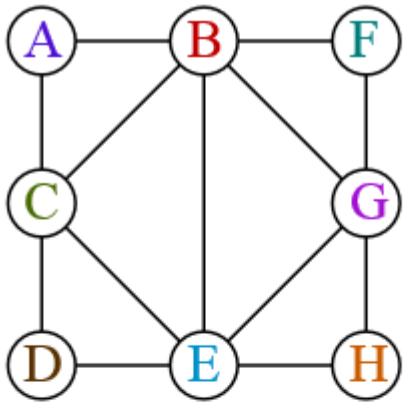
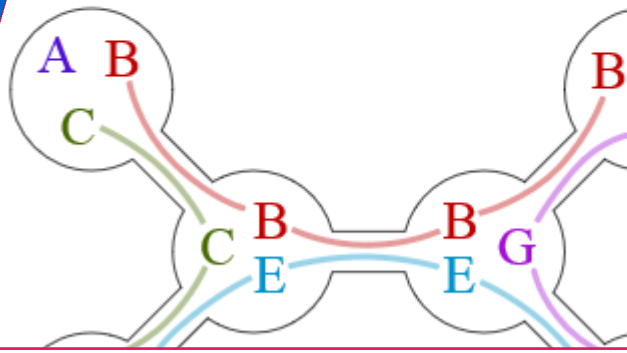
NET WORKS

Structural graph parameters

PART 1: TREEWIDTH





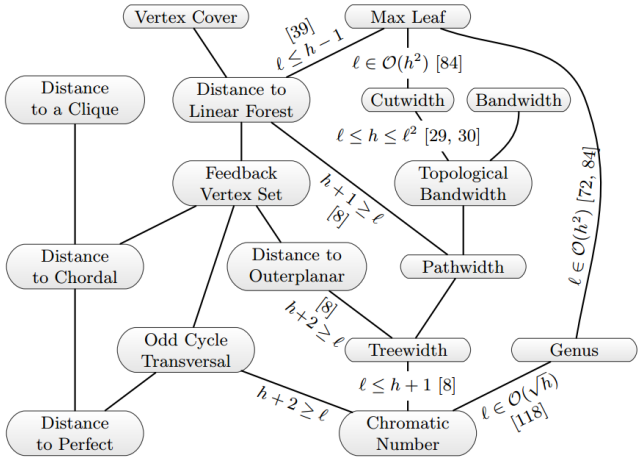
Bart M. P. Jansen (edited by mjd -2021)

TU/e

Networks Training Week 2017
January 31st – February 2nd 2017, Doorn, Netherlands

The goal for this micro-course

1. Introduce & motivate structural parameters of graphs
   – *Different ways of measuring the complexity of a graph*
2. Relate these parameters to each other



3. Use this hierarchy of parameters as a methodology for

   generating & answering meaningful research problems

The plan for this micro-course

Monday:

Introduction to graph parameters & exploring treewidth

Wednesday:

More parameters & relations between them

Thursday:

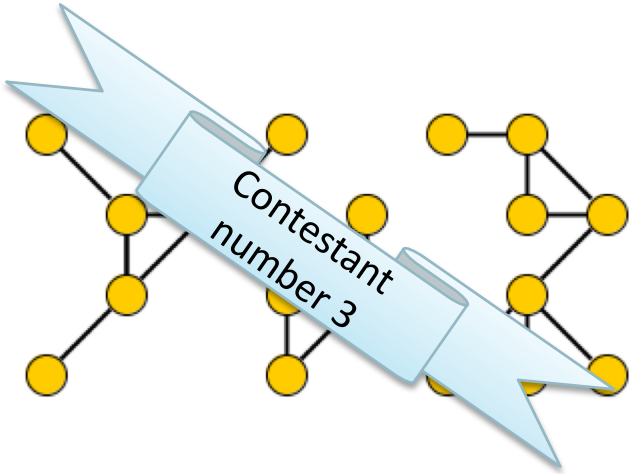
Applications to combinatorics & algorithmics

To get all noses in the same direction

All my graphs are finite

- … simple
- … and undirected (edge uv = edge vu)

Interrupt me any time!

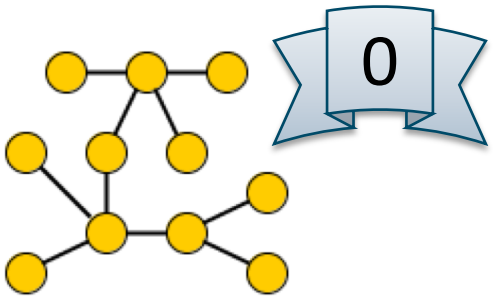


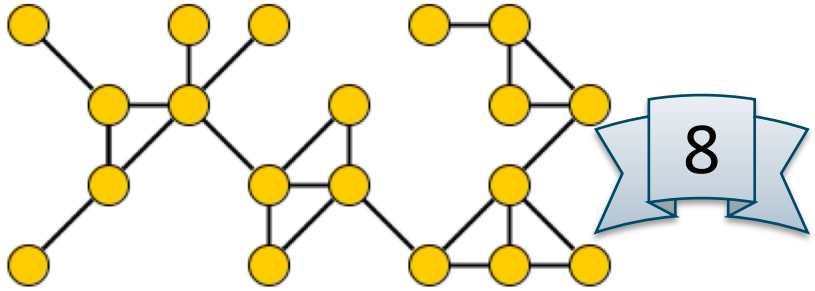
A ~~BEAUTY~~ CONTEST FOR GRAPHS

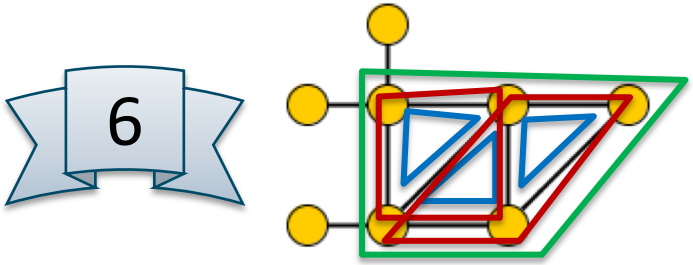
complexity

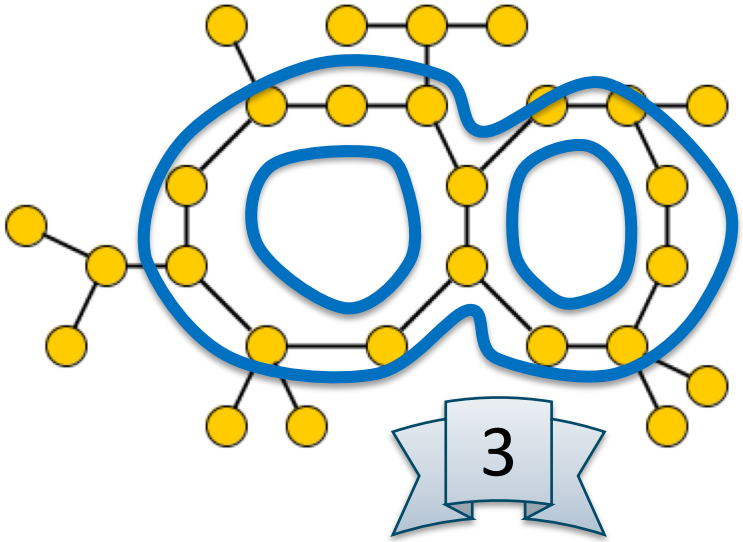
A complexity contest for graphs – Judge #1

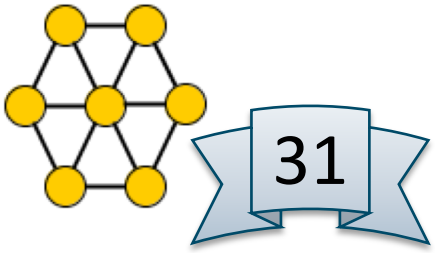
Number of cycles





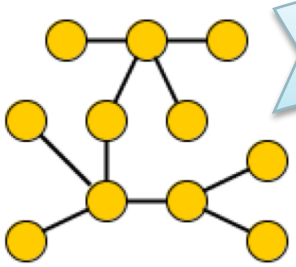




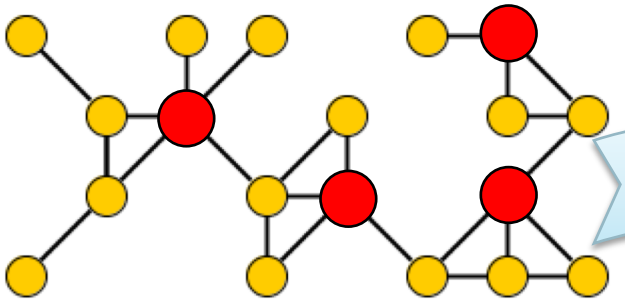


A complexity contest for graphs – Judge #2

Vertex-deletion distance to acyclic

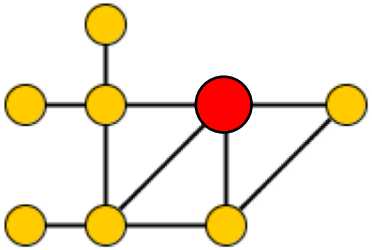


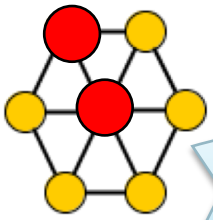
0



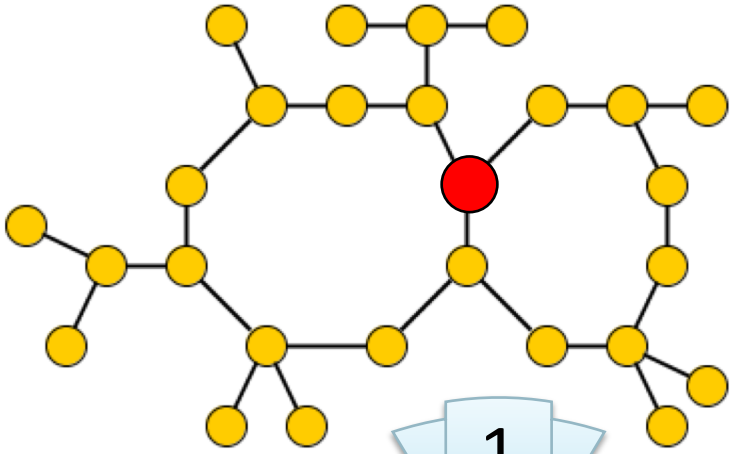
4

1





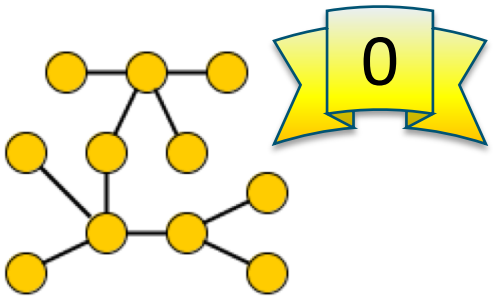
2

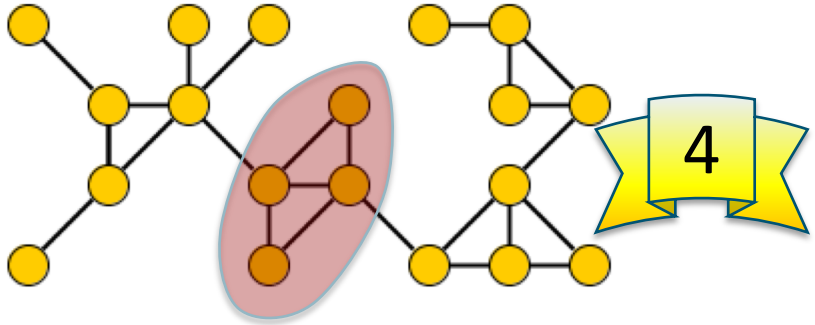


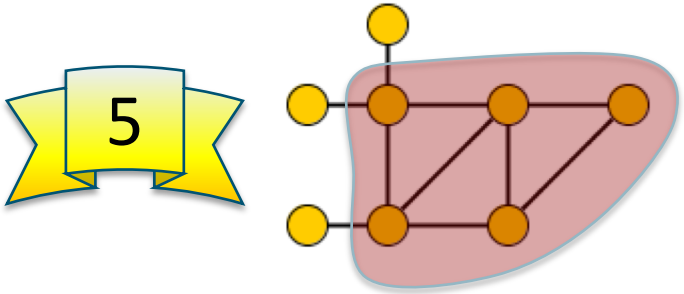
1

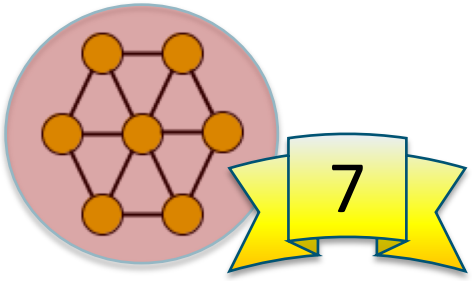
A complexity contest for graphs – Judge #3

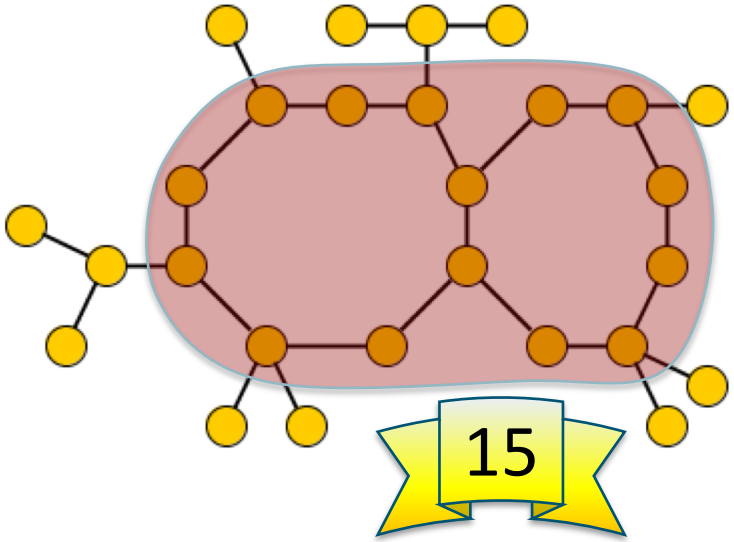
Length of longest cycle



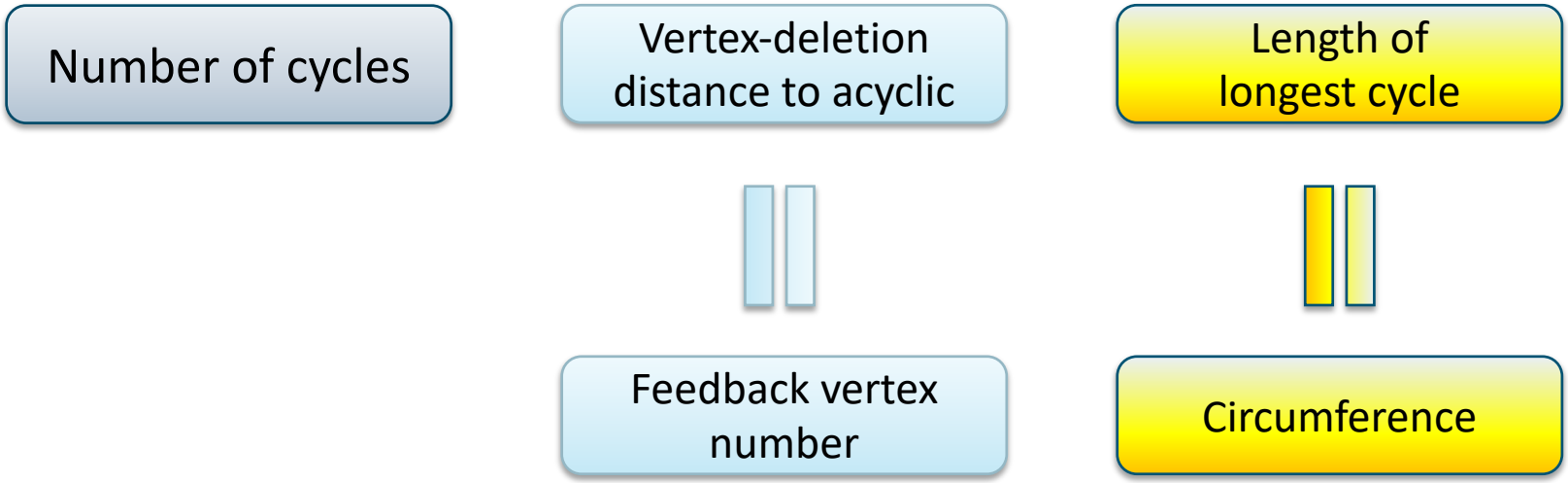








Measuring graph complexity



These are graph parameters:

functions assigning integers to abstract graphs

These 3 parameters give value 0 to all trees

Which parameter captures graph complexity in a useful way?

