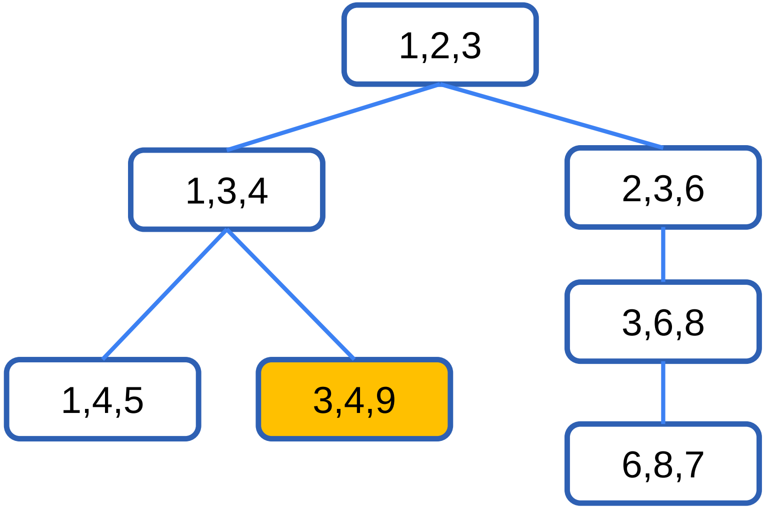
Cops and robbers



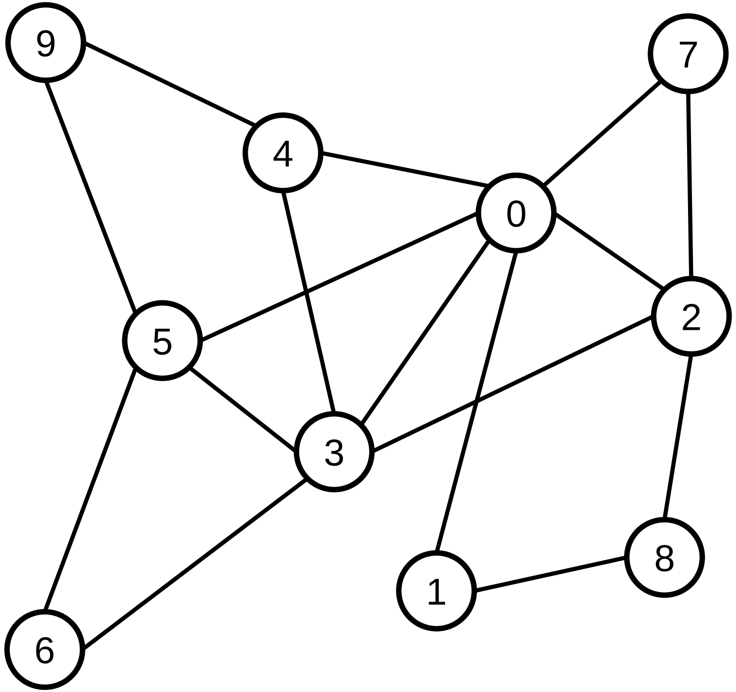


Cops and robbers



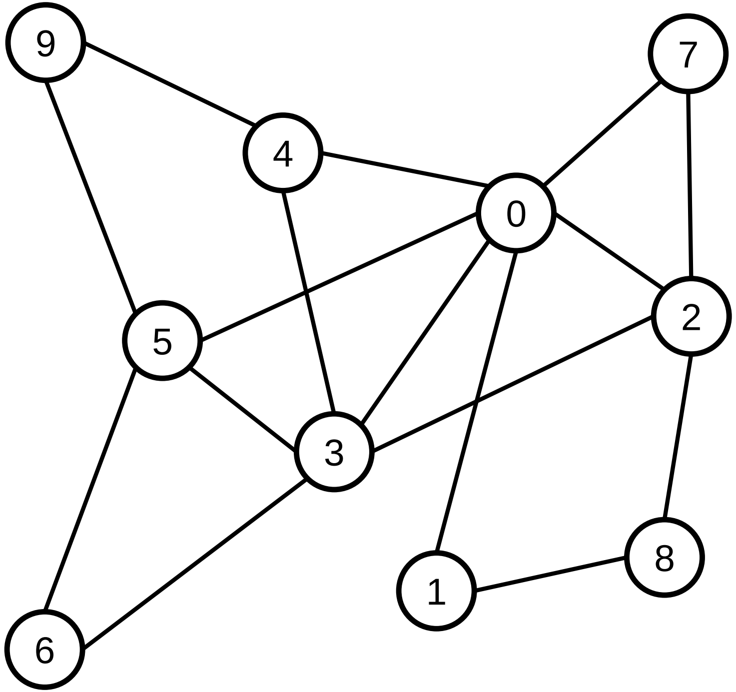


Computing a tree decomposition



Remove a node, triangulate its neighbours, and make a bag

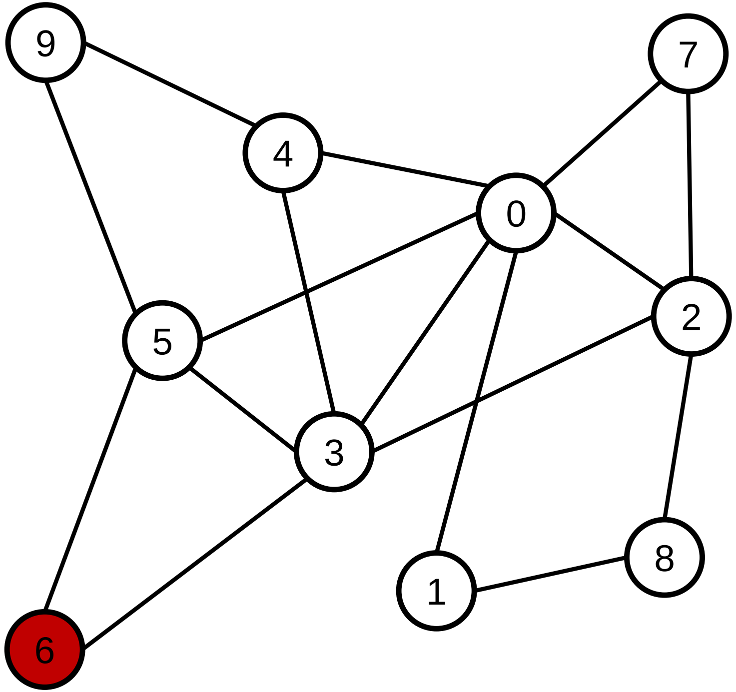
Computing a tree decomposition



removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

Remove a node, triangulate its neighbours, and make a bag

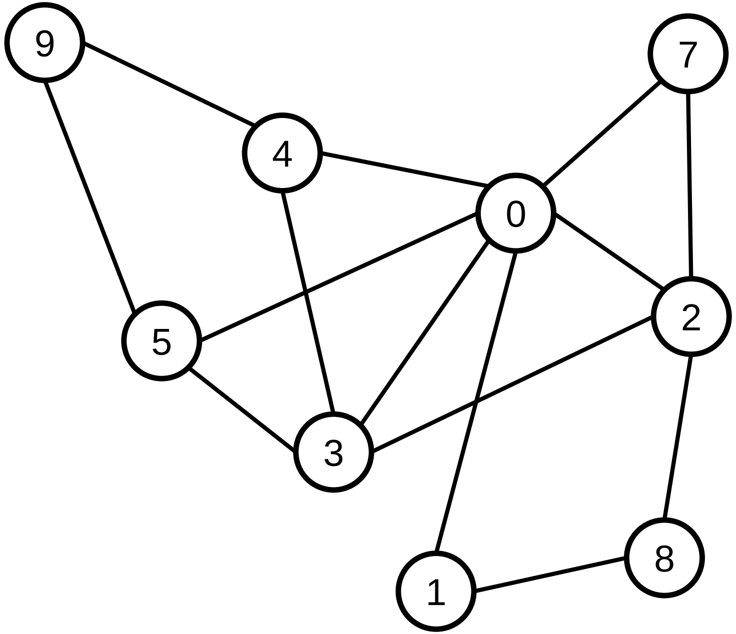
Computing a tree decomposition



removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

Remove a node, triangulate its neighbours, and make a bag

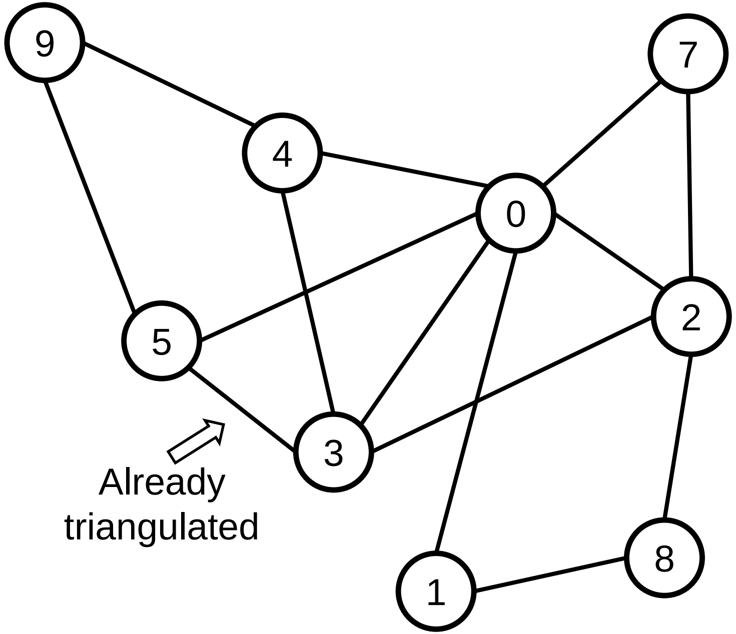
Computing a tree decomposition



removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

Remove a node, triangulate its neighbours, and make a bag

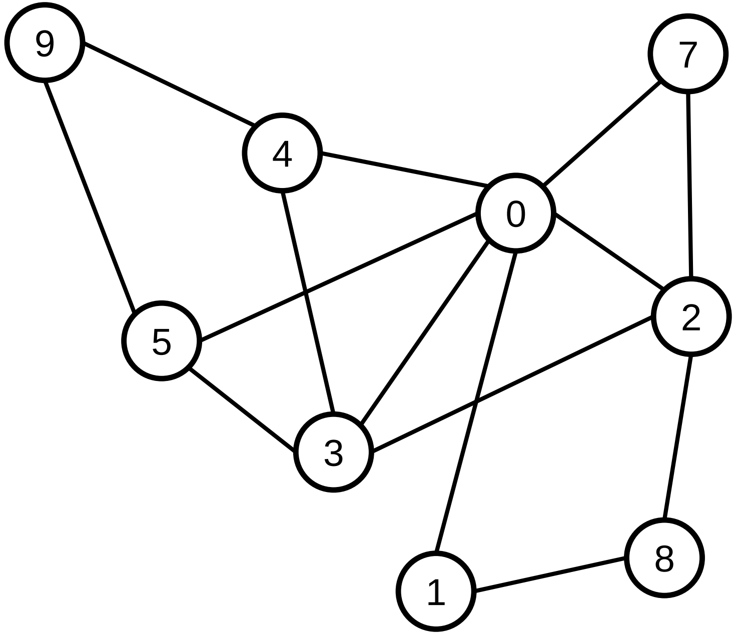
Computing a tree decomposition



removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

Remove a node, triangulate its neighbours, and make a bag

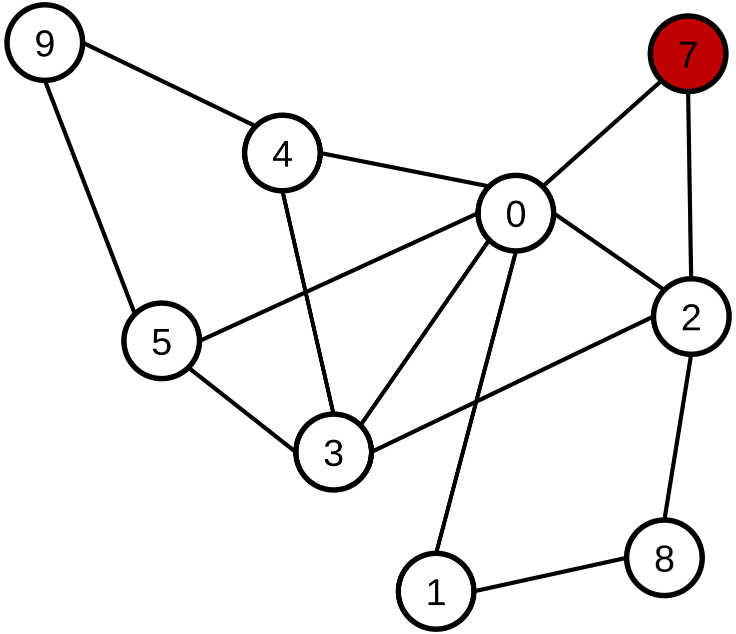
Computing a tree decomposition



removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

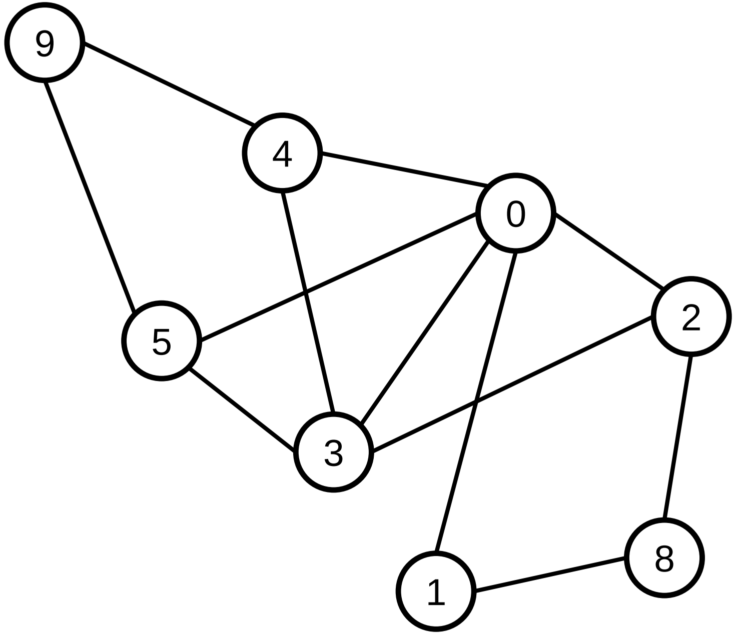


removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

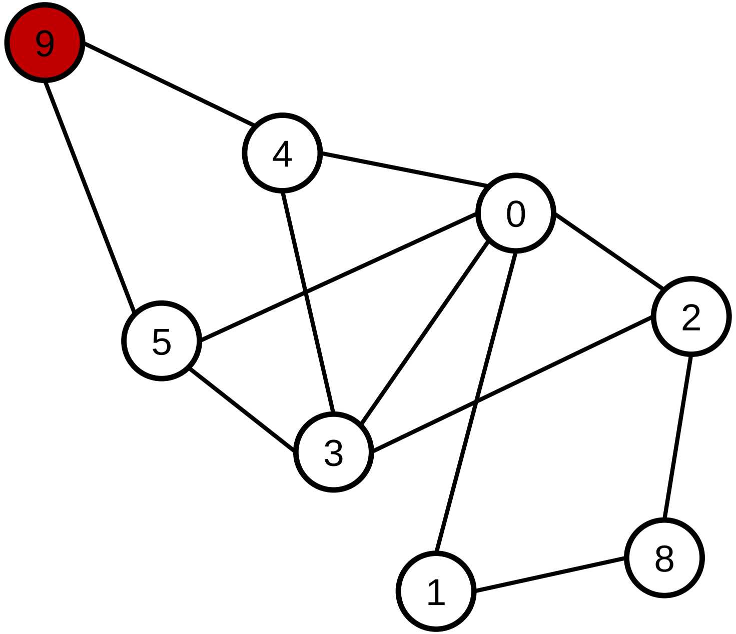
removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4



Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4



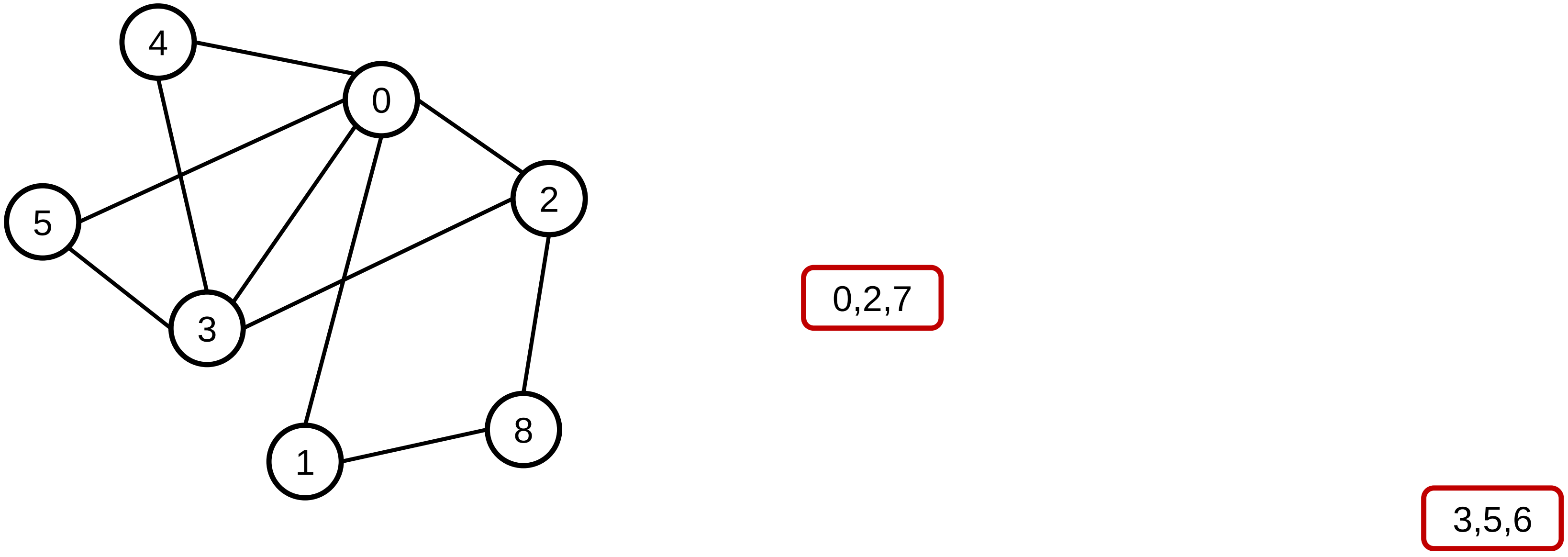
0,2,7

3,5,6

Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

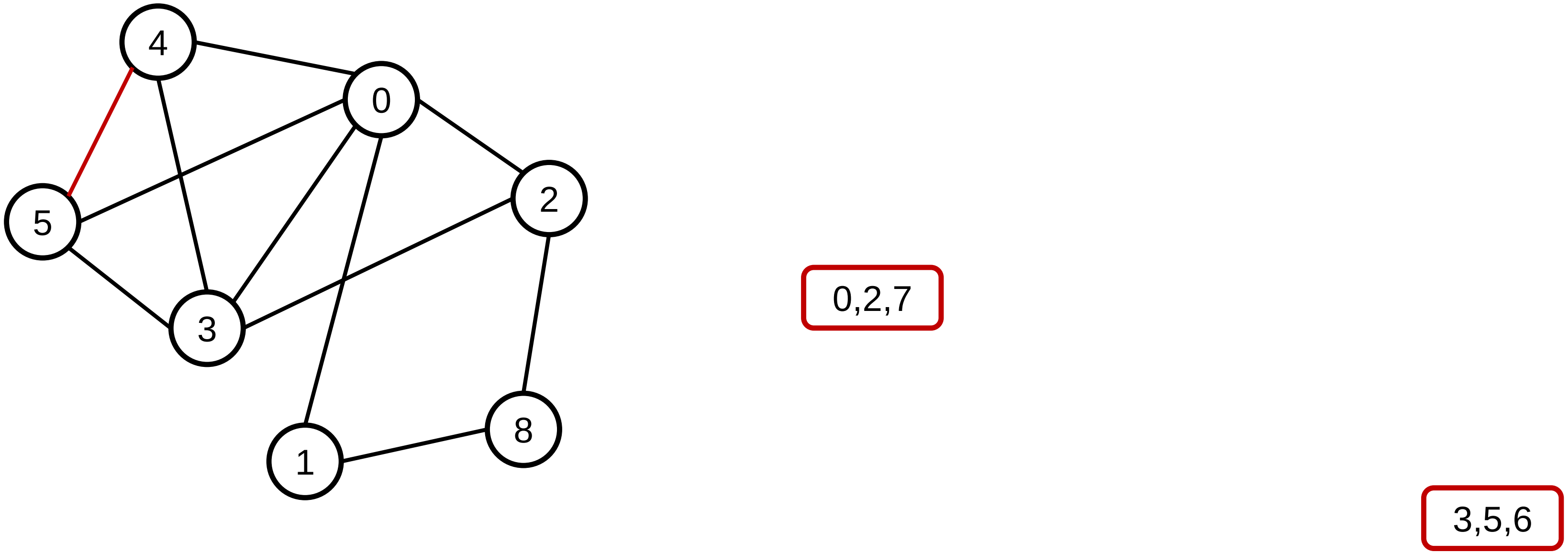
removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4



Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

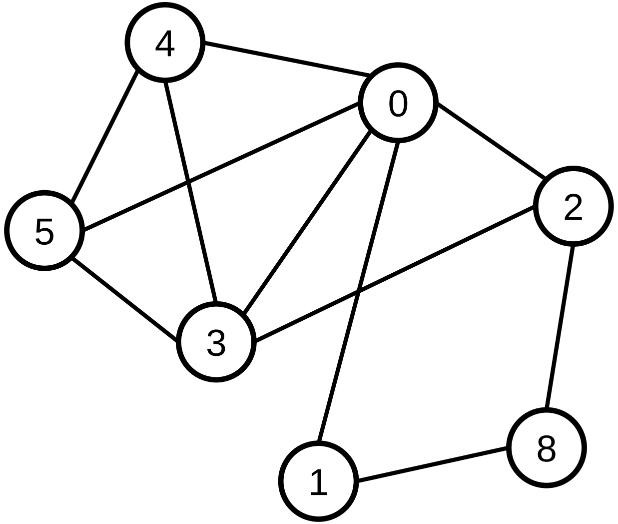
removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4



Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4



0,2,7

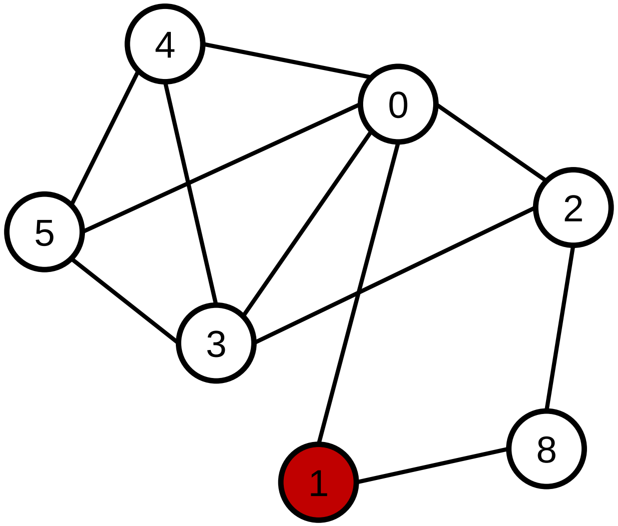
4,5,9

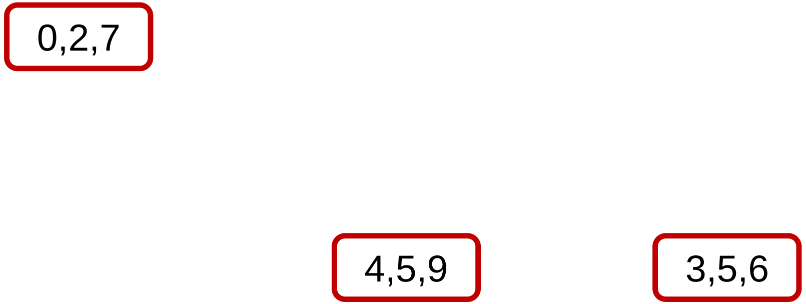
3,5,6

Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

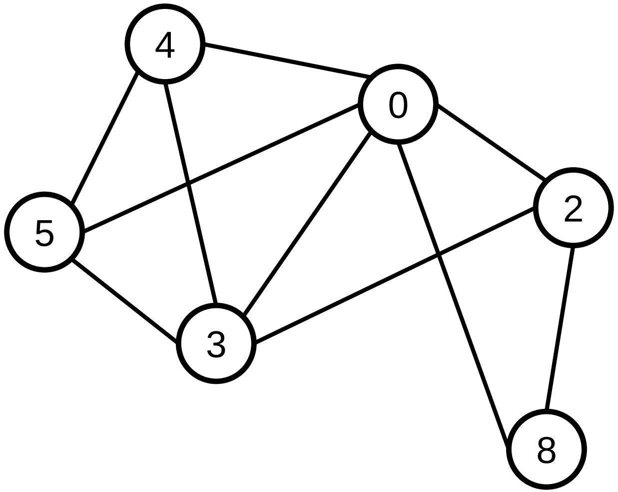


