Graph Modification Problems

Let Π be a graph property. There are the following well-known graph modification problems for an input G:

1. Edge Deletion Problem: Can we obtain a graph in Π by deleting k edges from G?
2. Node Deletion Problem: Can we obtain a graph in Π by deleting k nodes from G?
3. Node/Edge Deletion Problem: Can we obtain a graph in Π by deleting k nodes and l edges from G?
4. Edge Insertion Problem: Can we obtain a graph in Π by inserting k edges in G?

Generalization

Definition

$\Pi_{i,j,k}$-Graph Modification Problem

Input: A graph $G = (V, E)$

Parameter: $i, j, k \in \mathbf{N}$

Question: Can we obtain a graph in Π by removing up to i nodes, removing up to j edges, and inserting up to k edges in G?

The Leizhen Cai Theorem

Theorem

Let Π be a graph property with a finite characterization by obstruction sets.

Then the $\Pi_{i,j,k}$-Graph Modification Problem can be solved in $O(N^{i+2j+2k}|G|^{N+1})$ steps and is thus fixed paramater tractable.

N is the number of nodes in the largest graph in the obstruction set, i.e., a constant.

Proof of the Leizhen Cai Theorem

Lemma

Let Π *be a hereditary graph property that can be checked in* $T(G)$ *steps.*

Then it takes $O(|V|T(G))$ *many steps to find a minimal forbidden induced subgraph for any* $G = (V, E) \notin \Pi$.

In this context, the term "minimal" refers to the order "induced subgraph".

Proof of the Leizhen Cai Theorem

Proof of the lemma:

Let $V = \{v_1, \ldots, v_n\}$.

$H := G$
for $i = 1, \ldots, n$ **do**
  if $H - \{v_i\} \notin \Pi$ **then** $H := H - \{v_i\}$
od

Upon termination, H is a minimal forbidden induced subgraph.

Proof of the Leizhen Cai Theorem

Input: $G = (V, E)$

Parameter: $i, j, k \in \mathbf{N}$

Question: $G \in \Pi_{i,j,k}$

while $i + j + k > 0$ **do**
  $H :=$ minimal forbidden induced subgraph of G
  Modify G by removing an edge or node or by inserting an edge from/in H
  Let $i := i - 1$, $j := j - 1$, or $k := k - 1$.
  if $G \in \Pi$ **then** answer YES
od

Answer NO

Proof of the Leizhen Cai Theorem

Running time:

Find H: $O(|V| \cdot |V|^N)$ according to the lemma

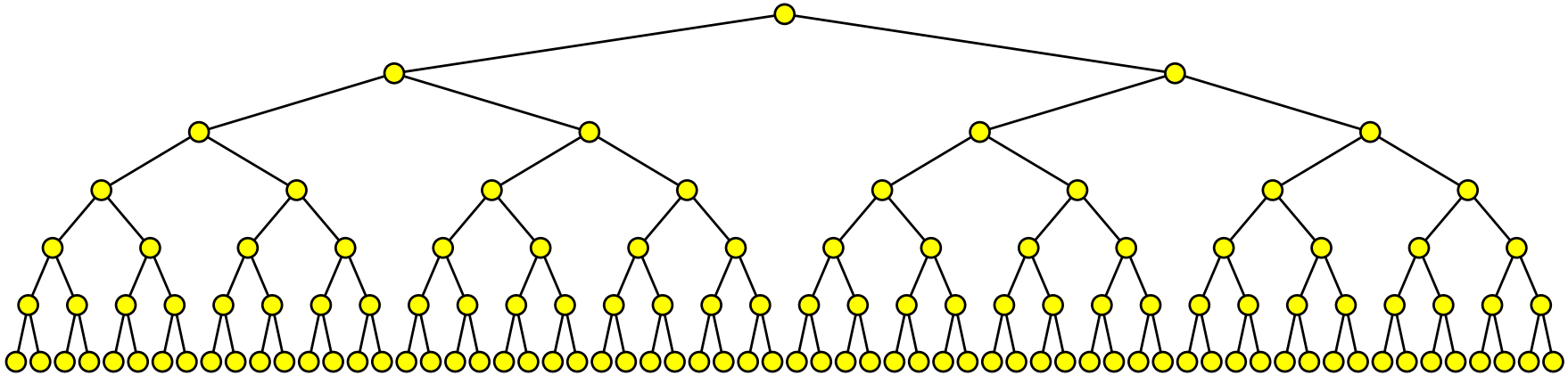
There are only N ways of removing a node from H