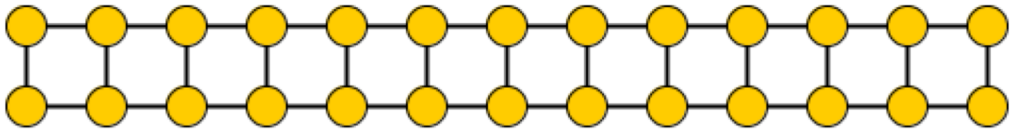
Complexity of a ladder graph

Number of cycles

Vertex-deletion distance to acyclic

Length of longest cycle

None of these, really: ladder graphs score arbitrarily high



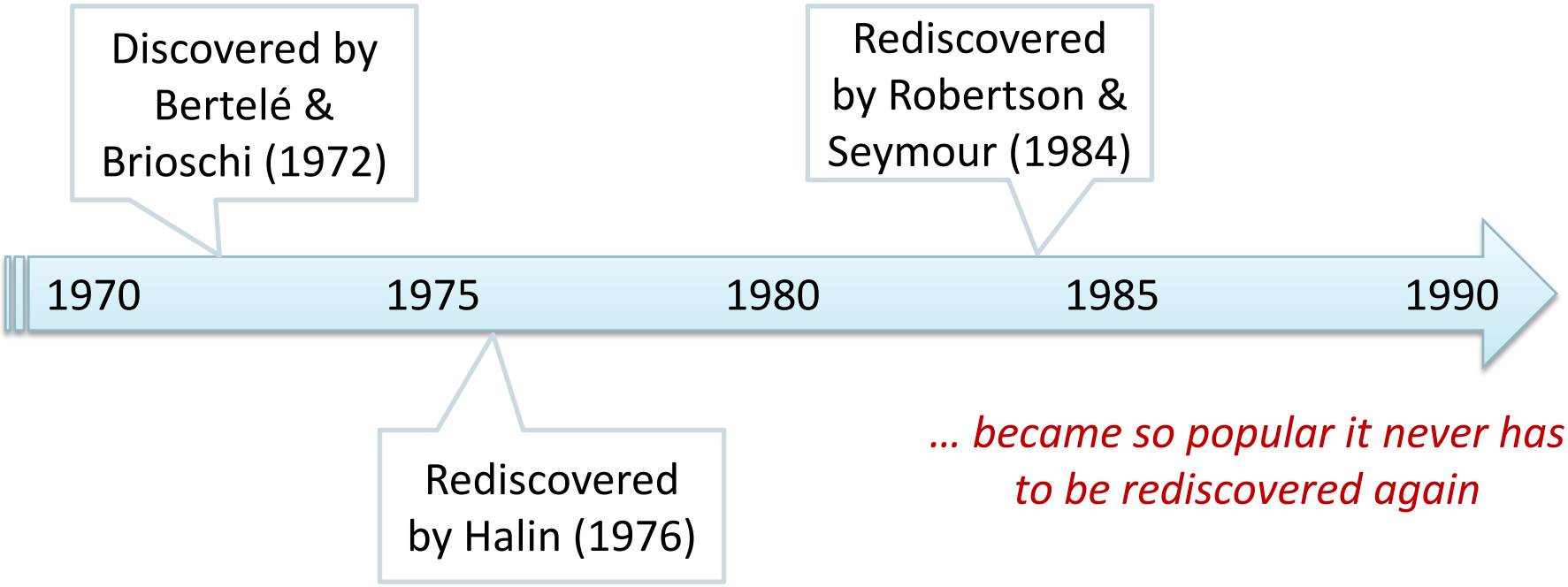
Which parameter captures graph complexity in a useful way?

TREEWIDTH

History

The treewidth of a graph is a useful graph complexity parameter

Intuitively: measures how treelike the graph is



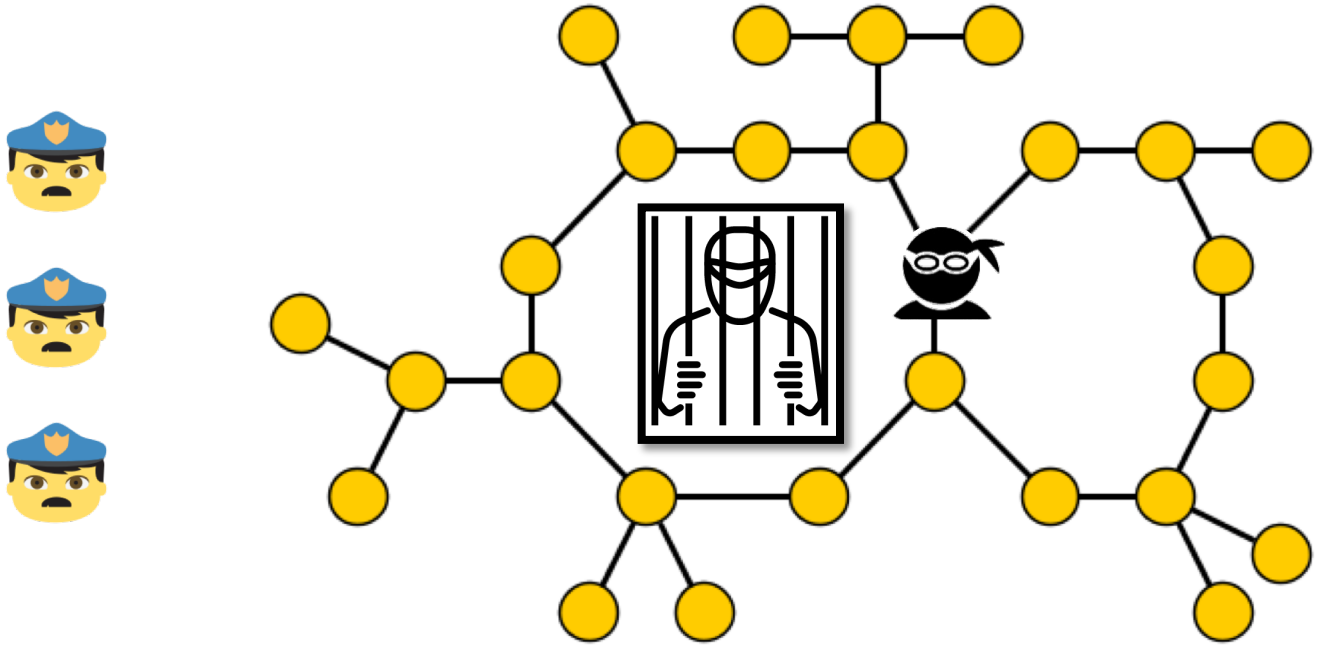
Definition of treewidth

Many cryptomorphic equivalent definitions of treewidth:

1. Minimum width of a tree decomposition
2. Minimum clique number of a chordal supergraph
3. Minimum cost of an elimination order
4. Minimum k for which the graph is a partial k-tree
5. Minimum number of cops needed to catch a visible robber

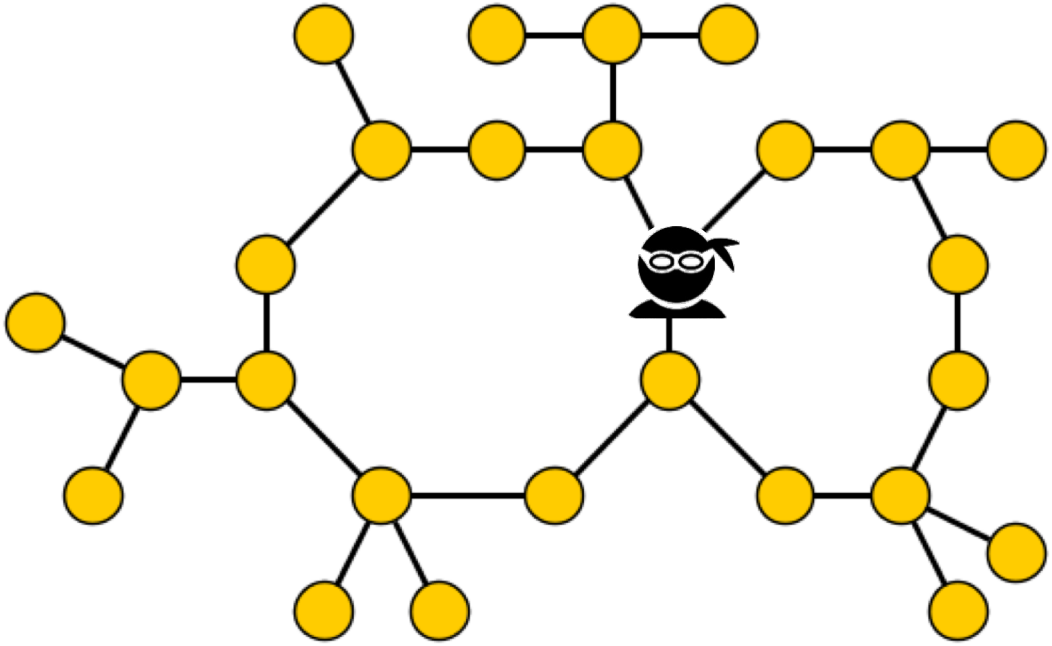
Cops and robbers on graphs

- Fast robber is running along the edges of a graph
- Cops in a helicopter try to catch him by landing and blocking vertices
- Helicopters are slow: robber sees helicopter coming and runs away



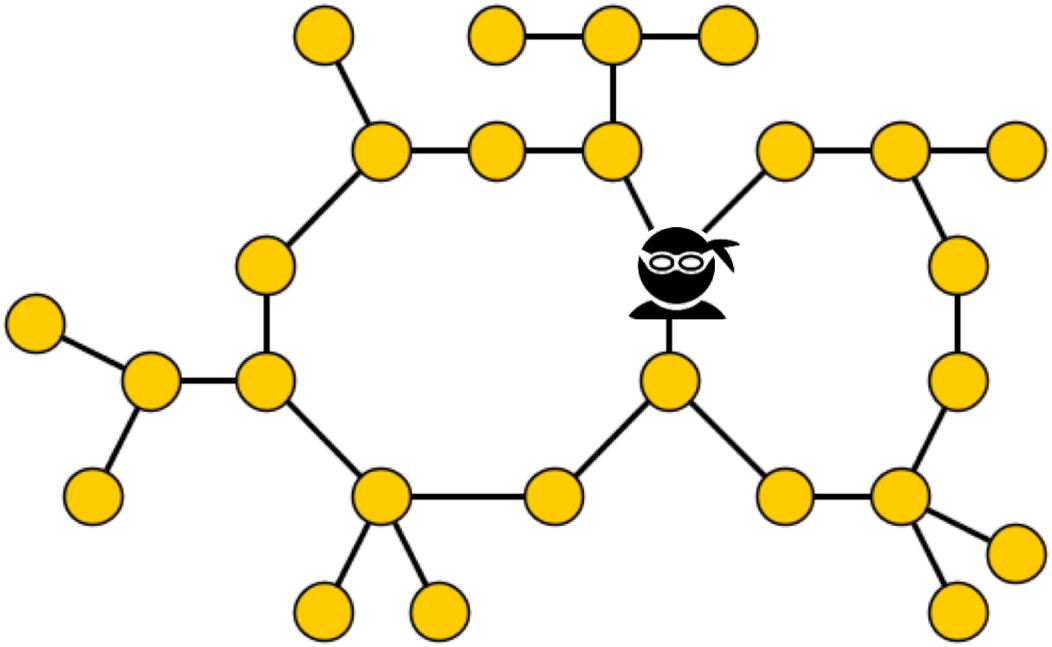
Graph parameter from cops and robbers

What is the smallest number of cops needed to catch any robber on G?



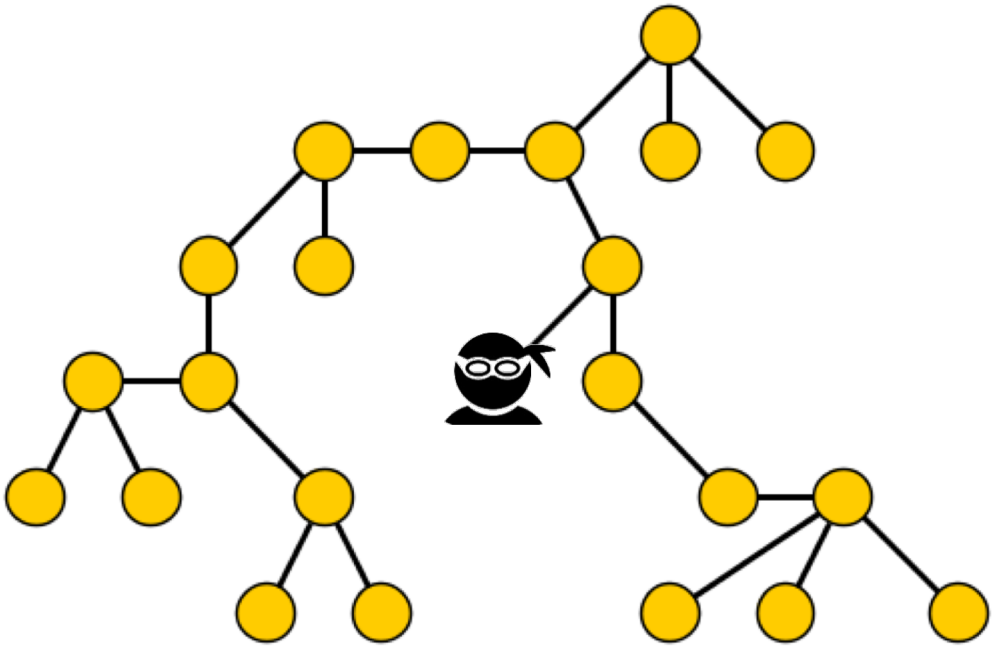
Robbers can hide on cycles

Fact. For any graph with a cycle, you need at least 3 cops.

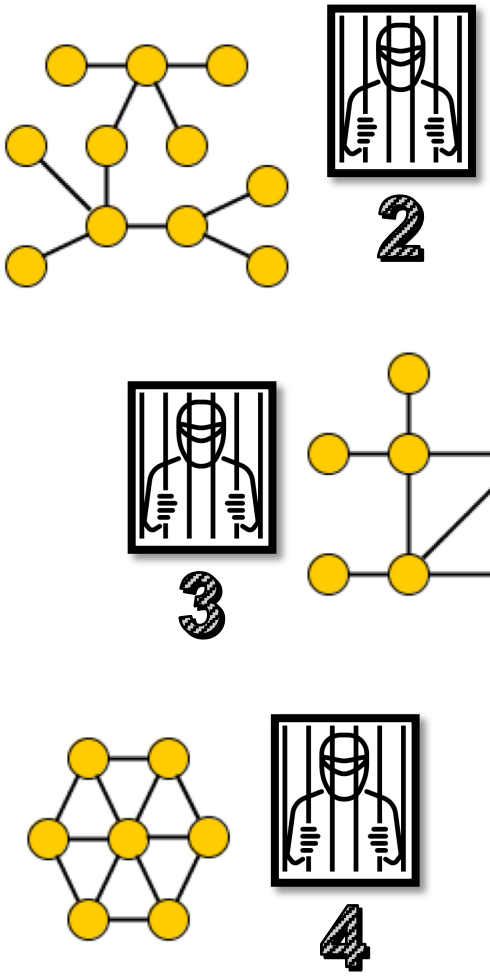


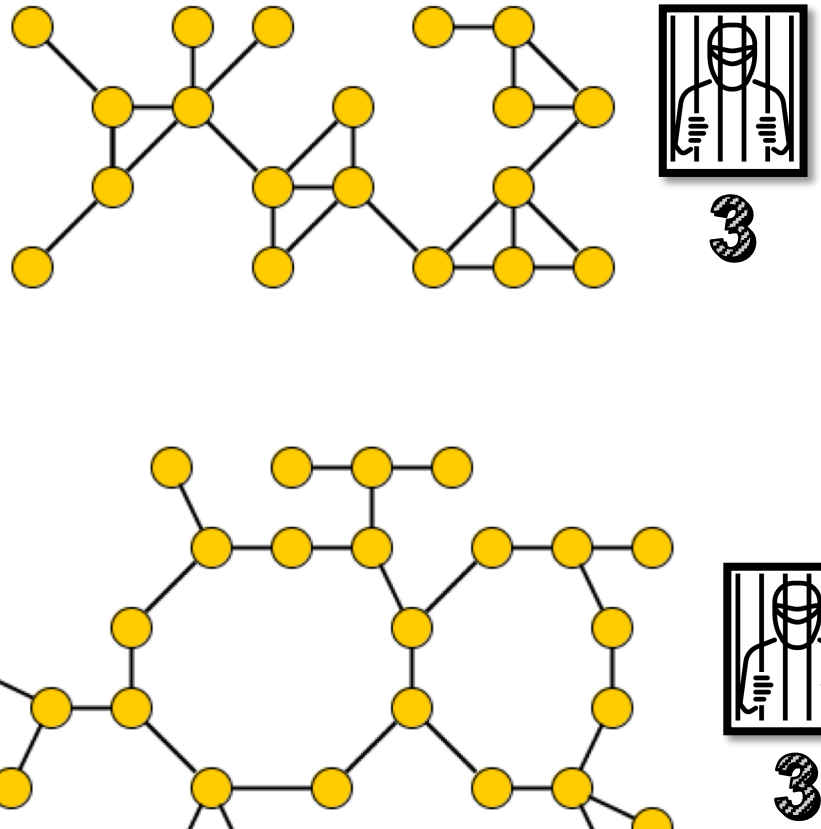
Robbers cannot hide on trees

Fact. In a tree, 2 cops can catch any robber.