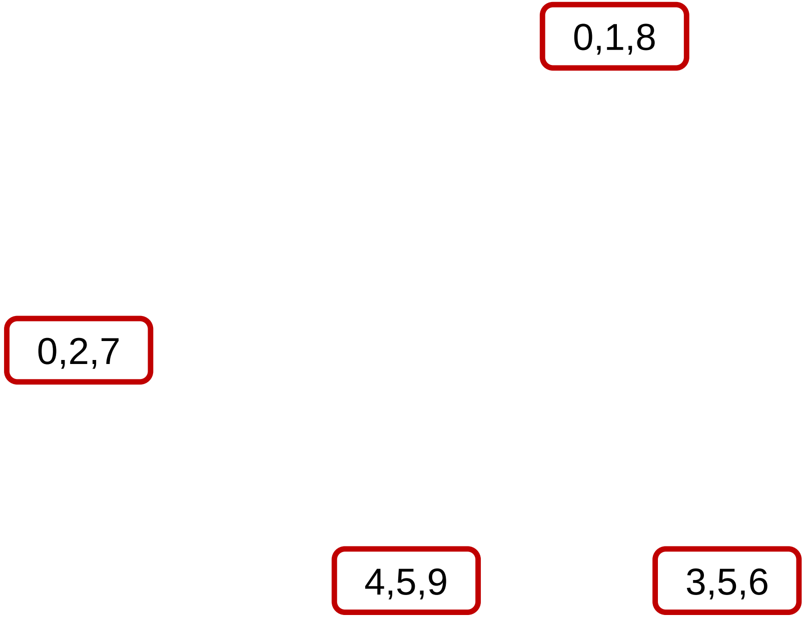


Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

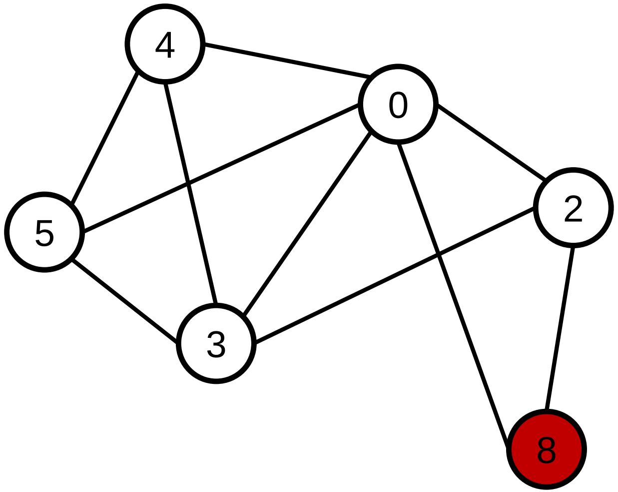


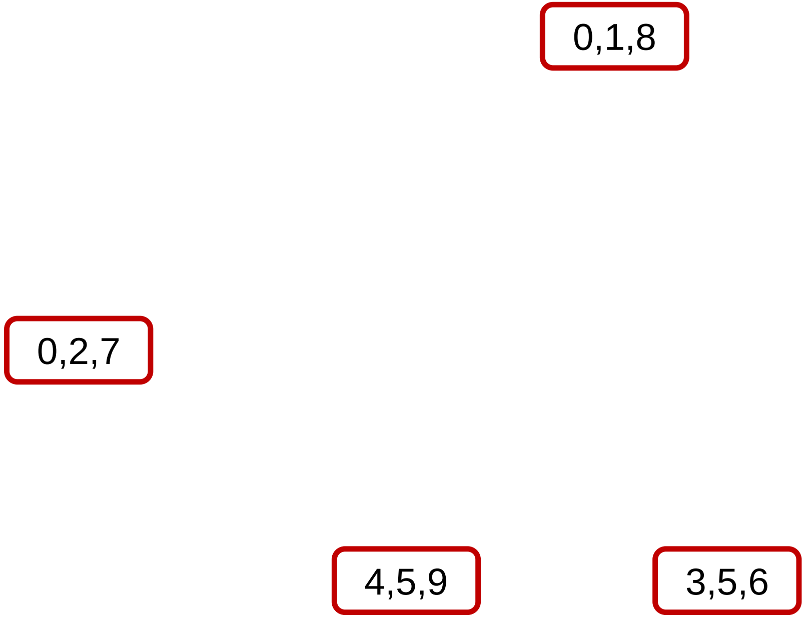


Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

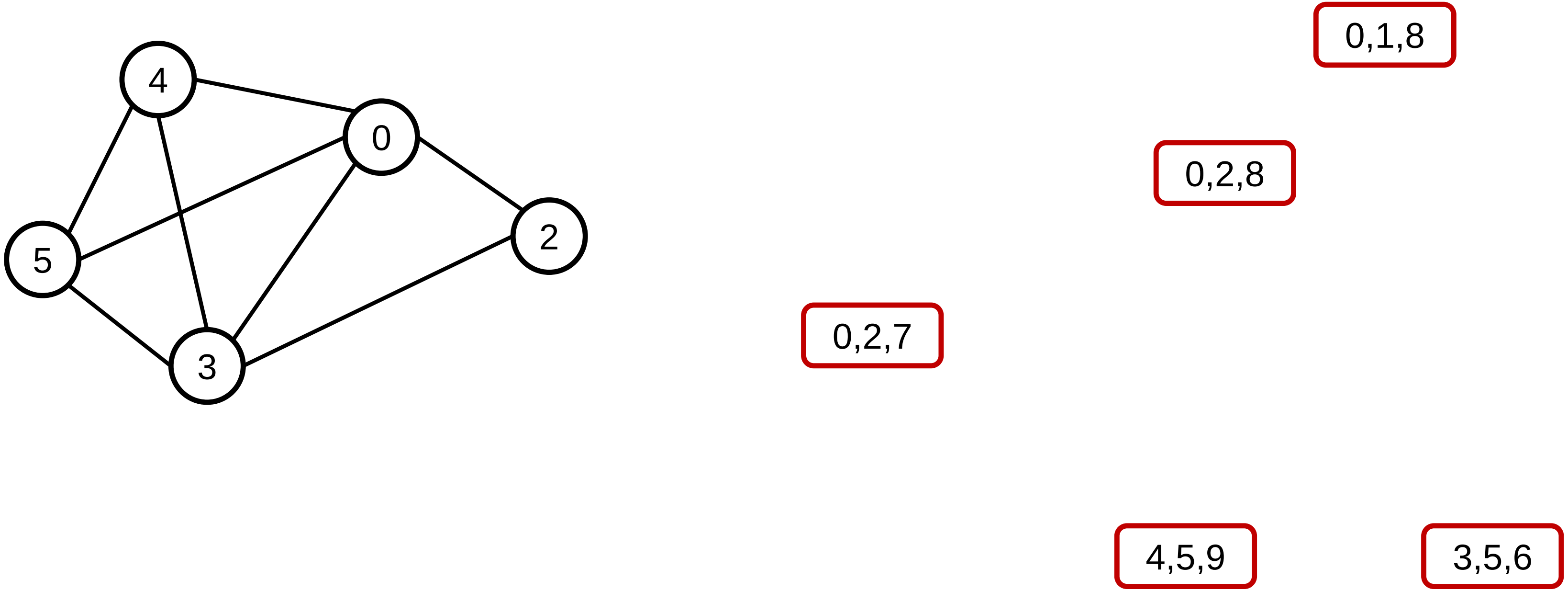




Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

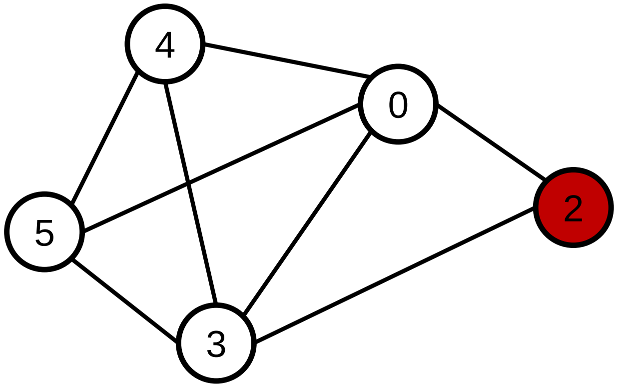
removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