There are only $\binom{N}{2}$ ways of deleting or inserting an edge from/in H

Total running time

$$O(N^{i+2j+2k}|V|^{N+1}).$$

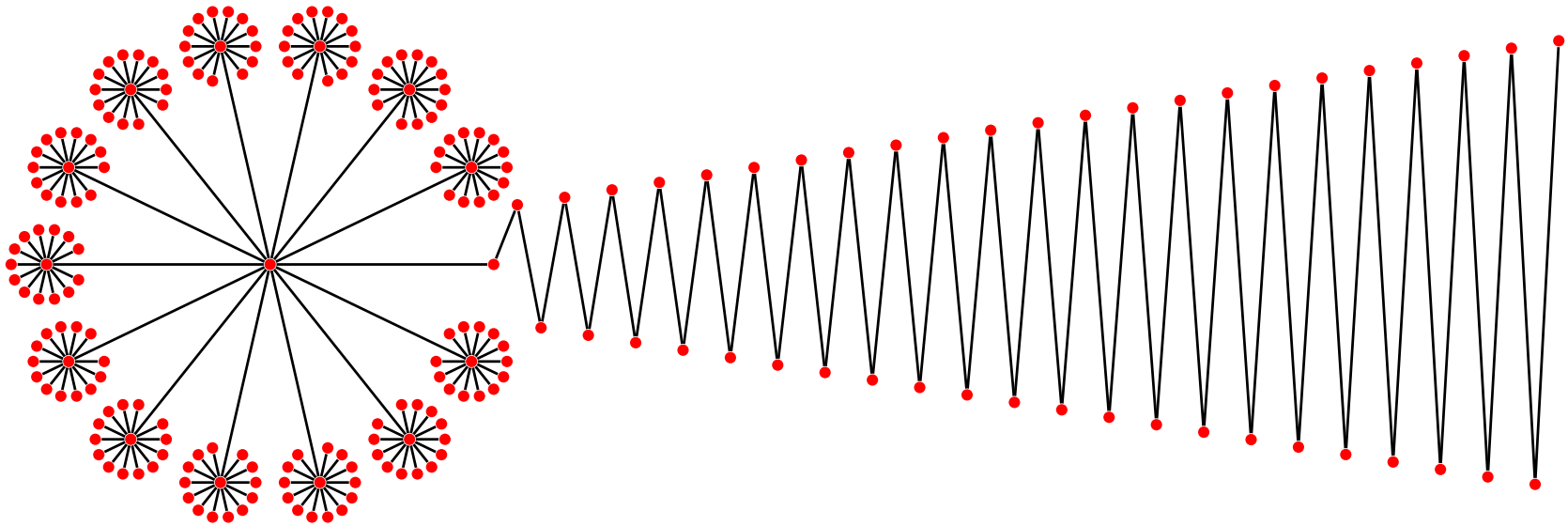
Interleaving — Search Trees and Problem Kernels



In a search tree, the parameter decreases as it approaches the leaves, whereas this is not necessarily the case for the size of the instance (which could even grow).

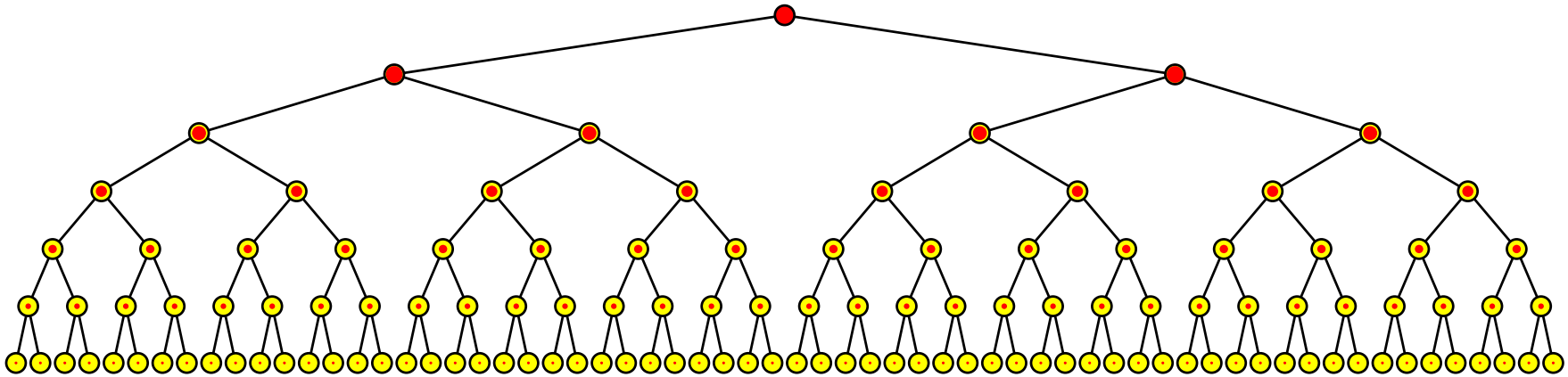
If the expansion of a node in the search tree takes $p(n)$ steps, then the total running time becomes $O(s(k, n)p(n))$, where $s(k, n)$ denotes the size of the search tree.