Catching robbers on the contest graphs



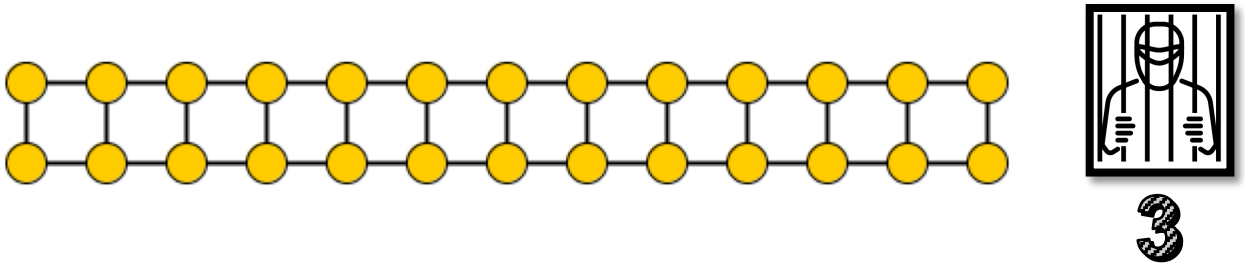


Catching robbers on ladder graphs

Number of cycles

Vertex-deletion distance to acyclic

Length of longest cycle



In a ladder graph, 3 cops can catch any robber

Cop number versus other parameters

Number of cycles

Vertex-deletion distance to acyclic

Length of longest cycle

I. $\text{CopNumber}(G) \leq \text{NumberOfCycles}(G) + 2$

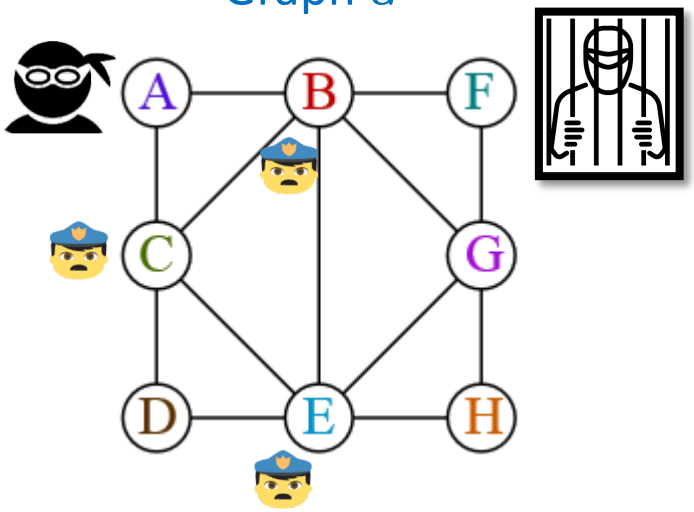
II. $\text{CopNumber}(G) \leq \text{DistanceToAcyclic}(G) + 2$

III. $\text{CopNumber}(G) \leq \text{LongestCycle}(G)$

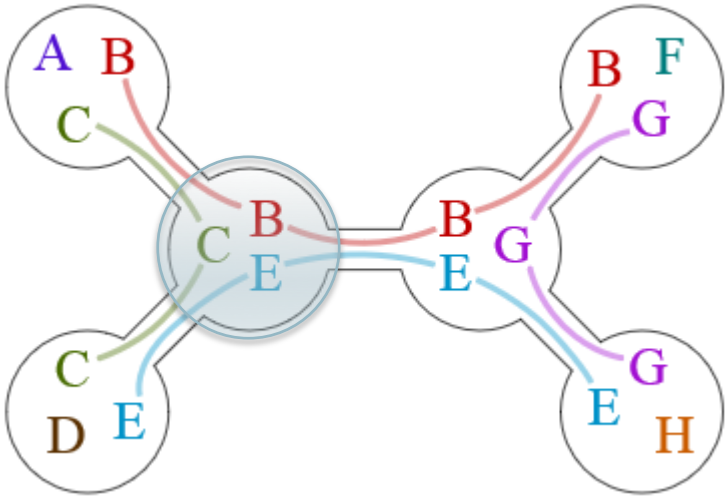
The cop number is small on a larger class of graphs …
yet graphs with small cop number still have useful structure

Representing search strategies

Graph G



Tree T encoding strategy for 3 cops



Each node $v \in V(T)$ gives cop-occupied positions in state v

1. Start with cops in locations specified by start node (arbitrary)
2. Consider $x \in V(G)$ that robber runs to, find x in T
3. Change cop state as indicated by tree-neighbor towards x

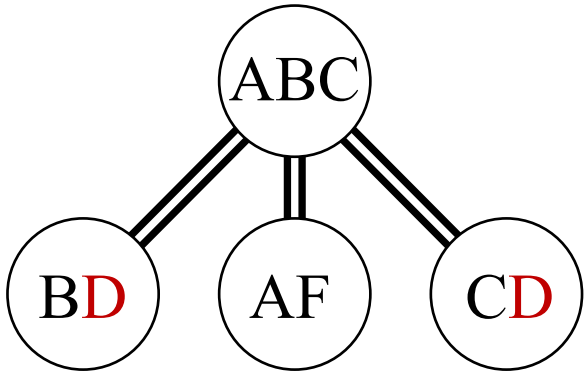
Which trees represent valid strategies?

To be able to catch a robber hiding along edge uv:

A. $\forall uv \in E(G)$, some state has simultaneous cops on u and v

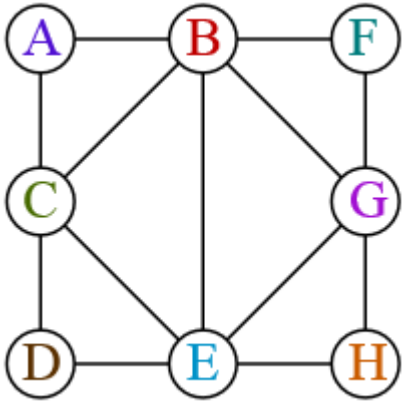
To ensure next cop move is well-defined:

B. $\forall v \in V(G)$, states with v-cop form connected subtree of T

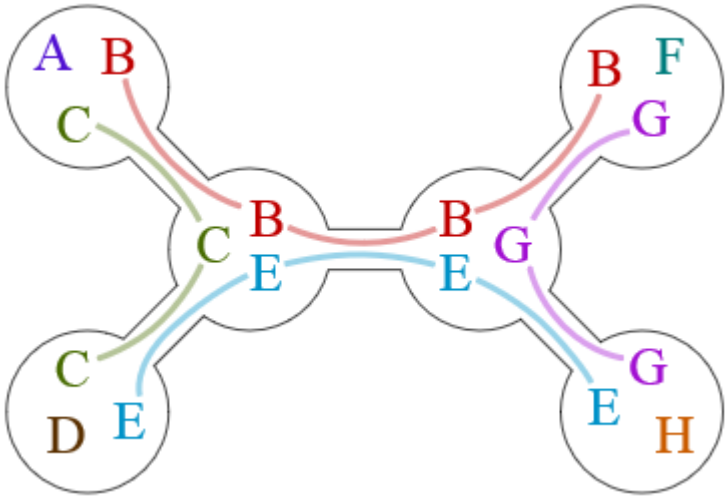


Representing search strategies

Graph G



Tree T encoding strategy for 3 cops



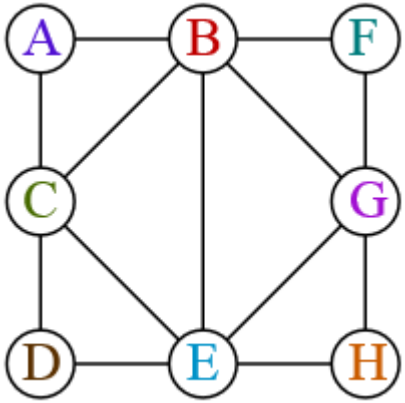
A. $\forall uv \in E(G)$, some state has simultaneous cops on u and v

B. $\forall v \in V(G)$, states with v-cop form connected subtree of T

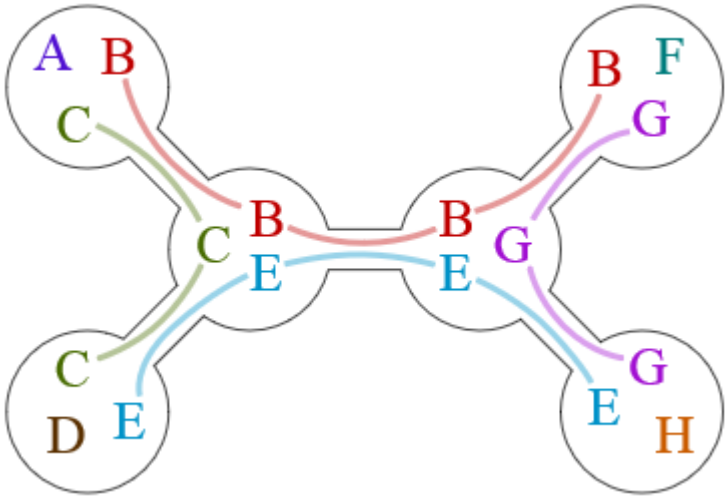
Necessary conditions (A),(B) are sufficient to ensure validity.
There is always an optimal strategy of this form.

Tree decompositions

Graph G



Tree decomposition (T, X)



A tree decomposition of G is a pair (T, X) where T is a tree and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

B. $\forall v \in V(G)$, bags containing v form connected subtree of T

Treewidth

The width of a tree decomposition is the maximum bag size – 1

The treewidth of G is the min. width of its tree decompositions

Theorem. Graph G has treewidth at most $k-1$
$\Leftrightarrow$
k cops are sufficient to catch any robber on G.

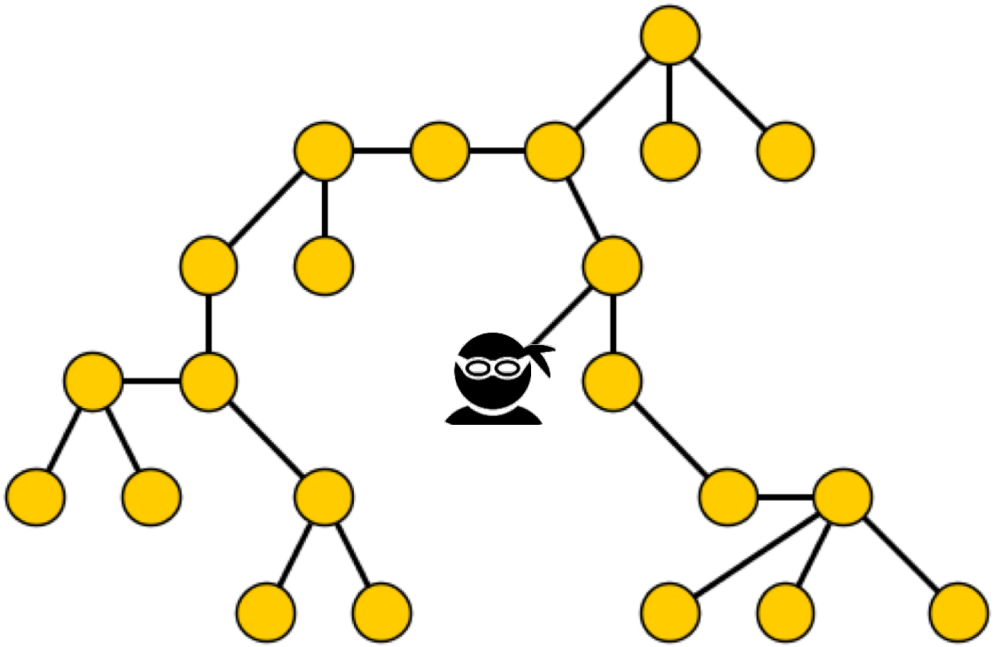
A tree decomposition of G is a pair (T, X) where T is a tree and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

B. $\forall v \in V(G)$, bags containing v form connected subtree of T

Robbers cannot hide on trees

Fact. A tree has treewidth 1.



Connected subgraphs yield connected subtrees

For $v \in V(G)$ let $X^{-1}(v)$ be the nodes of T whose bag contains v

For $S \subseteq V(G)$ let $X^{-1}(S)$ be the nodes of T whose bag intersects S

Lemma 1. If H is a connected subgraph of G, then $X^{-1}(V(H))$ is a connected subtree of T.

A tree decomposition of G is a pair (T, X) where T is a tree and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

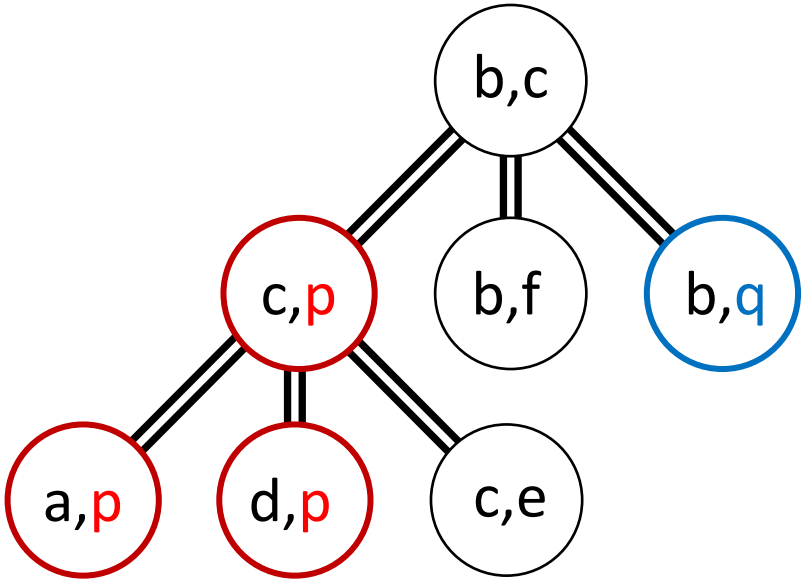
B. $\forall v \in V(G)$, bags containing v form connected subtree of T

PROPERTIES OF TREEWIDTH

Bags of the decomposition form separators

Lemma 1. If H is a connected subgraph of G, then $X^{-1}(V(H))$ is a connected subtree of T.

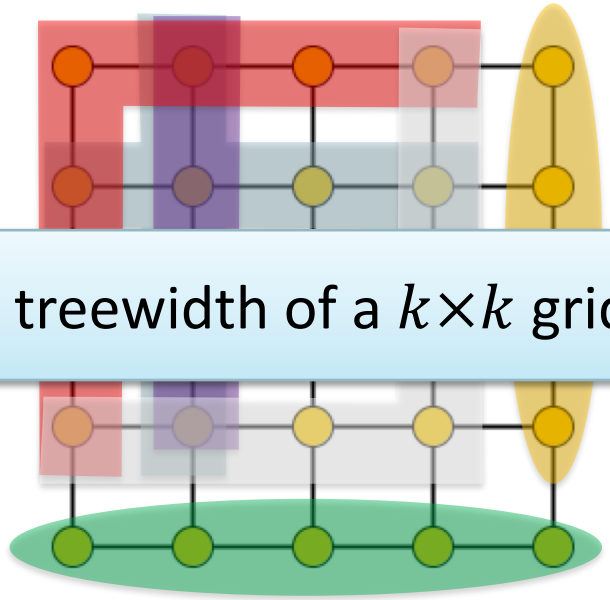
Let (T, X) be a tree decomposition of G. Let $p, q \in V(G)$. Let $w \in V(T)$ such that all paths from $X^{-1}(p)$ to $X^{-1}(q)$ in T go through node w. Then all paths from p to q in G intersect $X(w)$.



Implies robber will be caught

The treewidth of a grid

How many cops are needed to catch a robber on the $k \times k$ grid?



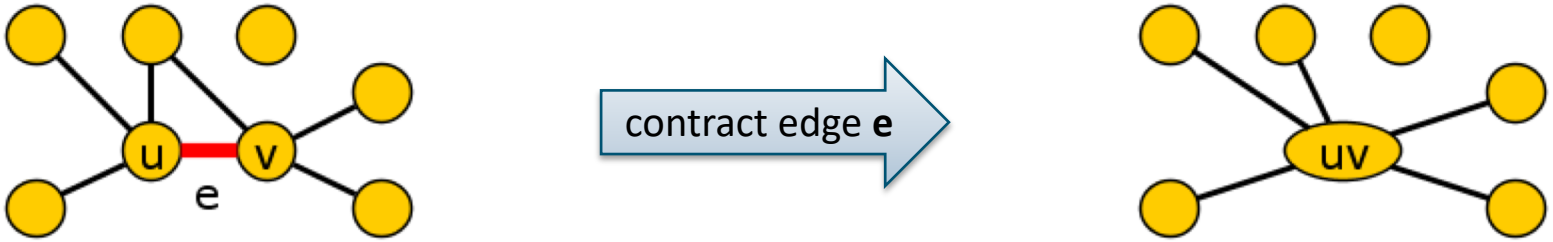
Lemma. The treewidth of a $k \times k$ grid is exactly k.