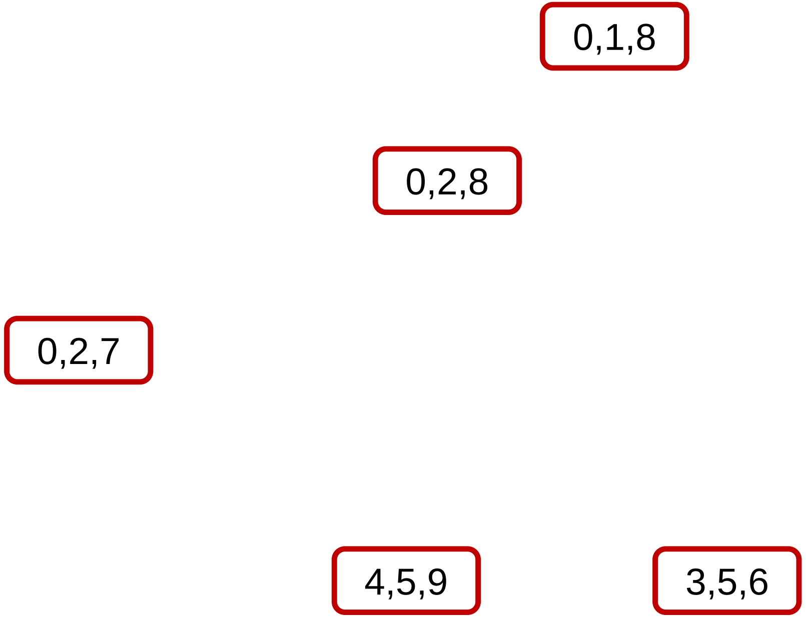


Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

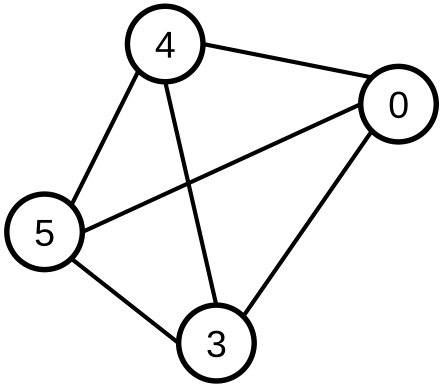


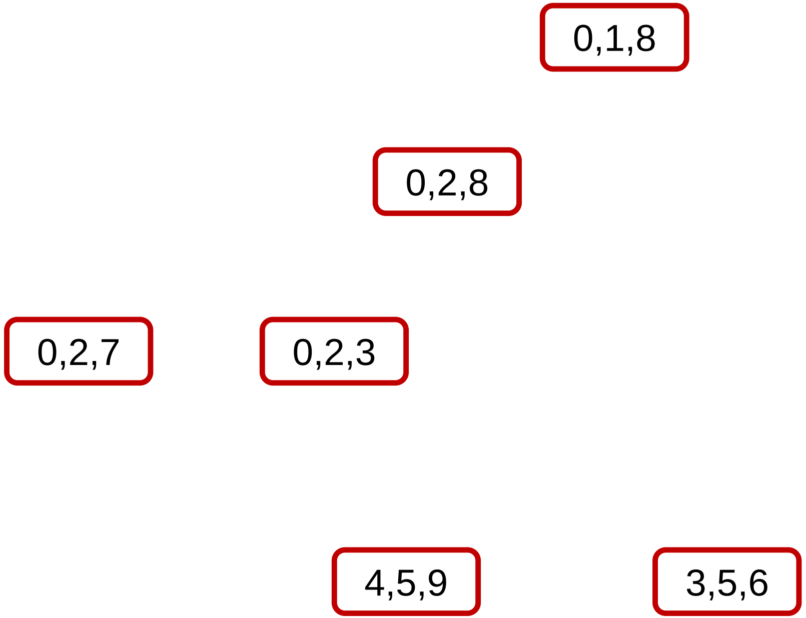


Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

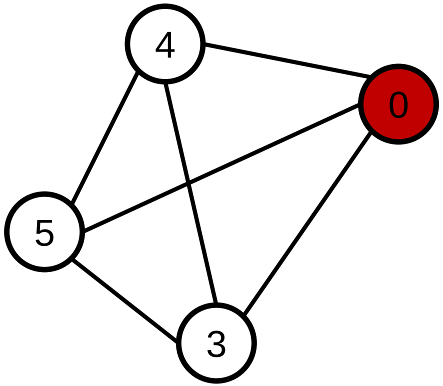


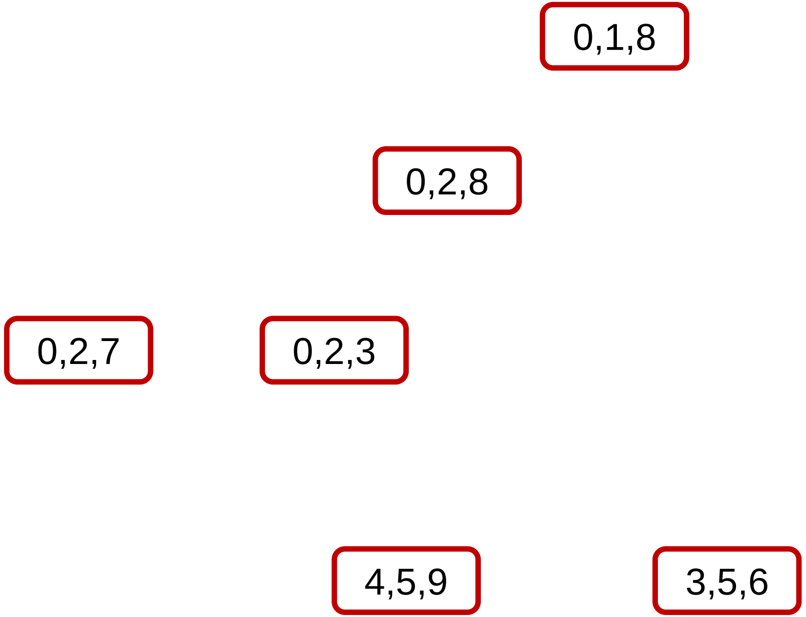


Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

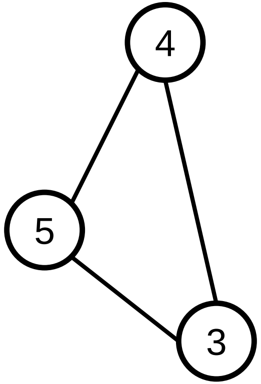


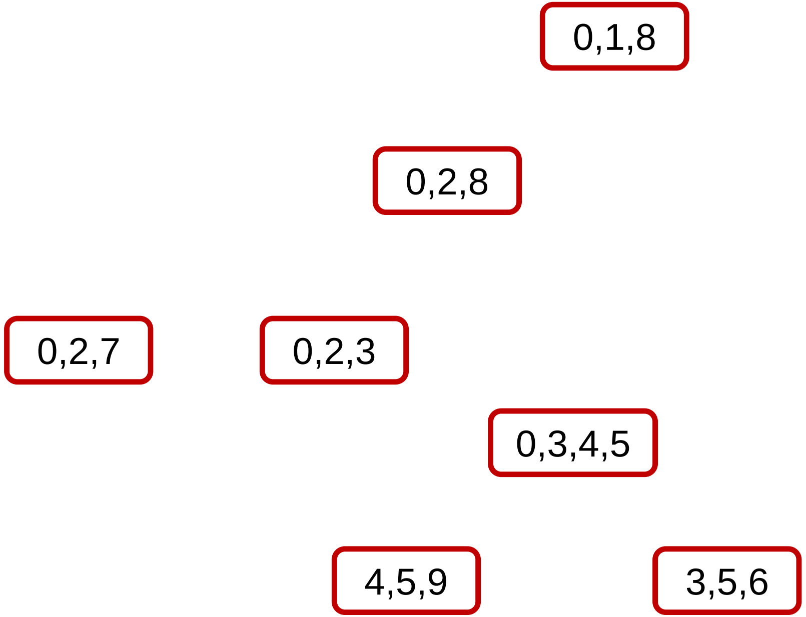


Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

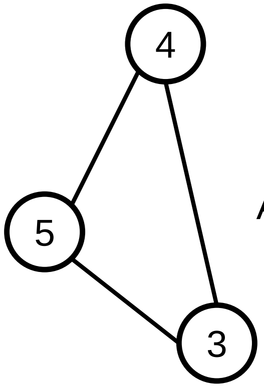




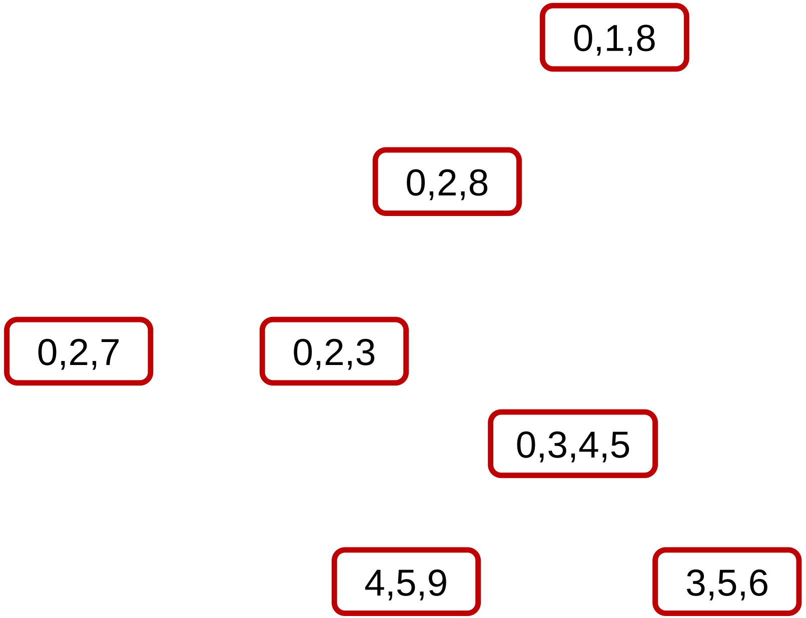
Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