Interleaving



The size of this graph never drops below $n/2$ within the entire search tree of a vertex cover algorithm, provided that no reduction to the problem kernel is performed.

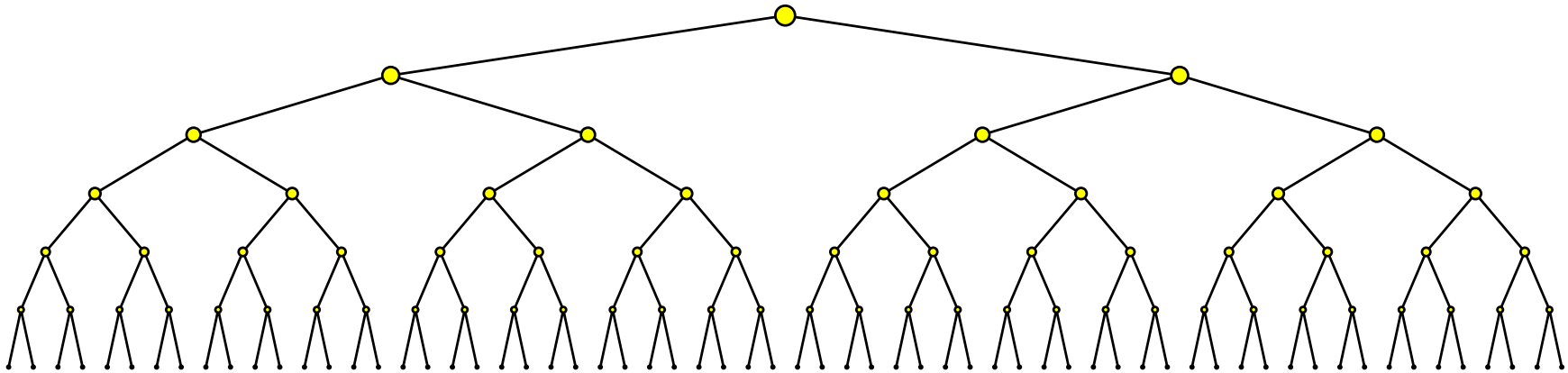
Interleaving



The size of the parameter is reflected by the size of the red dots. The recursion stops when the parameter is small. In the leaves, the parameter is bounded by a constant.

Nearly all nodes are close to a leaf $\Longrightarrow$ nearly all nodes have a small parameter.