Robber can hide from k cops on the following subgraphs:

bottom row, remainder of last column,
and all "crosses" in the top-left $(k-1) \times (k-1)$ grid
including off-diagonal ones

Contracting edges does not increase treewidth

Lemma. If graph H can be obtained from G by a sequence of *edge deletions, vertex deletions, and edge contractions,* then H is a minor of G and $treewidth(H) \leq treewidth(G)$.



Corollary. If you can obtain a $k \times k$ grid from G by deleting edges, deleting vertices, and contracting edges, then $treewidth(G) \geq treewidth(\boxplus_{k \times k}) = k$.

The excluded grid theorem

Corollary. If contains a $k \times k$ grid as a minor, then $treewidth(G) \geq k$.

In some sense, large grid minors are the only reason for large TW

Excluded grid theorem. [RS'86, RST'94, KK'12, LS'14, CC'14, Chuzhoy'16]
If $treewidth(G) = k$, then G contains a $k' \times k'$ grid as a minor, where $k' \in \Omega\left(k^{\frac{1}{20}}\right)$.

Balanced separators from tree decompositions

Lemma. If the n-vertex graph G has treewidth k, then there is a separator $S \subseteq V(G)$ of size $k+1$ such that each connected component of $G - S$ has at most $\frac{n}{2}$ vertices.

Proof.

1. Consider tree decomposition (T, X) of G
2. Pick a root node r for T. Then for each node $v \in V(T)$:
   - Let T_v be the subtree of T rooted at v
   - Let $X(T_v) \coloneqq \bigcup_{u \in V(T_v)} X(u)$ be vertices of G in bags of T_v
3. Pick deepest $v \in V(T)$ such that $|X(T_v)| > \frac{n}{2}$, use $S \coloneqq X(v)$

EXERCISES ON TREEWIDTH

Small-treewidth graphs have a low-degree vertex

Lemma. If G has treewidth k,
then G has a vertex of degree at most k.

Proof strategies:

1. If $k+1$ cops can catch any robber on G, it has a deg-k vertex
2. Find a degree-k vertex by inspecting leaves of decomposition

Corollary.
An n-vertex graph of treewidth k has at most nk edges.

Cliques are represented in tree decompositions

Lemma. If $S \subseteq V(G)$ forms a clique in G, then any tree decomposition for G has a bag containing all vertices of S.

Helly property. If T is a tree and $T_1, \ldots, T_k$ are connected subgraphs of T such that $V(T_i) \cap V(T_j) \neq \emptyset$ for all $1 \leq i, j \leq k$, then $V(T_1) \cap V(T_2) \cap \cdots \cap V(T_k) \neq \emptyset$.

Corollary.
If G has a clique of size k, then $treewidth(G) \geq k - 1$.

CLOSING REMARKS

An arboretum of graphs of bounded treewidth

series-parallel graphs (treewidth ≤ 2)

k-outerplanar graphs

k vertex-deletions away from acyclic

Summary

Treewidth is a graph parameter measuring treelike-ness

- Definition in terms of tree decompositions
- Definition in terms of cops catching a fast robber

Concept of treewidth gives a research methodology

- Generalize results on trees to bounded-treewidth graphs

Graphs of treewidth k are more general than:

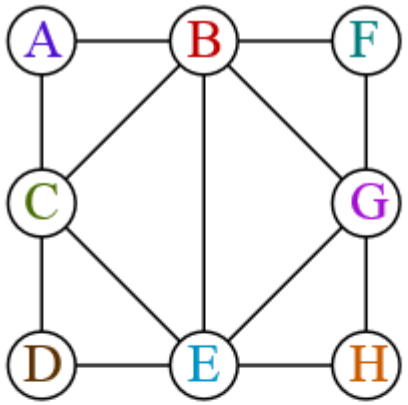
- Graphs with k cycles
- Graphs that are k vertex-deletions away from acyclic

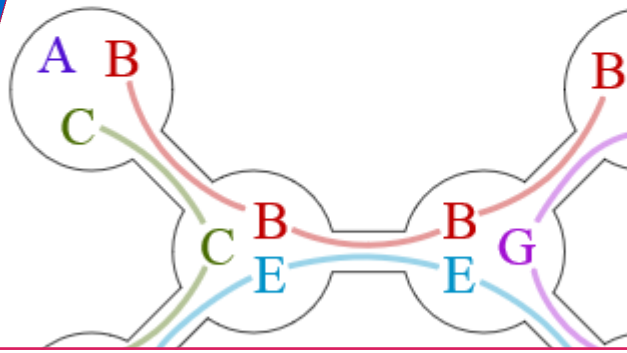
More parameters Wednesday, applications on Thursday

NET WORKS

Structural graph parameters

PART 2: A HIERARCHY OF PARAMETERS





Bart M. P. Jansen (edited by mjd – 2021)

TU/e

Networks Training Week 2017
January 31st – February 2nd 2017, Doorn, Netherlands

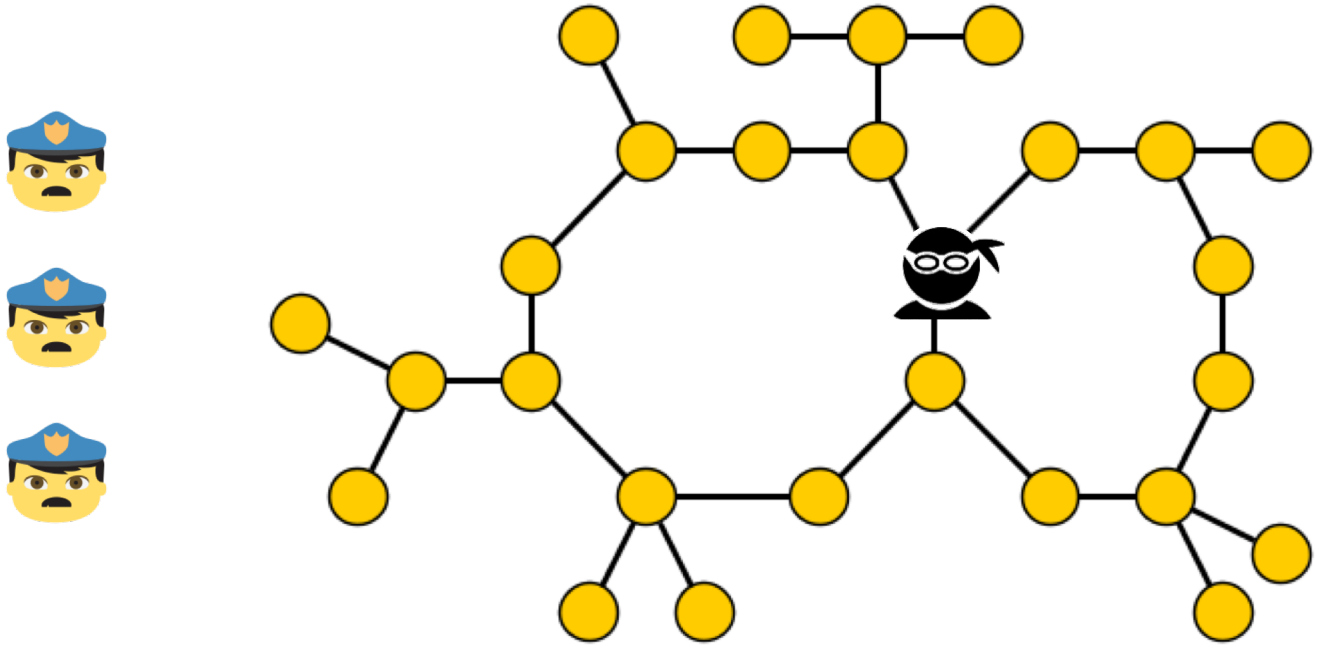
Recap

Yesterday we explored treewidth and 3 other graph parameters:

1. Number of cycles
2. Vertex-deletion distance to an acyclic graph
3. Length of the longest cycle
4. Treewidth = number of cops to catch a fast robber – 1

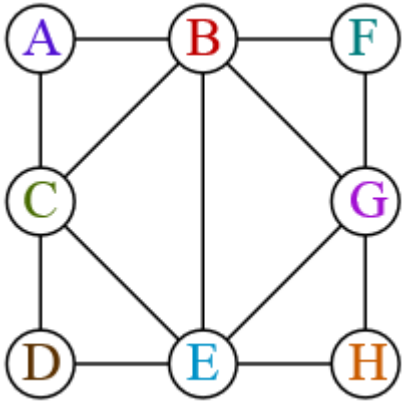
Cops and robbers on graphs

- Fast robber is running along the edges of a graph
- Cops in a helicopter try to catch him by landing and blocking vertices
- Helicopters are slow: robber sees helicopter coming and runs away

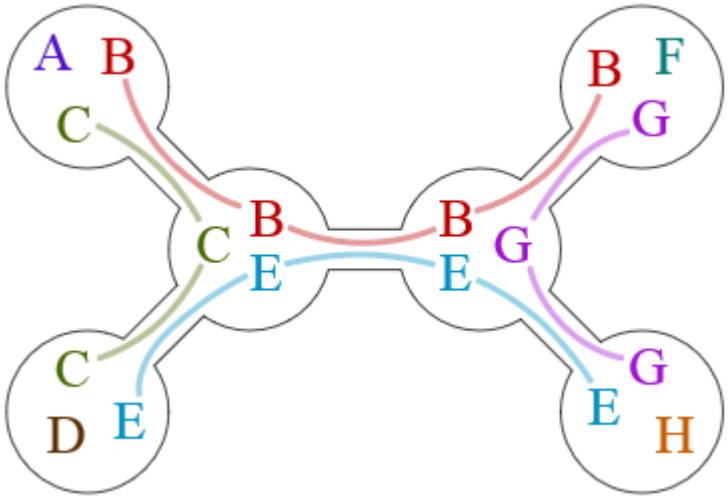


Tree decompositions

Graph G



Tree decomposition (T, X)



A tree decomposition of G is a pair (T, X) where T is a tree and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

B. $\forall v \in V(G)$, bags containing v form connected subtree of T

Treewidth

The width of a tree decomposition is the maximum bag size – 1

The treewidth of G is the min. width of its tree decompositions

Theorem. Graph G has treewidth at most $k-1$
$\Leftrightarrow$
k cops are sufficient to catch any robber on G.

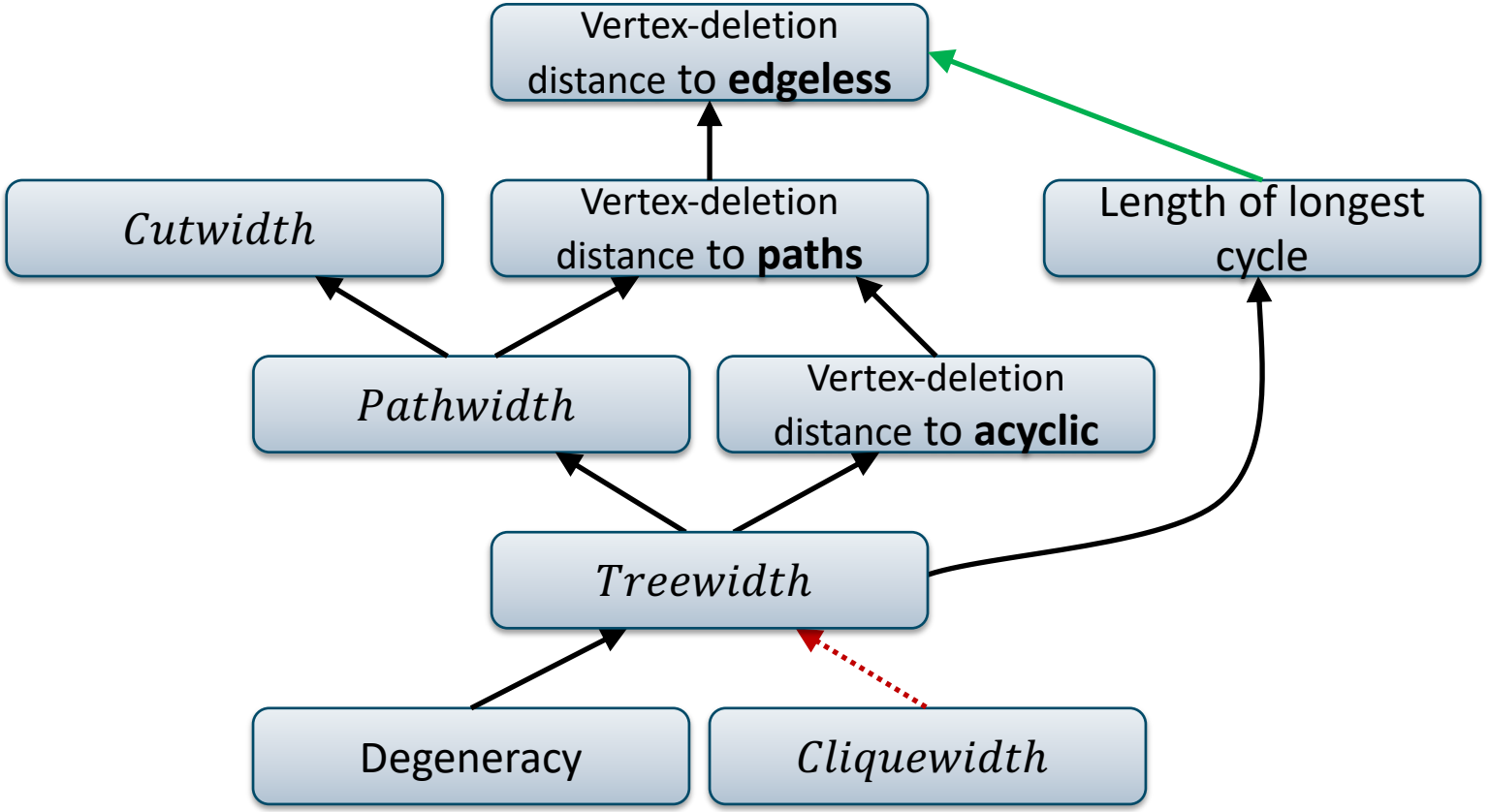
A tree decomposition of G is a pair (T, X) where T is a tree and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

B. $\forall v \in V(G)$, bags containing v form connected subtree of T

Plan for today

More graph parameters and relations between them



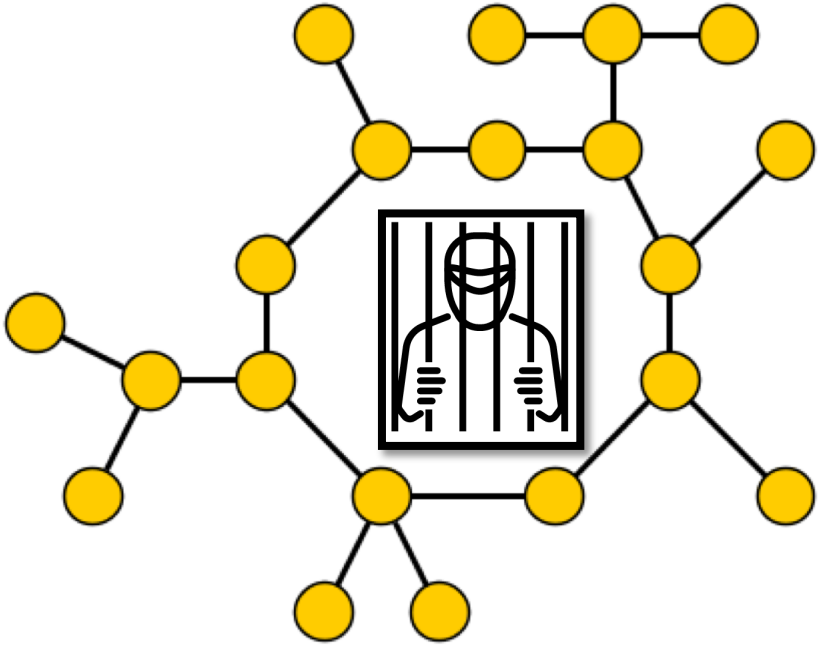
Methodology for generating graph problems & solving them

A NEW PARAMETER

Catching a special robber

- In the game defining treewidth, cops are slow but can see the fast robber
- For the next parameter, we change the rules of the game:

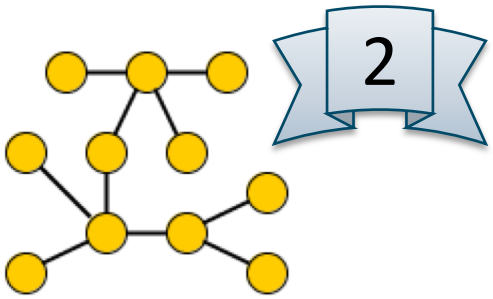
What if the robber is a fast invisible man?