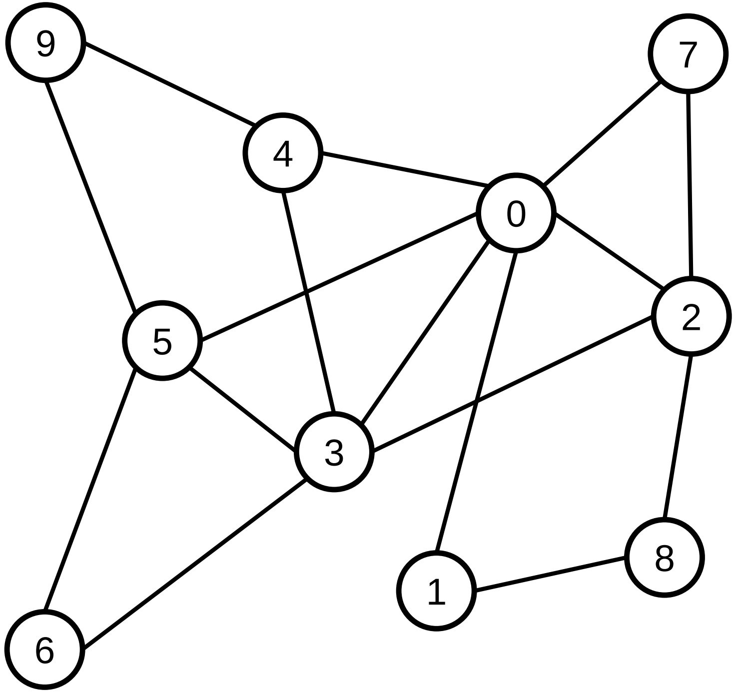


Already in bag
{0,3,4,5}

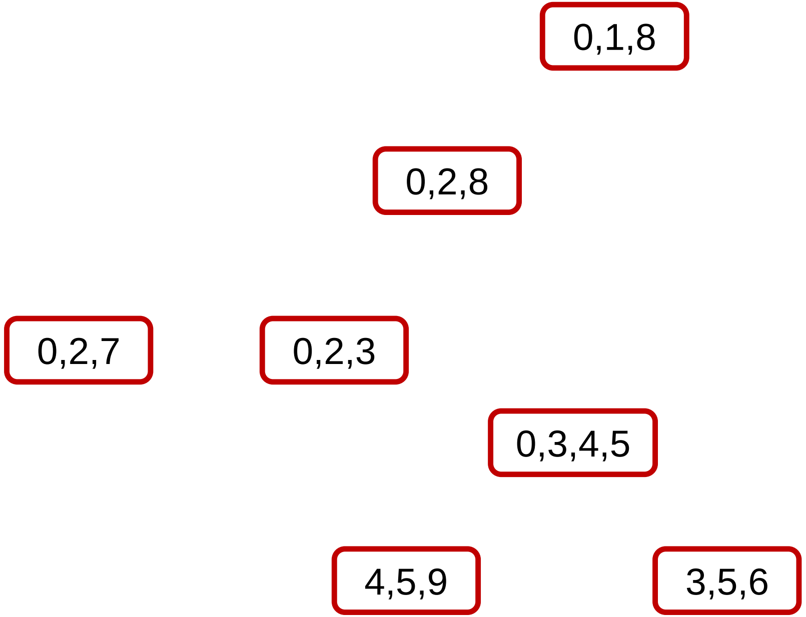


Remove a node, triangulate its neighbours, and make a bag

Computing a tree decomposition

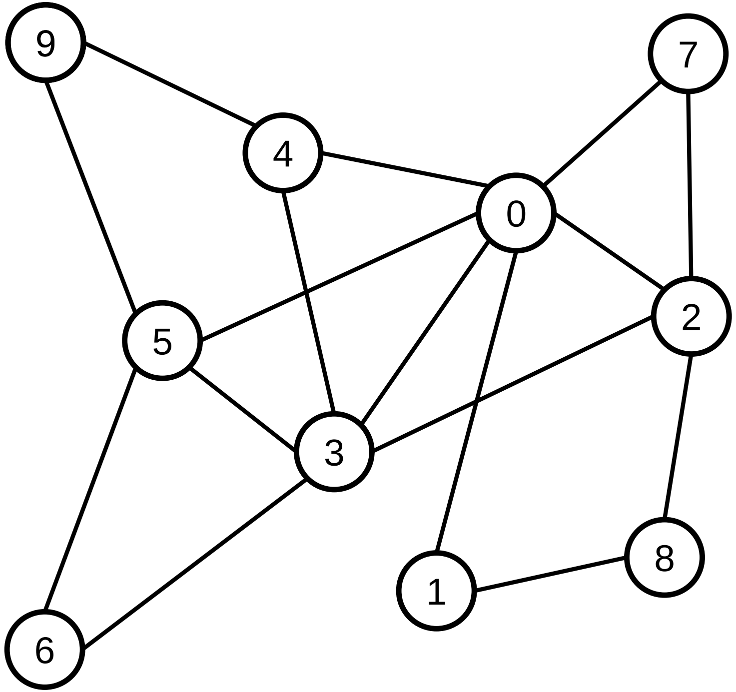


removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4

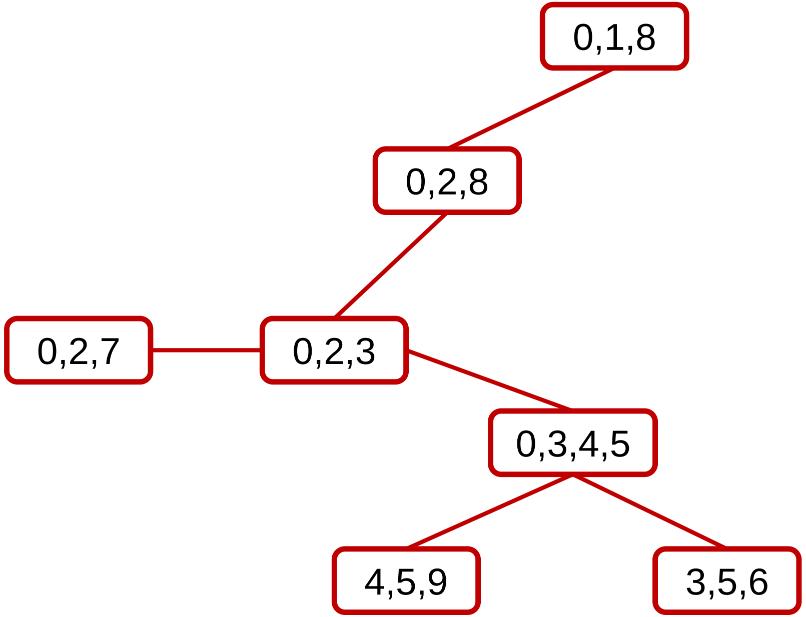


Connect each bag with the one for which the intersection is maximal

Computing a tree decomposition



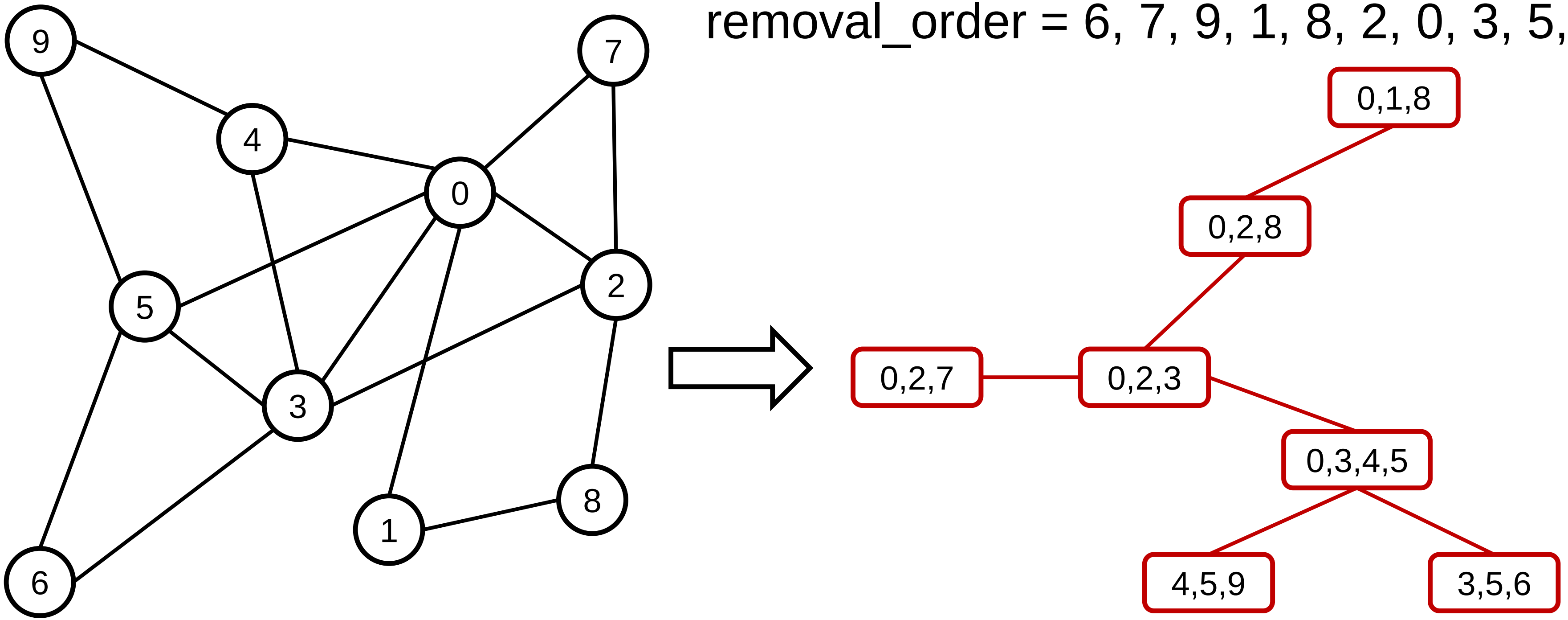
removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4



Connect each bag with the one for which the intersection is maximal

Computing a tree decomposition

removal_order = 6, 7, 9, 1, 8, 2, 0, 3, 5, 4



We have a tree decomposition!

Which node removal order?

- We assumed a node removal order
  - Results vary with the removal order
- Computing the treewidth (and the relative decomposition) is NP-Hard
  - $N!$ possible orderings
  - $O(2^n)$ with dynamic programming
- We use a heuristic which gives a good solution

Minimum degree heuristic

- When we remove a node to compute the tree decomposition, we remove the node with the minimum degree
- Since removing a node changes the degree of its neighbours, we compute the minimum degree every iteration