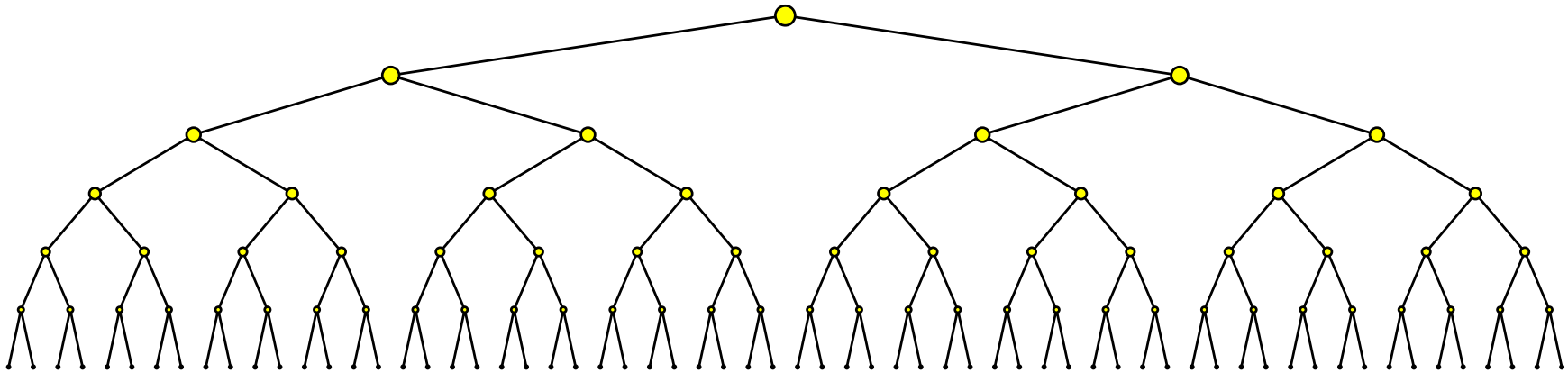
Interleaving



If we perform a reduction to problem kernel after each expansion of a node in the search tree, then the instances decrease in size as we approach the leaves.

A more detailed analysis reveals that the total running time is only $O(s(n,k) + t(n) + r(n))$ rather than $O(s(n,k)t(n))$, where $r(n)$ denotes the time required for the reduction to problem kernel.

Search Trees and Dynamic Programming



If all nodes are close to leaves and there are many of them, then some must be identical.

$\Longrightarrow$ We can improve the running time by computing the respective solutions only once and storing them in a database.

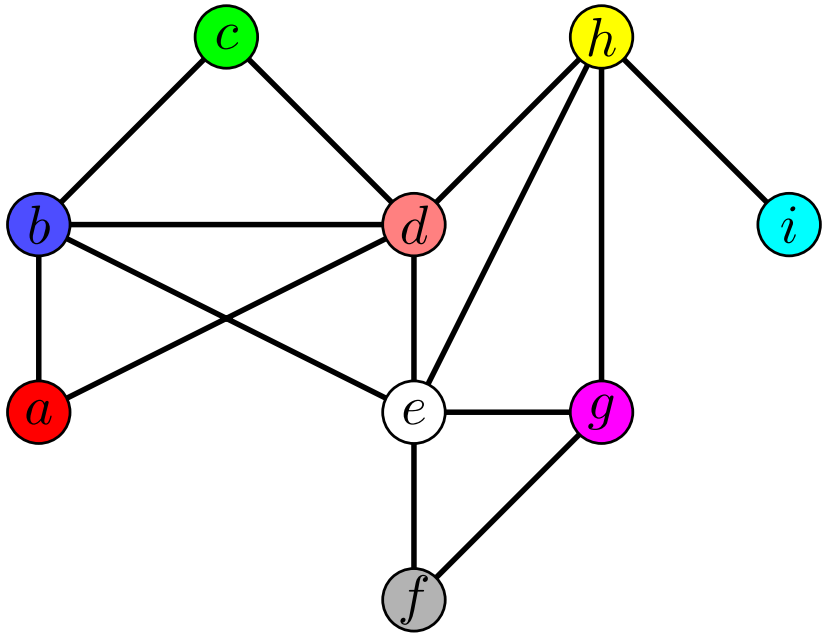
Search Trees and Dynamic Programming

Example: Vertex Cover and the 2^k algorithm

- Each node of the search tree is an induced subgraph.
- After a reduction to problem kernel, the size of a graph is bounded by $2k'$ from above if we are looking for a solution of size k'.
- There are at most $O\left(\binom{2k}{2k'}\right)$ induced subgraphs of size at most $2k'$.
- The running time is $O\left(2^{k-k'}\binom{2k}{2k'}\right)$ if we store solutions of size $2k'$ in the database.
- If we choose the value of k' optimally, the running time becomes 1.886^k.