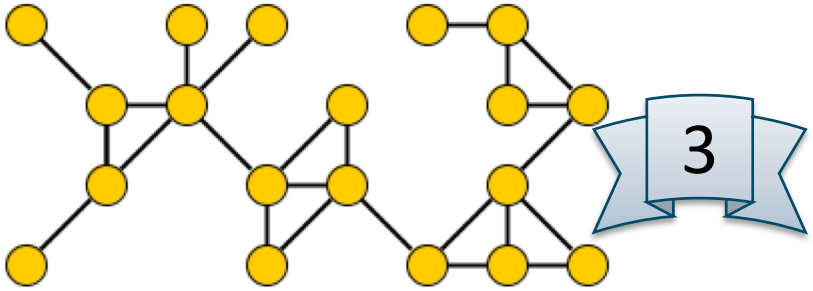


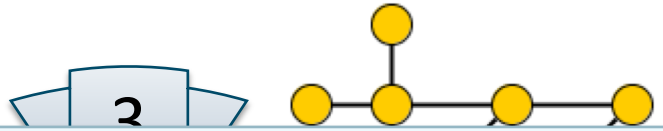
How many cops are needed to catch this invisible robber?

Catching invisible robbers on the contest graphs

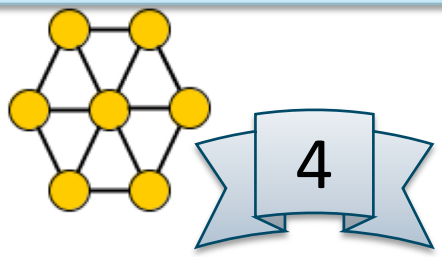
Number of cops to catch invisible robber

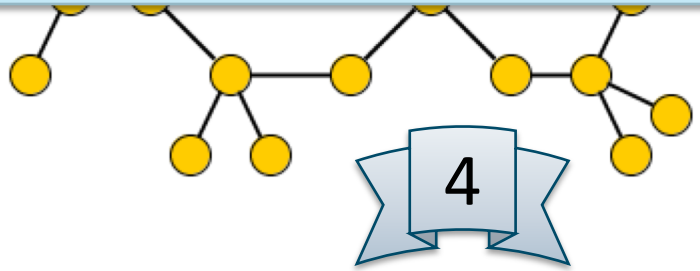






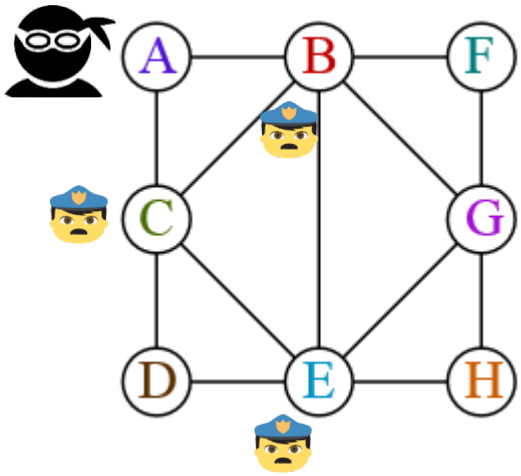
Observation. To catch an invisible robber,
you need a fixed strategy for searching the graph.



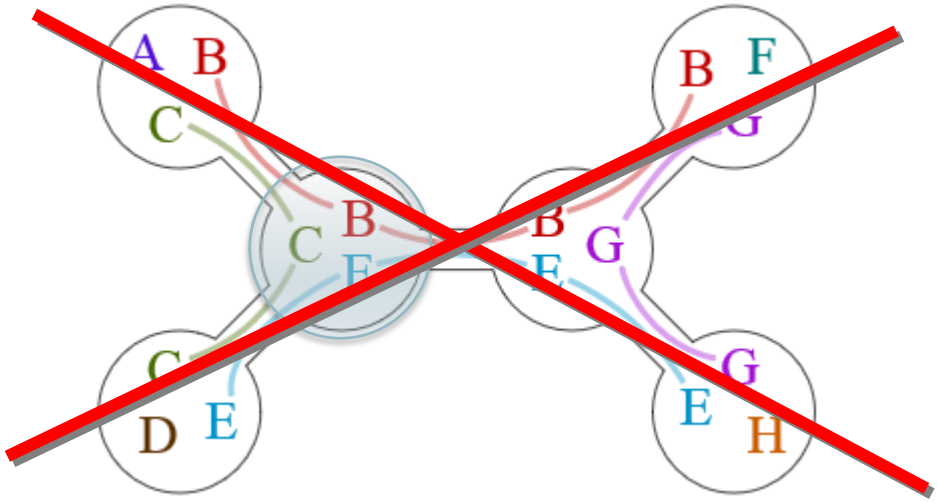


Representing search strategies

Graph G



Tree T encoding strategy for 3 cops



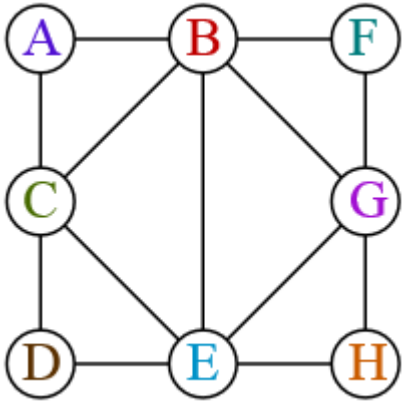
Each node $v \in V(T)$ gives cop-occupied positions in state v

Observation. Search strategy to catch an invisible robber is linear.

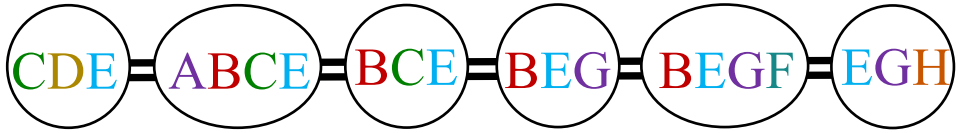
~~3. Change cop state as indicated by tree-neighbor towards x~~

Path decompositions

Graph G



Path decomposition (T, X)



A path decomposition of G is a pair (T, X) where T is a path and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

B. $\forall v \in V(G)$, bags containing v form connected subtree of T

Pathwidth

The width of a path decomposition is the maximum bag size – 1

The pathwidth of G is the min. width of its path decompositions

Theorem. Graph G has pathwidth at most $k-1$
$\Leftrightarrow$
k cops are sufficient to catch any invisible robber on G.

A path decomposition of G is a pair (T, X) where T is a path and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

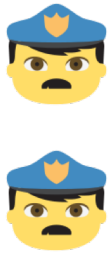
A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

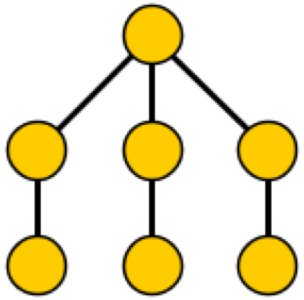
B. $\forall v \in V(G)$, bags containing v form connected subtree of T

Catching robbers on trees

On a tree, a visible robber can be always caught by 2 cops

What about invisible robbers?





Invisible robber knowing the 2-cop strategy remains uncaught!

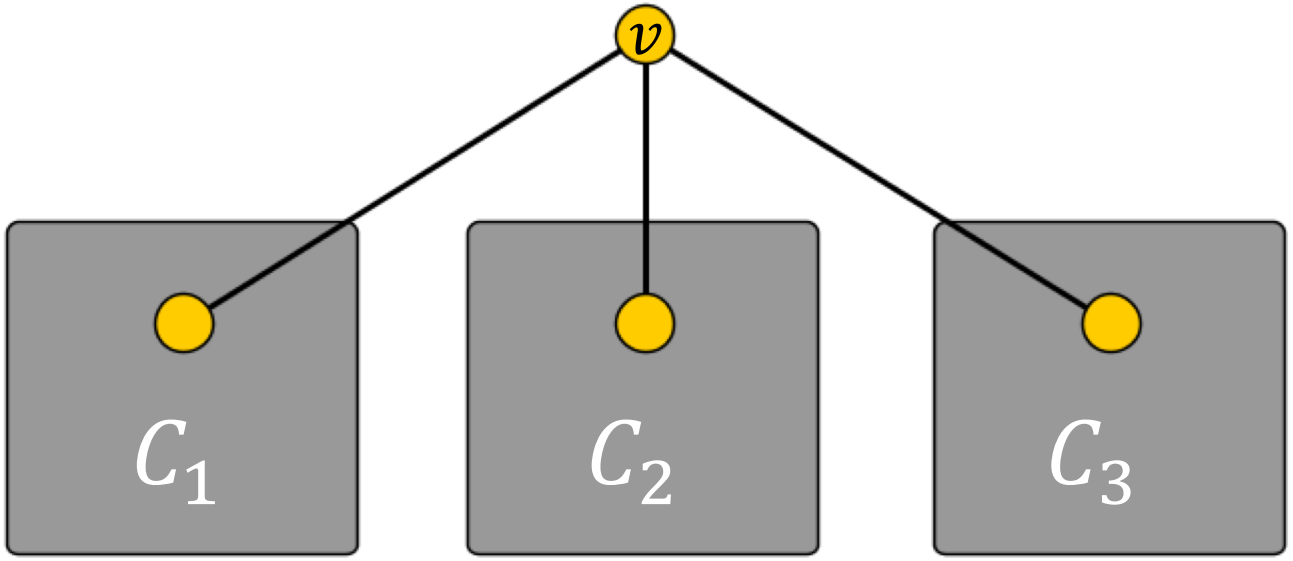
Simple lower bound for bag size of a path decomposition

Hiding places for invisible robbers

If $pathwidth(C_i) = k$ for $i \in \{1,2,3\}$

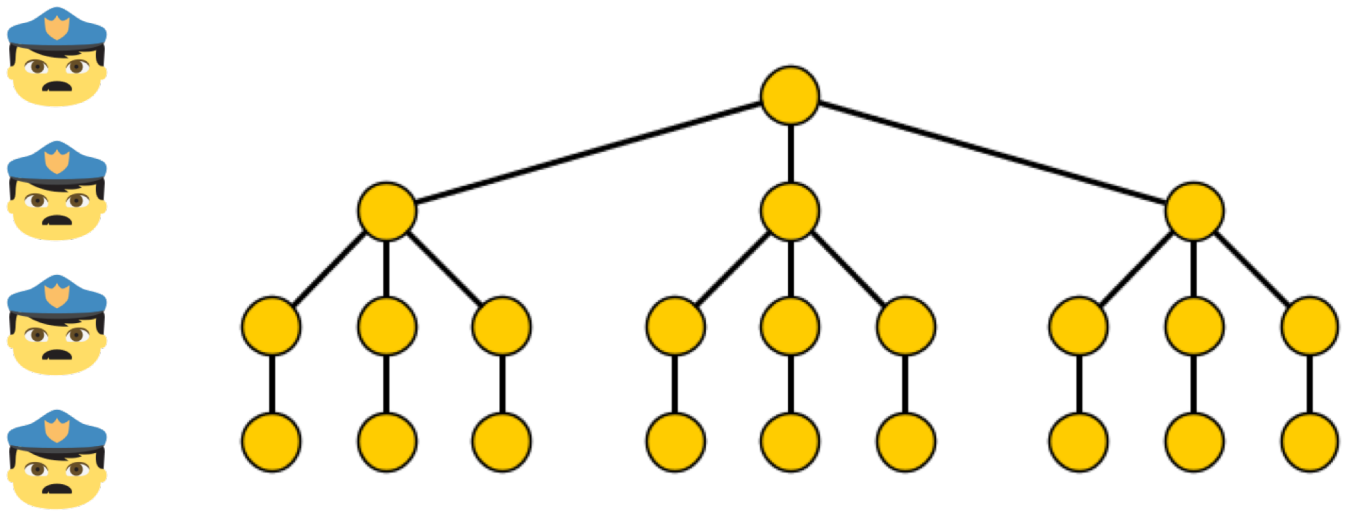
… then $pathwidth(C_1 \cup C_2 \cup C_3 \cup \{v\}) = k + 1$

when v is connected to 1 vertex in each C_i



Analogous lower bound for bag size of a path decomposition

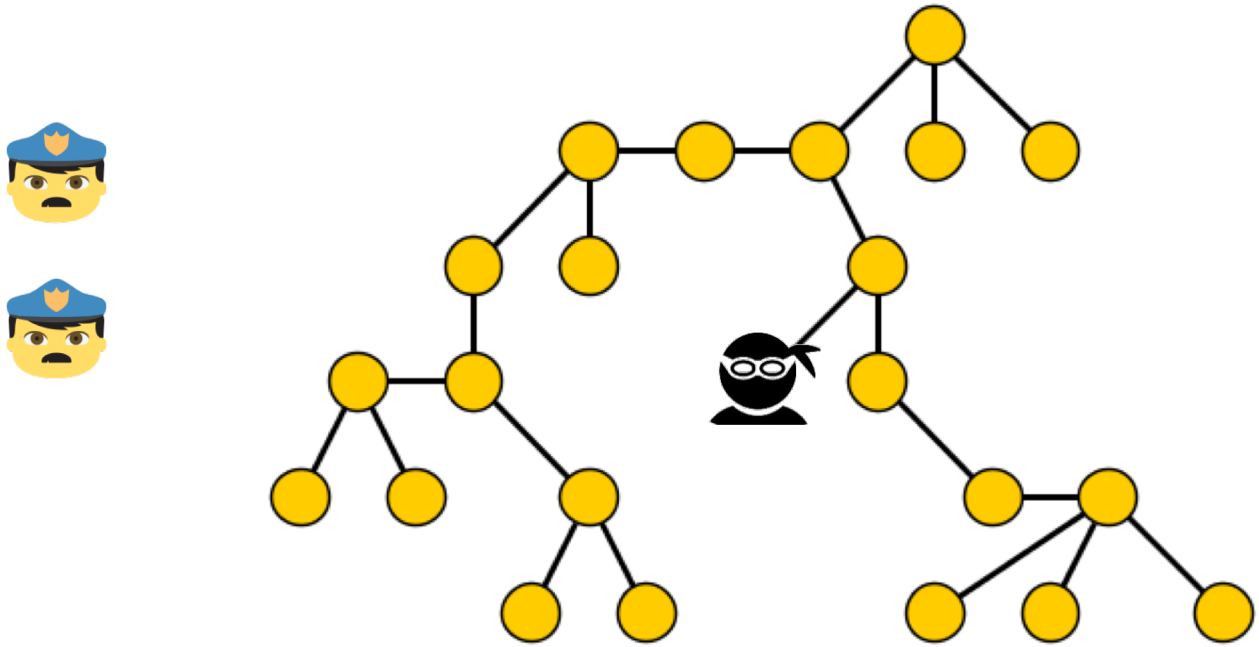
Acyclic hiding places for invisible robbers



Fact. A complete n-vertex ternary tree has pathwidth $\Omega(\log n)$.

Hunting invisible robbers on trees

Fact. In an n-vertex tree, $1 + \log n$ cops catch any invisible robber.

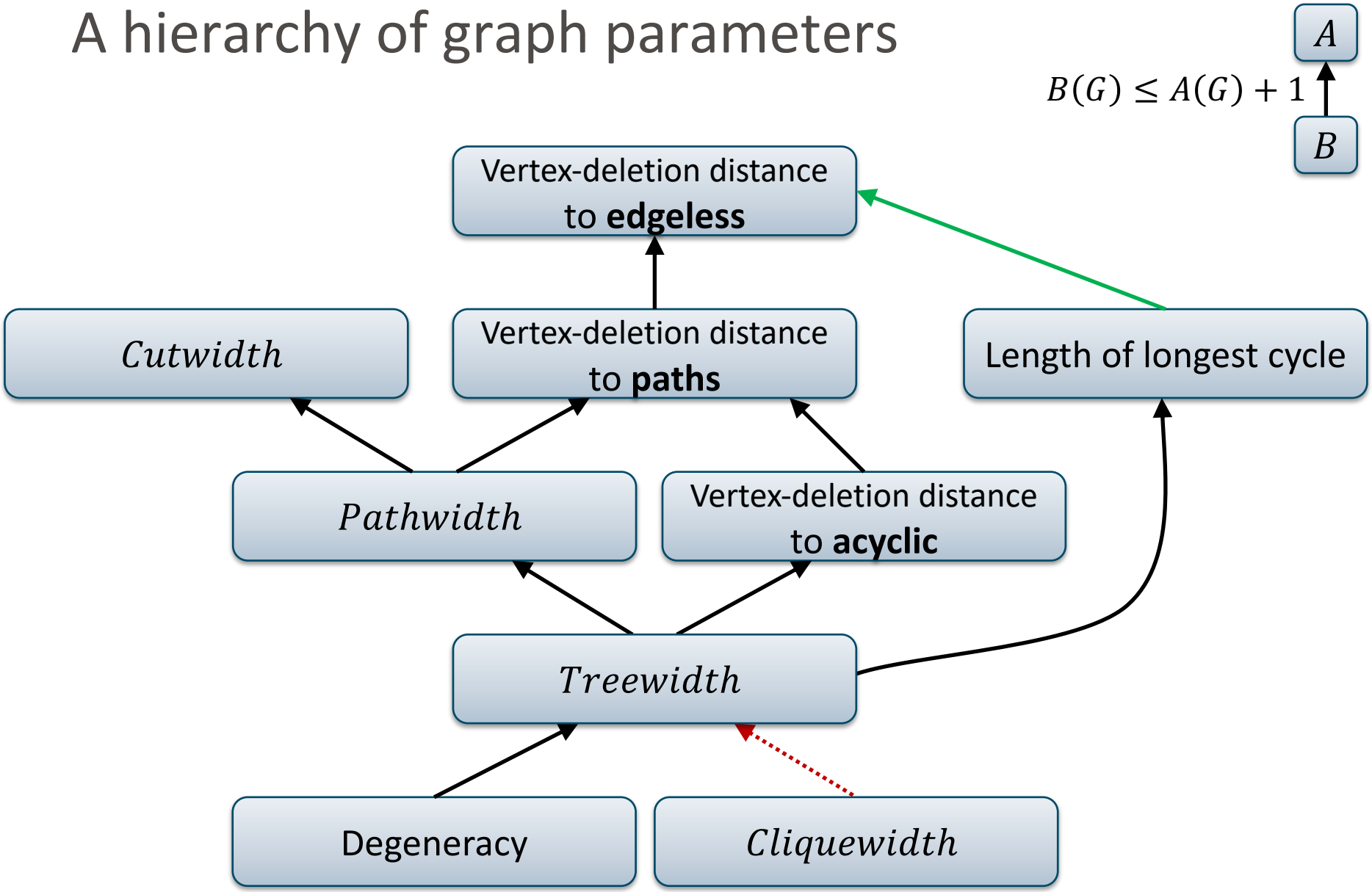


Relation between pathwidth and treewidth

Theorem. For any graph G with n vertices:
$pathwidth(G) \in O(treewidth(G) \cdot \log n)$.

Lemma. If the n-vertex graph G has treewidth k,
then there is a separator $S \subseteq V(G)$ of size $k+1$ such that each connected component of $G - S$ has at most $\frac{n}{2}$ vertices.

A hierarchy of graph parameters



Treewidth is bounded by circumference

Lemma. If the longest (simple) cycle in G has length k, then $treewidth(G) \leq k - 1$.

Proof idea:

- Start from a depth-first search tree in G
- Build a strategy for k cops to catch the robber

The graph minors project

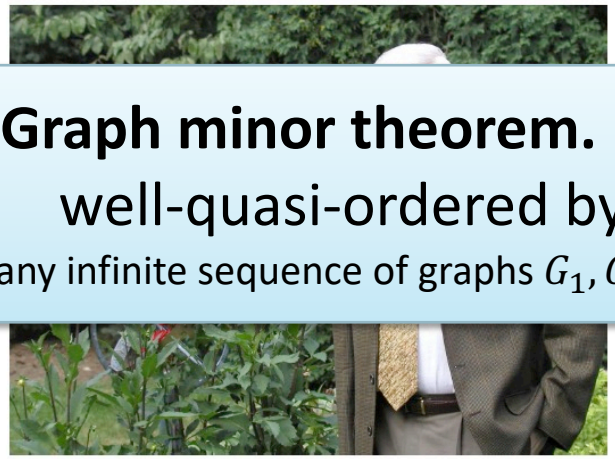
INTERMEZZO

The Graph Minors project

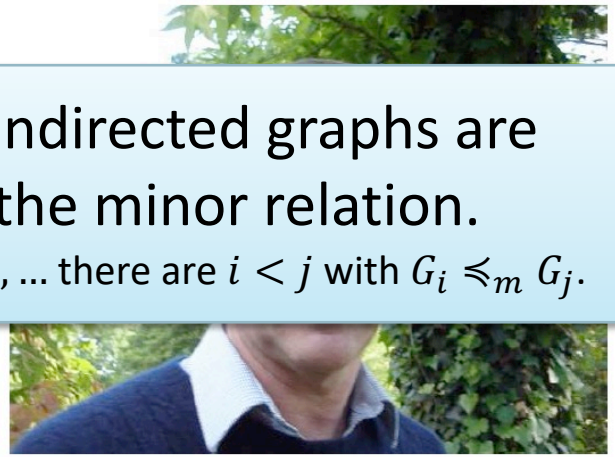
- Series of 20 papers spanning 500+ pages
- Developed from 1983 – 2000
- Resulted in concepts of treewidth, pathwidth, brambles, etc.

Graph minor theorem. Undirected graphs are well-quasi-ordered by the minor relation.

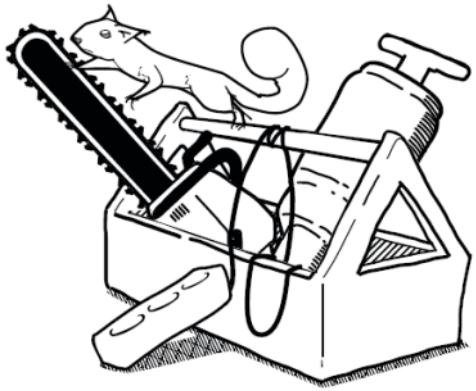
In any infinite sequence of graphs $G_1, G_2, \ldots$ there are $i < j$ with $G_i \preccurlyeq_m G_j$.



Neil Robertson



Paul Seymour



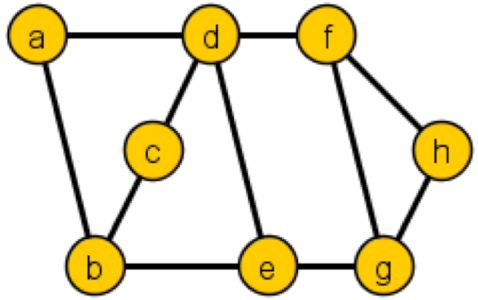
PARAMETERS BASED ON LINEAR STRUCTURE

Graphs with a linear structure

Pathwidth is a measure of the linear structure of a graph

Several other graph parameters are based on linear structure

Cutwidth, bandwidth, topological bandwidth …

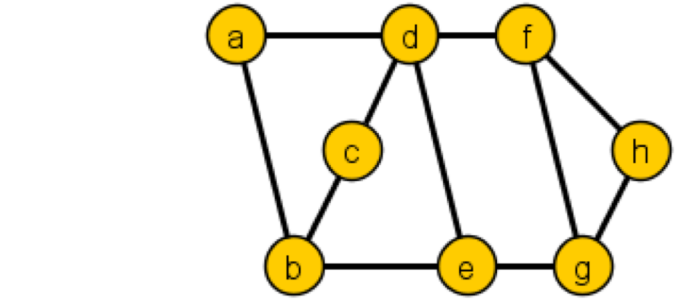


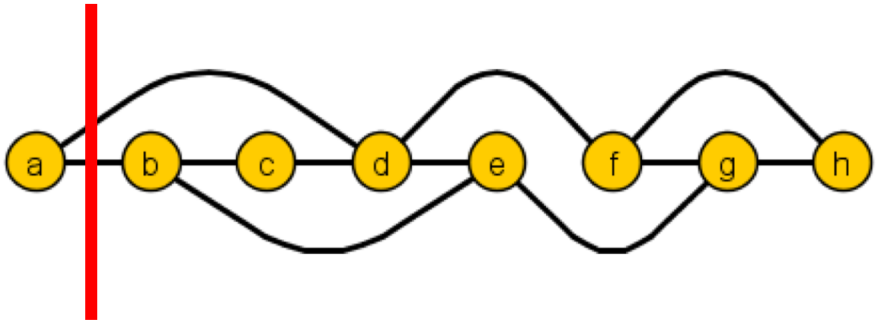
Graphs with a linear structure: cutwidth

The cutwidth of a linear ordering of $V(G)$ is:

max. # edges intersected by a vertical line between vertices

The cutwidth of a graph is min. cutwidth of any ordering of $V(G)$

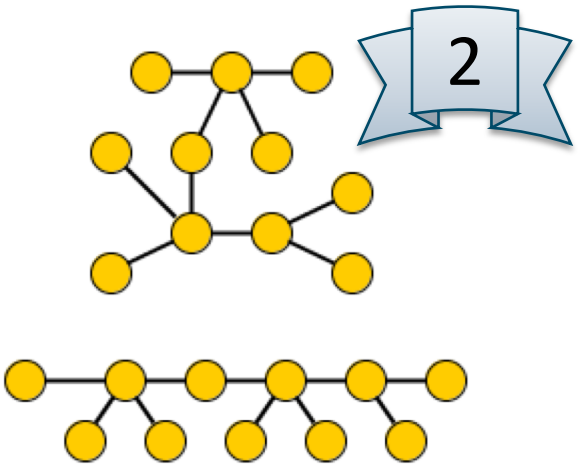


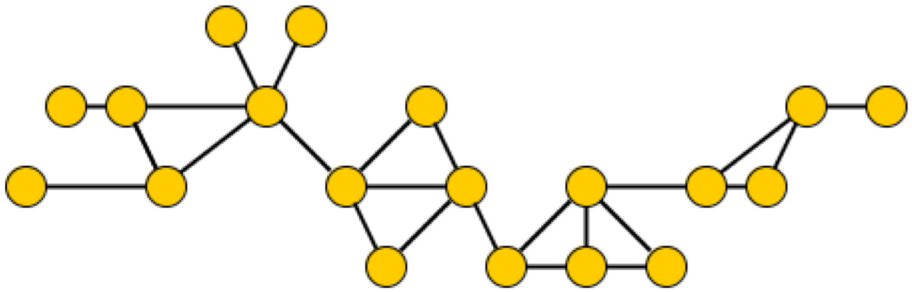


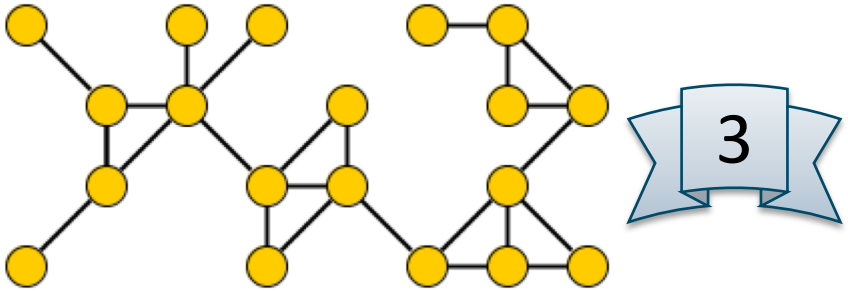
cutwidth 3

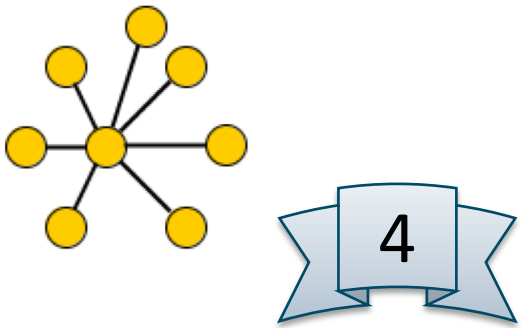
Cutwidth of some simple graphs

Cutwidth









Properties of cutwidth

Lemma. For any graph G: $pathwidth(G) \leq cutwidth(G)$.

Fact. For the star $K_{1,n}$ with n leaves: $cutwidth(K_{1,n}) = \left\lceil \frac{n-1}{2} \right\rceil$.

Fact. For the complete graph K_n: $cutwidth(K_n) = \left\lfloor \frac{n}{2} \right\rfloor \cdot \left\lceil \frac{n}{2} \right\rceil$.

And now, by popular request …

MENTIMETER

BEYOND TREEWIDTH

What if treewidth is too large?

Among all n-vertex graphs, the clique K_n is least treelike

K_n is the unique n-vertex graph of treewidth $n-1$

What about applications where a clique is a `simple' graph?

- Graph parameter that gives small values on trees and cliques?
- Should not be monotone when taking subgraphs

Cliquewidth

The cliquewidth of G is the minimum number of labels to construct it using the following operations in a tree-like fashion:

1. Create new vertex with label i
2. Disjoint union of two graphs G and H
3. Joining each i-labeled vertex to each j-labeled vertex ($i \neq j$)
4. Renaming label i to label j

Constructing a graph of cliquewidth 3

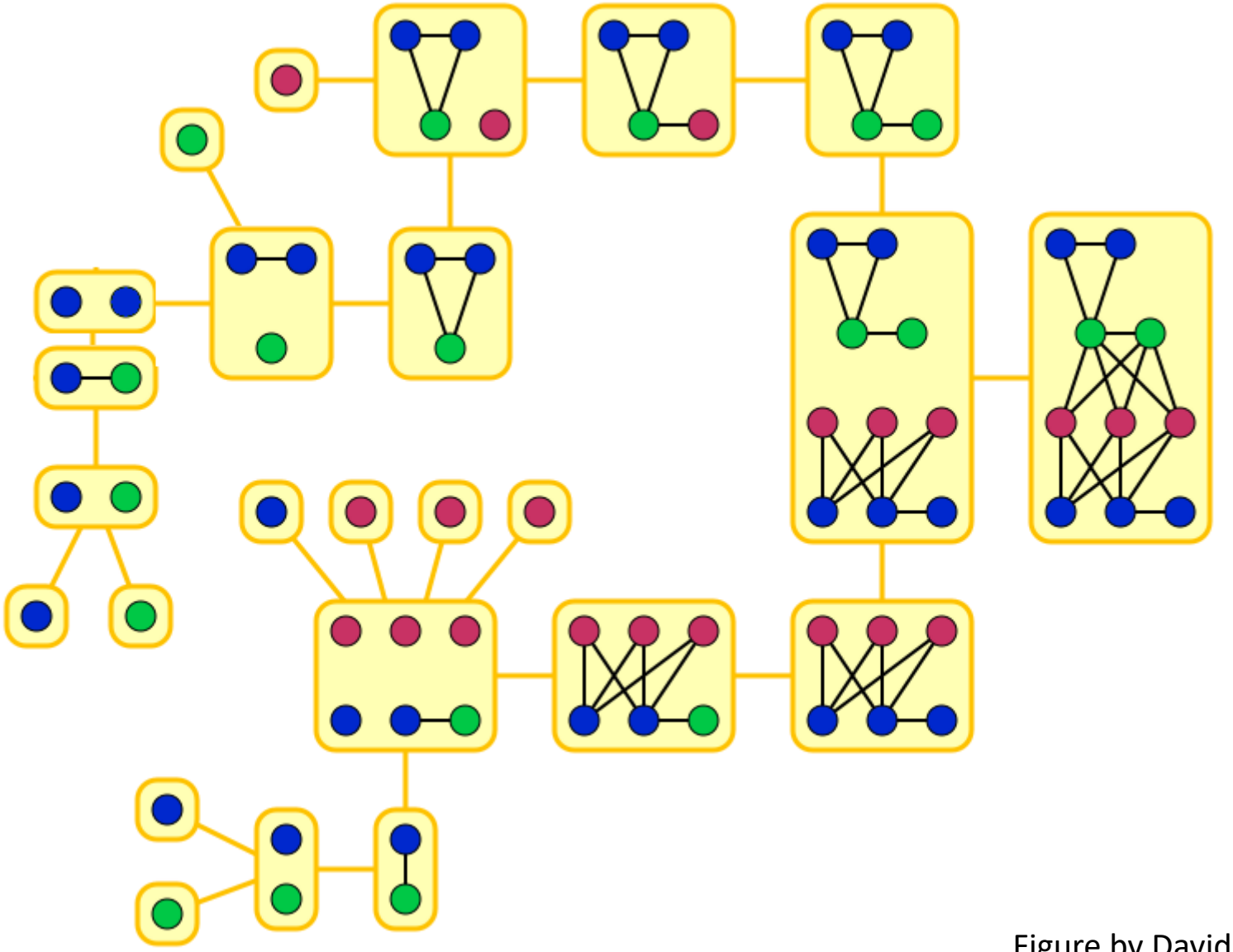


Figure by David Eppstein

Cliquewidth

The cliquewidth of G is the minimum number of labels to construct it using the following operations in a tree-like fashion:

1. Create new vertex with label i
2. Disjoint union of two graphs G and H
3. Joining each i-labeled vertex to each j-labeled vertex ($i \neq j$)
4. Renaming label i to label j

$$cliquewidth(K_n) = 2 \qquad treewidth(K_n) = n - 1$$

EXERCISES ON TREEWIDTH

Small-treewidth graphs have a low-degree vertex

Lemma. If G has treewidth k,
then G has a vertex of degree at most k.

Proof strategies:

1. If $k+1$ cops can catch any robber on G, it has a deg-k vertex
2. Find a degree-k vertex by inspecting leaves of decomposition

Corollary.
An n-vertex graph of treewidth k has at most nk edges.