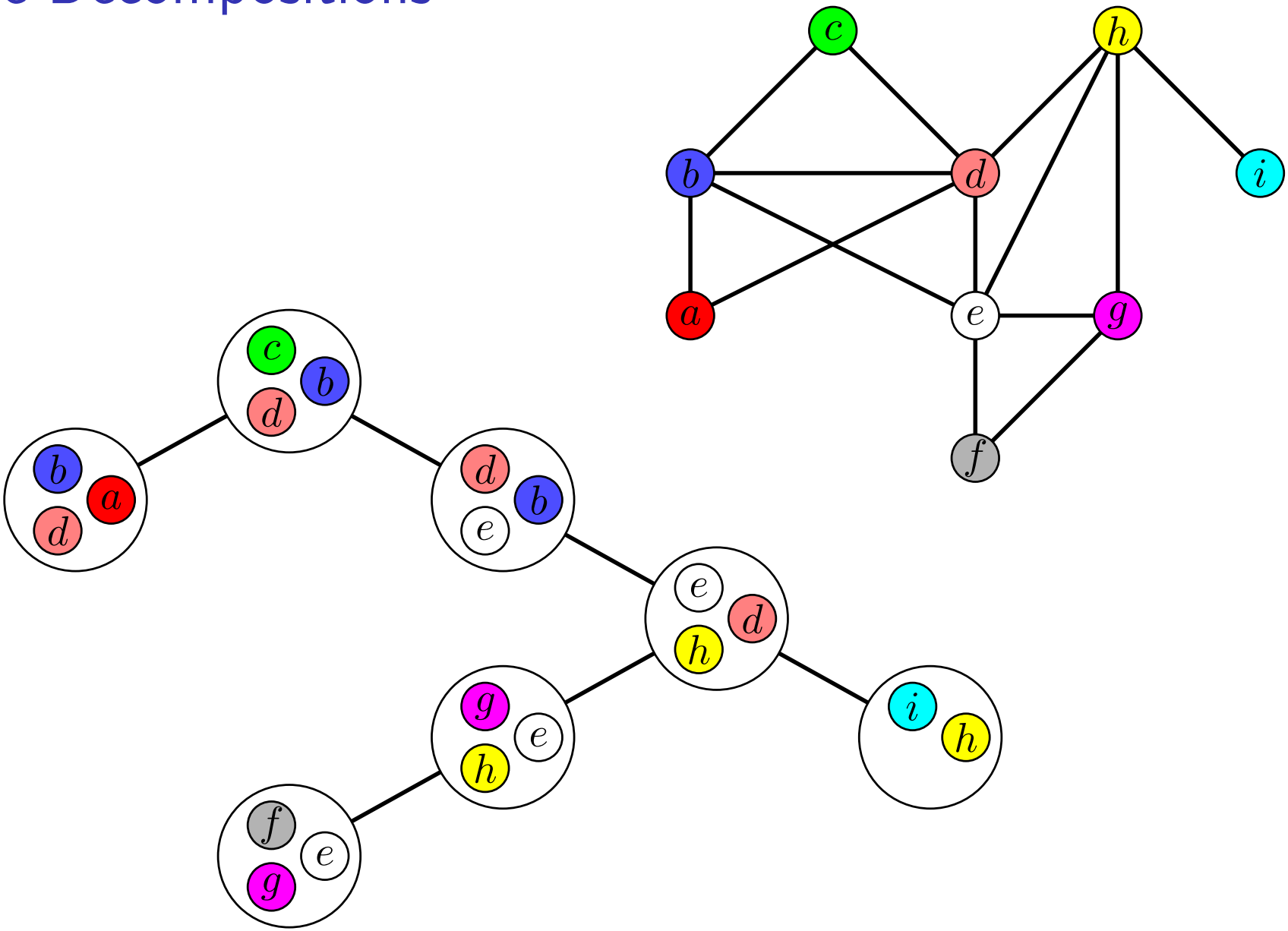
Tree Decompositions

A tree decomposition of a graph G is a tree, whose nodes are called bags. Every bag is a set of nodes from G.