Cliques are represented in tree decompositions

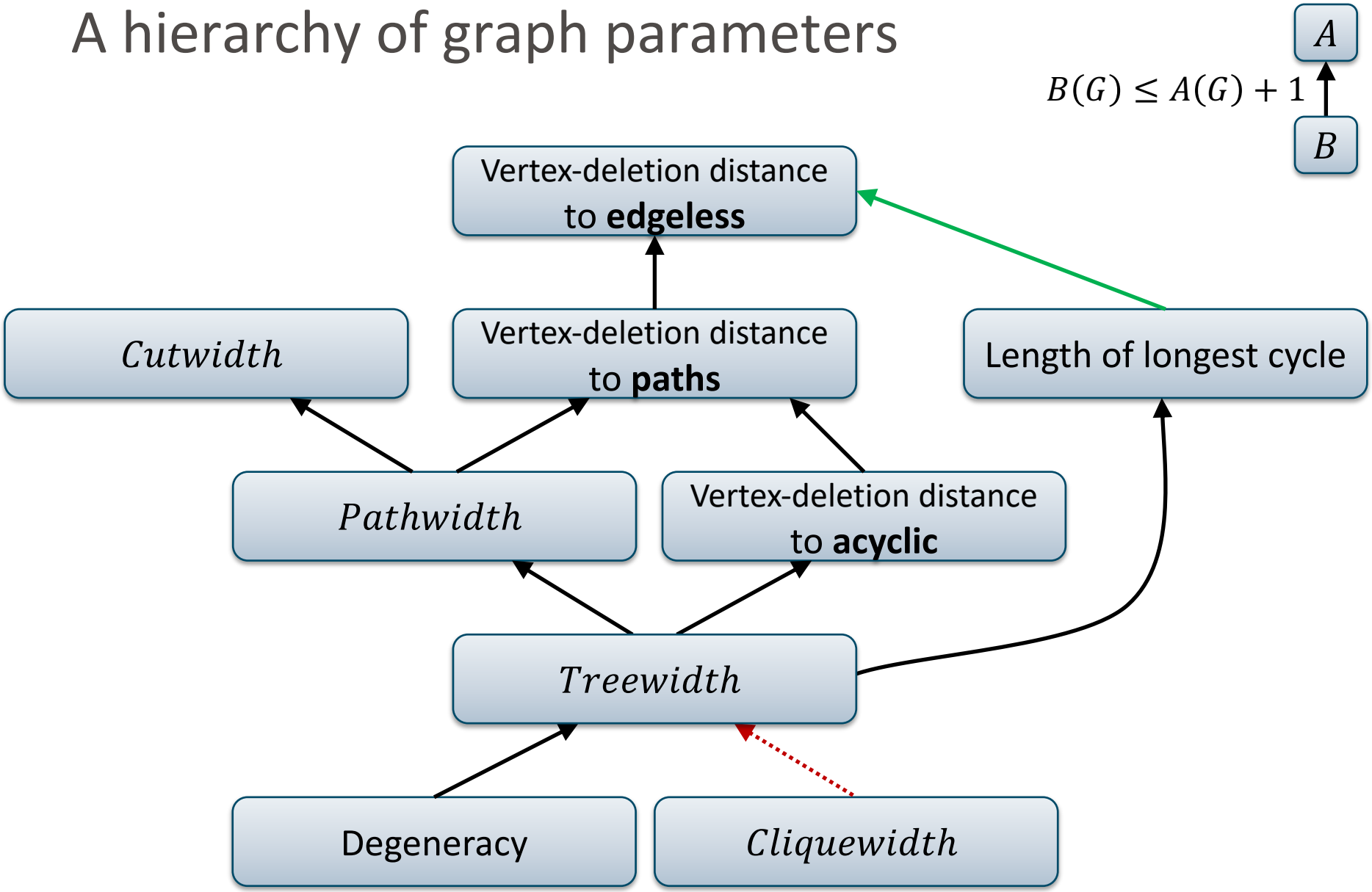
Lemma. If $S \subseteq V(G)$ forms a clique in G, then any tree decomposition for G has a bag containing all vertices of S.

Helly property. If T is a tree and $T_1, \ldots, T_k$ are connected subgraphs of T such that $V(T_i) \cap V(T_j) \neq \emptyset$ for all $1 \leq i, j \leq k$, then $V(T_1) \cap V(T_2) \cap \cdots \cap V(T_k) \neq \emptyset$.

Corollary.
If G has a clique of size k, then $treewidth(G) \geq k - 1$.

CLOSING REMARKS

A hierarchy of graph parameters



Summary

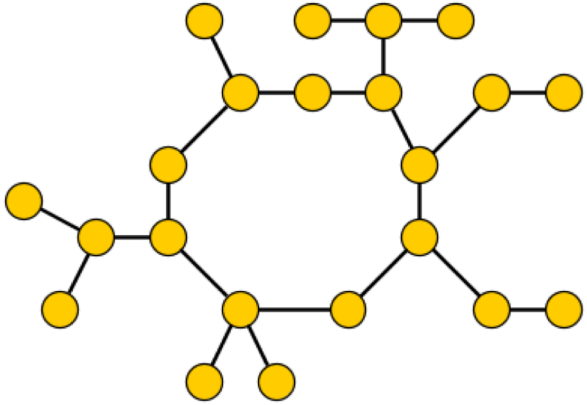
Various sets of rules for cops-and-robber games yield:

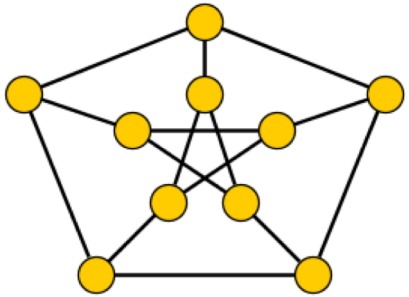
- Different graph parameters
- Associated graph decompositions representing strategies

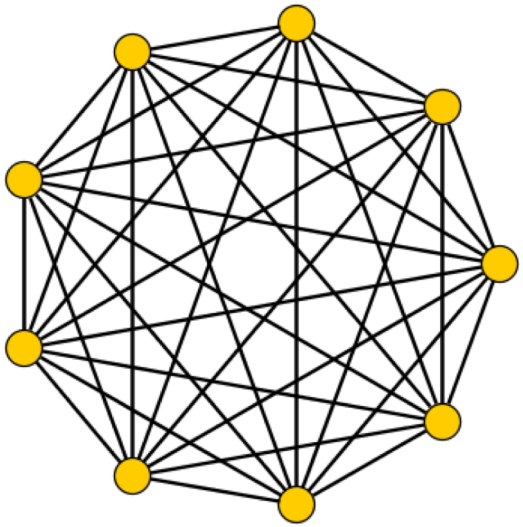
Hierarchy of parameters suggests a research methodology

- Tackle problem for small values of restrictive parameter
- Then generalize to more robust parameters

Applications on Thursday



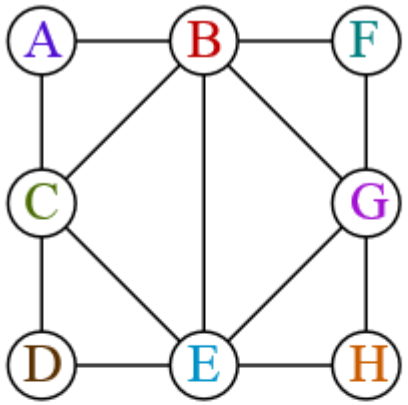


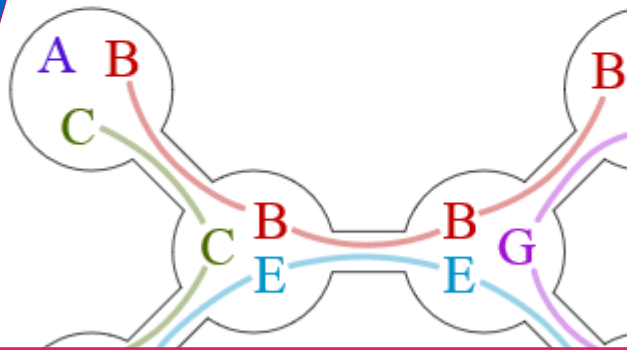


NET WORKS

Structural graph parameters

PART 3: APPLICATIONS





Bart M. P. Jansen (edited by mjd – 2021)

TU/e

Networks Training Week 2017
January 31st – February 2nd 2017, Doorn, Netherlands

Plan for today

Tackling a party planning problem

Algorithms on tree decompositions

ALGORITHMS EXPLOITING GRAPH STRUCTURE

Algorithms using tree structure

Many problems that are NP-complete in general,
can be solved efficiently on graphs of bounded treewidth

INDEPENDENT SET, DOMINATING SET, 3-COLORING,
HAMILTONIAN CYCLE, CLIQUE, FEEDBACK VERTEX SET,
DISJOINT PATHS, PARTITION INTO TRIANGLES, …

Tree-like structure can be exploited by dynamic programming

Two-step presentation:

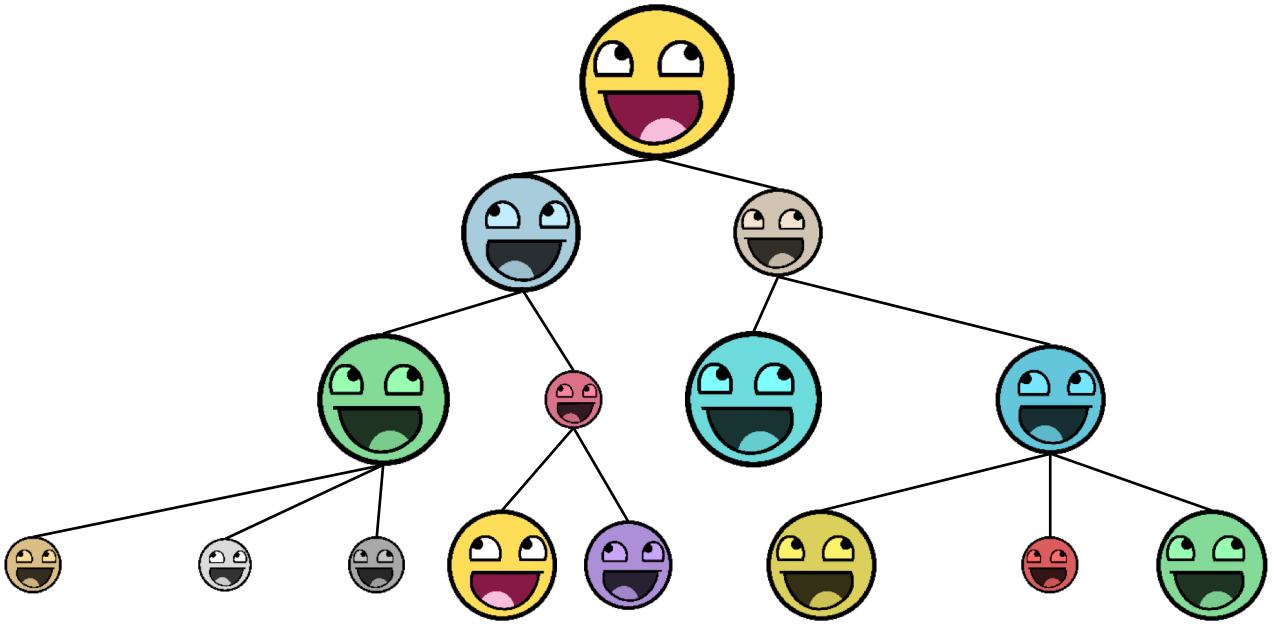
1. Dynamic programming on trees
2. Dynamic programming on tree decompositions

The party problem

We are planning a party in a hierarchical company

Each employee has a fun factor that they would contribute

Goal: *invite people that maximize the total fun factor*
Constraint: *a party is no fun when your direct boss is present*



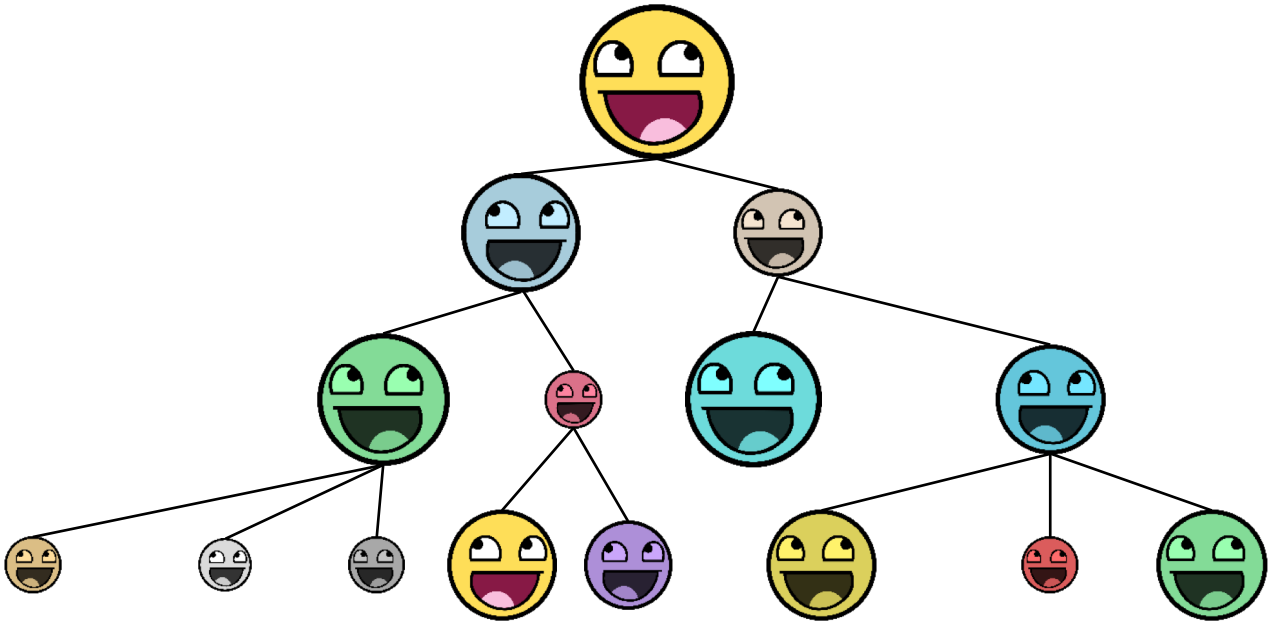
The party crowd forms an independent set

Input: A tree T with a weight $w(v)$ for each $v \in V(T)$

Task: Find independent set $S \subseteq V(T)$ maximizing $\sum_{v \in S} w(v)$

Easy if all weights are equal – weights make it interesting

Focus on computing the *value* of an optimal solution

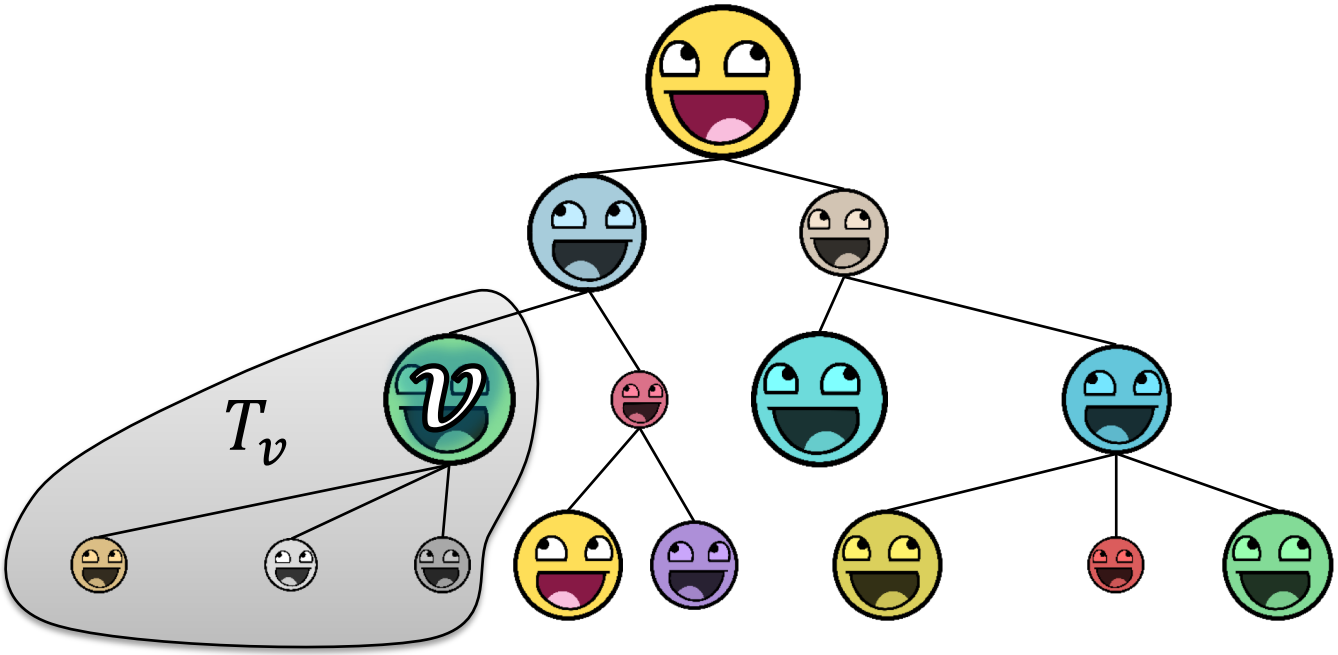


Solving the party problem

Input: A tree T with a weight $w(v)$ for each $v \in V(T)$

Task: Find independent set $S \subseteq V(T)$ maximizing $\sum_{v \in S} w(v)$

For $v \in V(T)$, let T_v denote the subtree rooted at v



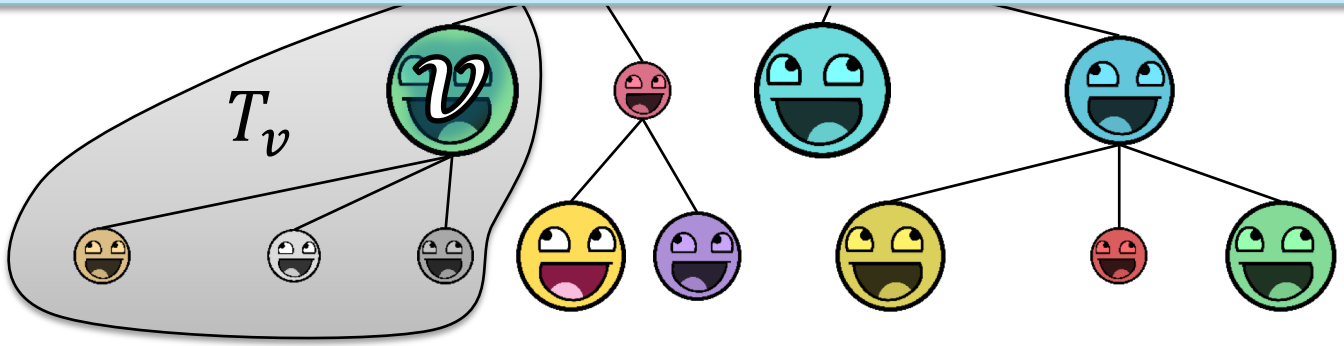
Subproblems for subtrees

$A[v] \coloneqq$ maximum weight of ind. set in T_v that contains v

$B[v] \coloneqq$ maximum weight of ind. set in T_v not containing v

For $v \in V(T)$, let T_v denote the subtree rooted at v

Observation.
If r is the root of T, then the maximum weight of an independent set in the entire tree is $\max(A[r], B[r])$