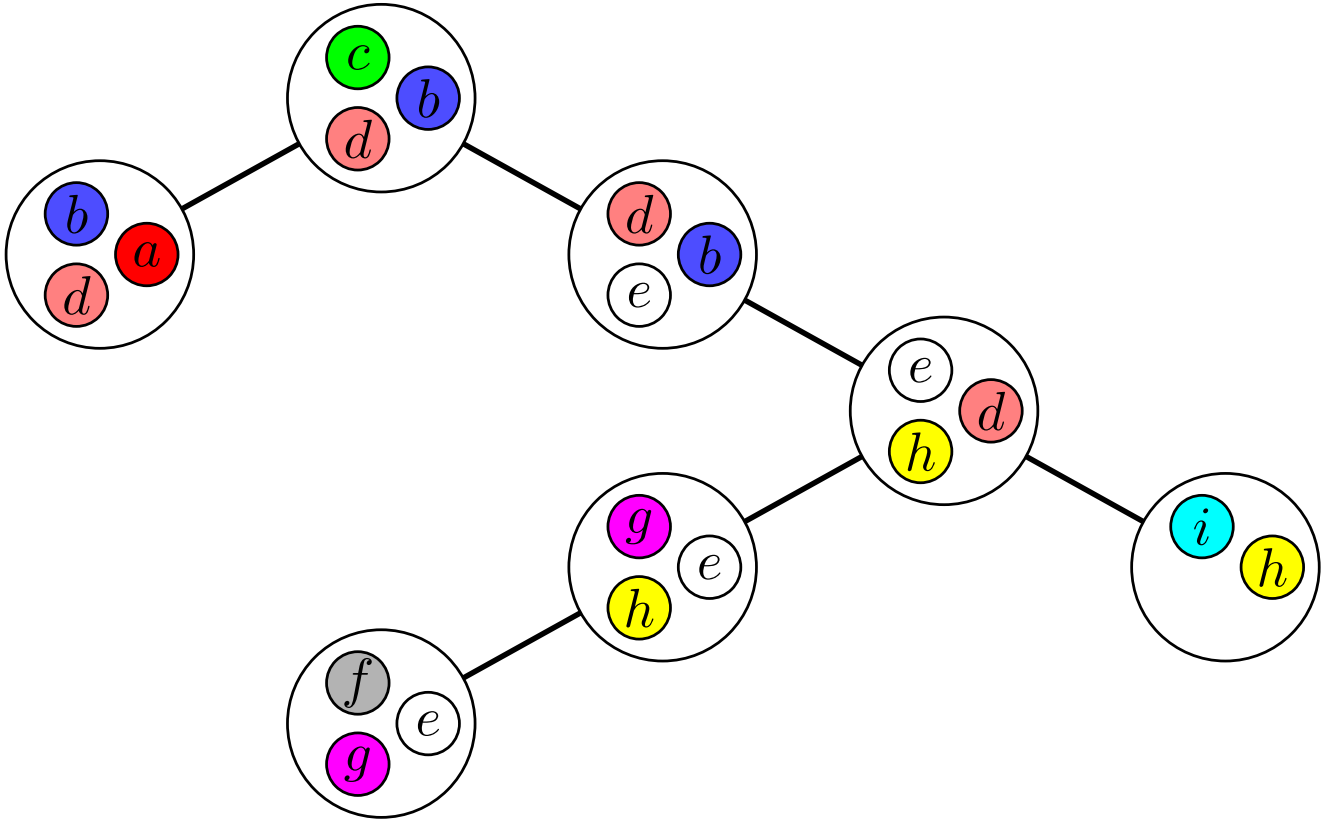


- Any node and any edge from G ist contained in at least one bag.
- A node contained in two bags A, B must be contained in any bag between A and B.

Tree Decompositions



Tree Decompositions and Treewidth

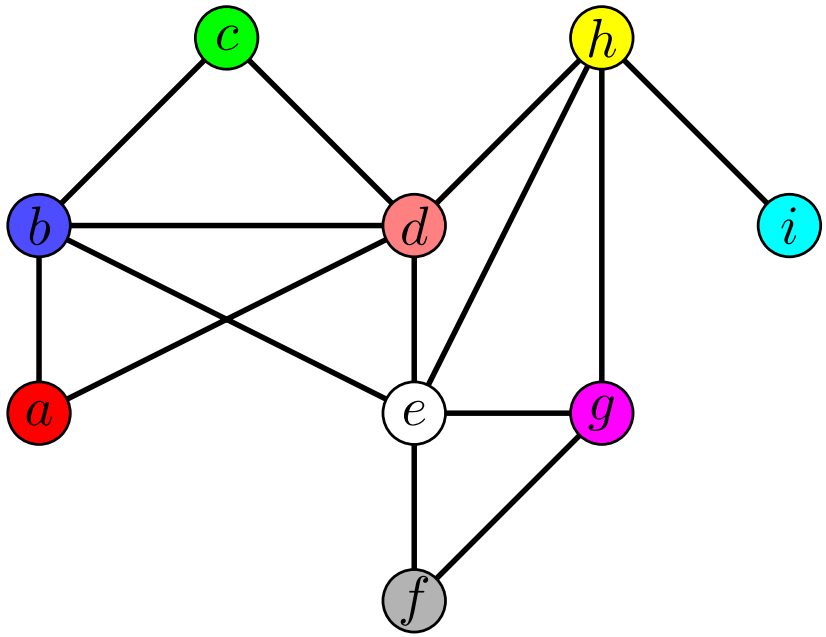


The width of a tree decompostion is the size of the largest bag minus 1.

$\Rightarrow$ Here, the treewidth is 2.

Tree Decompositions and Treewidth

Alternative definition:



The treewidth of G is the minimum number of cops, needed to catch a robber in G, minus 1.