An expression using child values

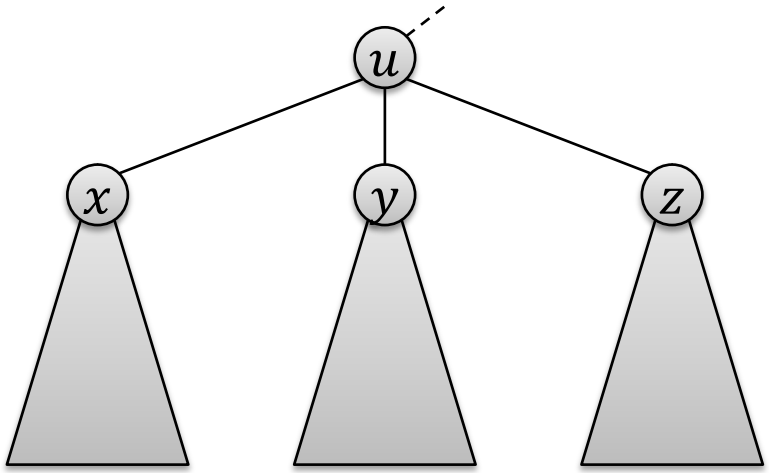
$A[v] :=$ maximum weight of ind. set in T_v that contains v

$B[v] :=$ maximum weight of ind. set in T_v not containing v

If vertex u has children x, y, z:

$$A[u] = w(u) + B[x] + B[y] + B[z]$$

$$B[u] = \max(A[x], B[x]) + \max(A[y], B[y]) + \max(A[z], B[z])$$



General recurrence

$A[v] :=$ maximum weight of ind. set in T_v that contains v

$B[v] :=$ maximum weight of ind. set in T_v not containing v

If $C(u)$ are the children of u:

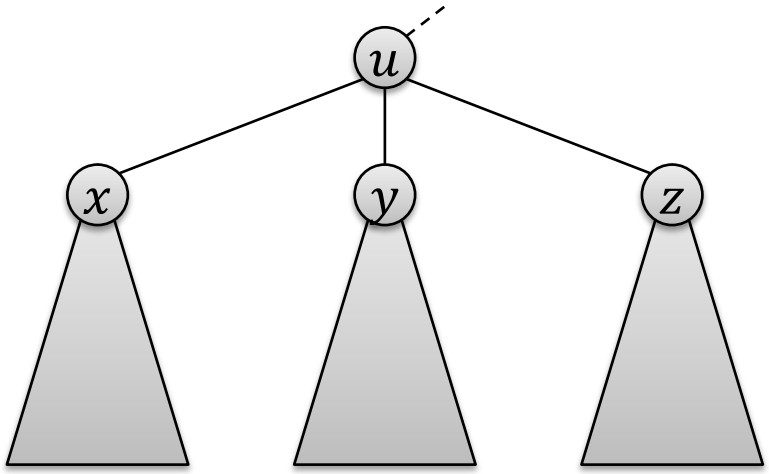
$$A[u] = w(u) + \sum_{x \in C(u)} B[x]$$

$$B[u] = \sum_{x \in C(u)} \max(A[x], B[x])$$

If u is a leaf:

$$A[u] = w(u)$$

$$B[u] = 0$$



Solving the party problem using the recurrence

Compute the values of $A[u]$ and $B[u]$ bottom-up in the tree

Theorem. An independent set of maximum weight in an n-vertex tree can be found in $O(n)$ time.

Which properties of trees did we exploit?

- The rooted tree decomposes in the input into subproblems
- Only two possible states per subproblem ($u \in S, u \notin S$)
- Easy to answer subproblems for v using values for $C(v)$

General algorithm for INDEPENDENT SET

Theorem. An independent set of maximum weight in an n-vertex tree can be found in $O(n)$ time.

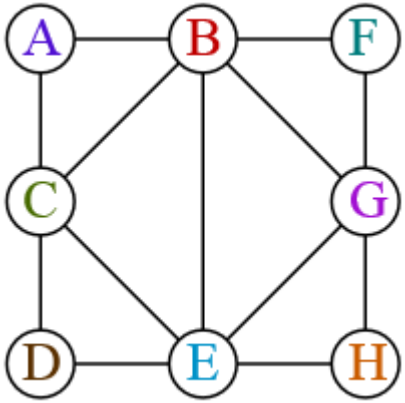
Algorithm is very efficient, but only works in trees

Goal: Adapt it to work efficiently for treelike graphs

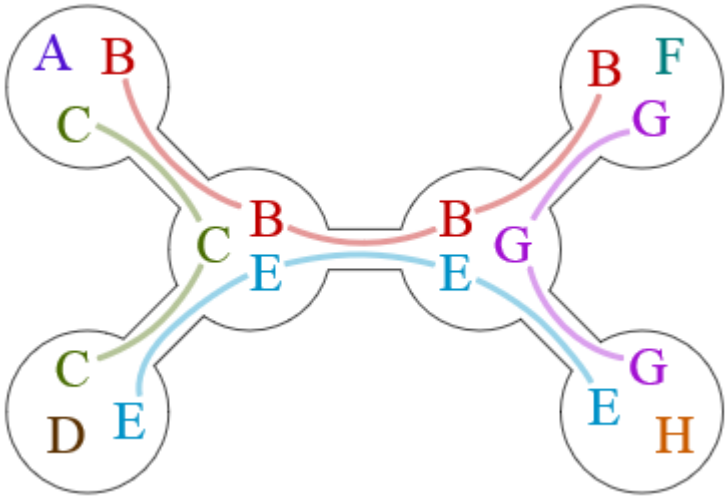
time $O(2^w \cdot n)$ for treewidth w

Tree decompositions

Graph G



Tree decomposition (T, X)



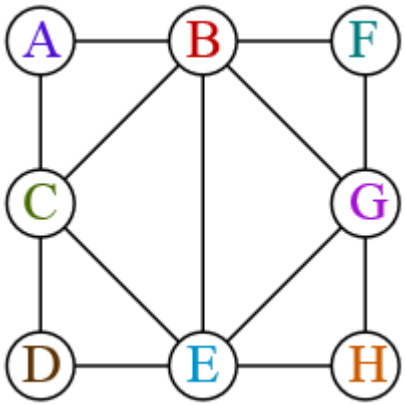
A tree decomposition of G is a pair (T, X) where T is a tree and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

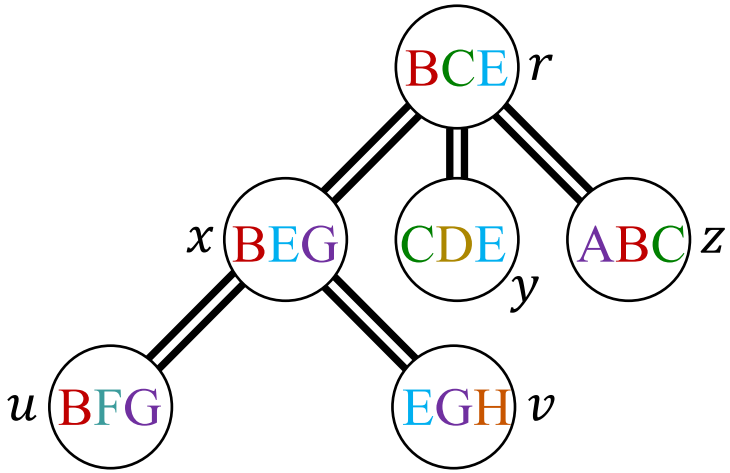
B. $\forall v \in V(G)$, bags $X^{-1}(v)$ containing v form connected subtree of T

Rooted tree decompositions

Graph G



Rooted tree decomposition (T, X)



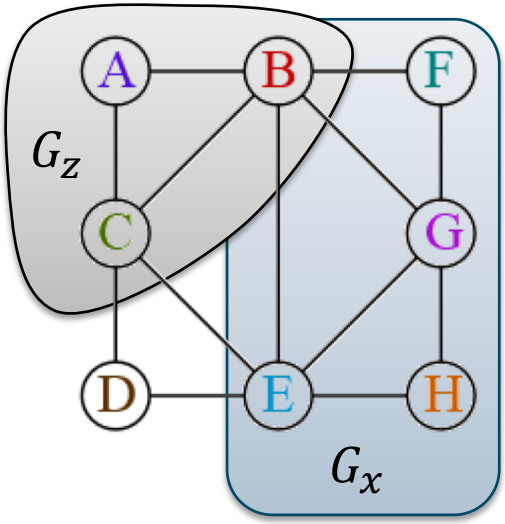
A tree decomposition of G is a pair (T, X) where T is a rooted tree and X assigns a bag $X(w) \subseteq V(G)$ to each $w \in V(T)$, such that:

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

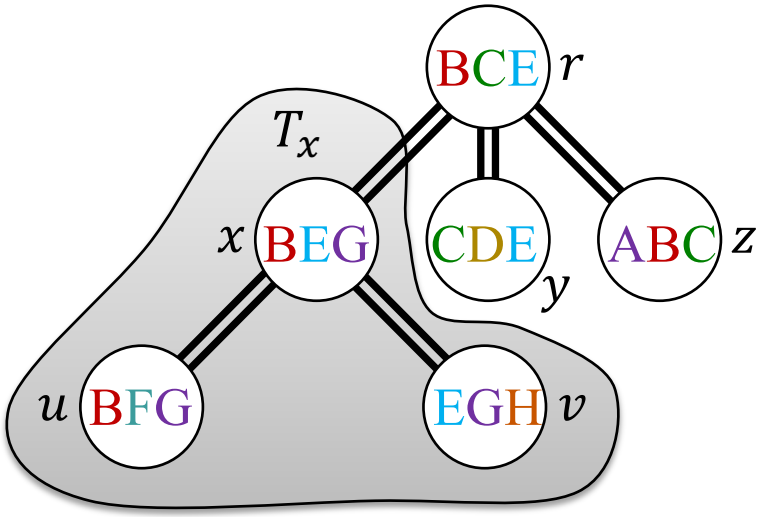
B. $\forall v \in V(G)$, bags $X^{-1}(v)$ containing v form connected subtree of T

The tree decomposes the graph

Graph G



Rooted tree decomposition (T, X)

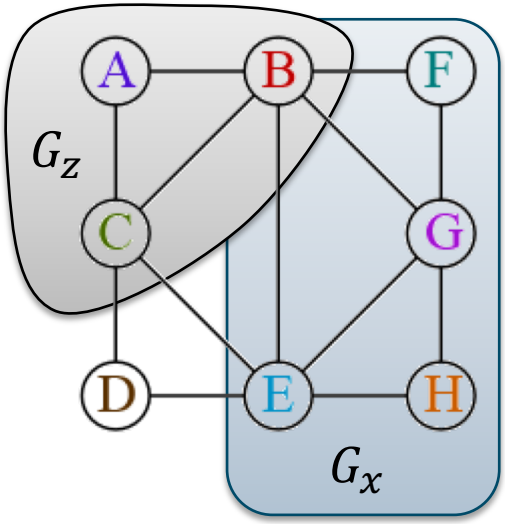


For $v \in V(T)$, let:

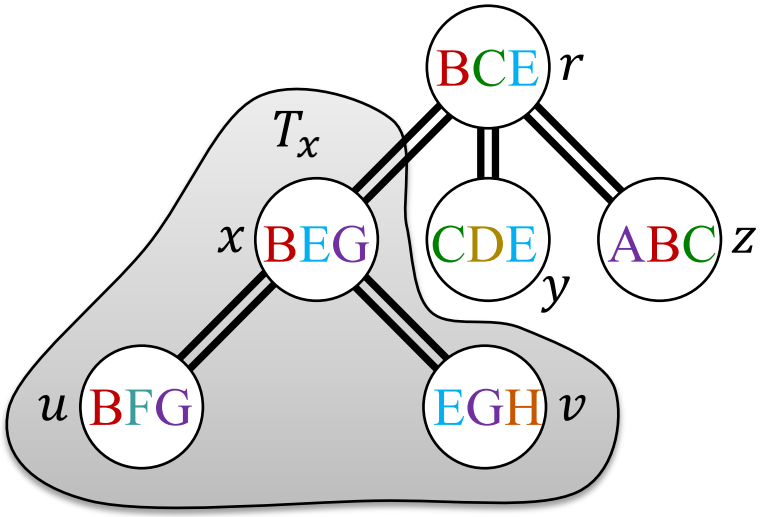
- T_v denote the subtree rooted at v
- $X(T_v) \coloneqq \bigcup_{u \in V(T_v)} X(u)$ be vertices of G in bags of T_v
- G_v be the subgraph of G induced by vertices $X(T_v)$

The tree decomposes the graph

Graph G



Rooted tree decomposition (T, X)



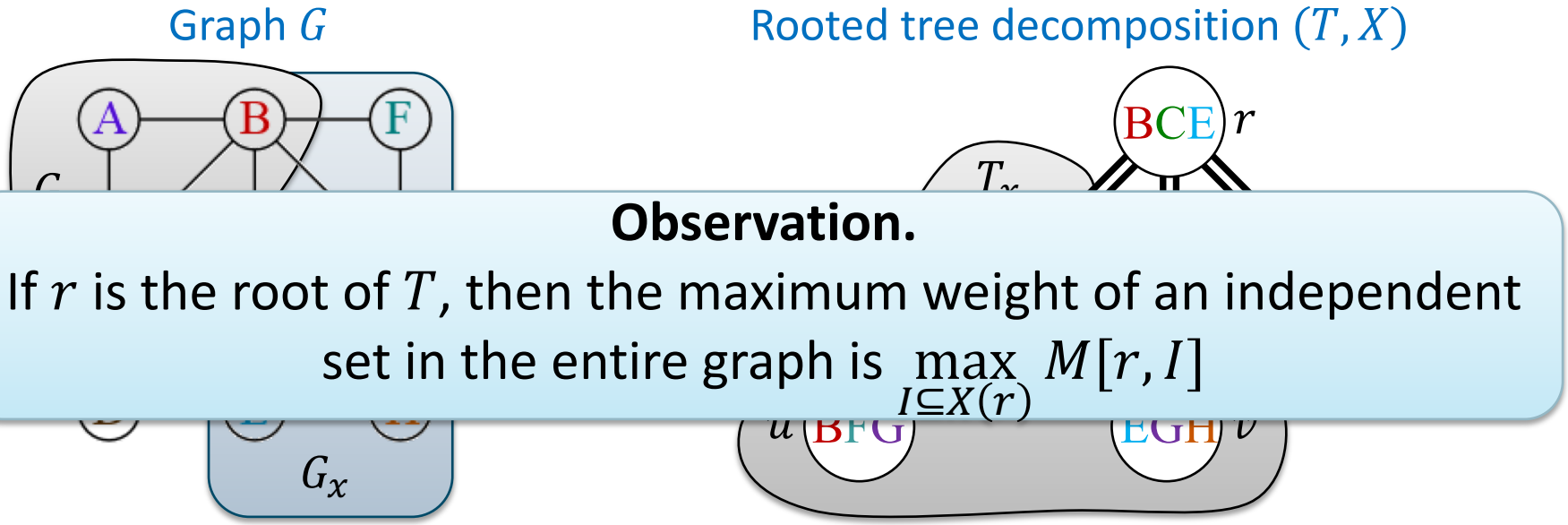
Claim. For any node $x \in V(T)$, each vertex of G_x that has a neighbor outside G_x, belongs to the bag $X(x)$.

A. $\forall uv \in E(G)$, some bag $X(w)$ contains both u and v

B. $\forall v \in V(G)$, bags $X^{-1}(v)$ containing v form connected subtree of T

Vertices in $X(T_v)$ only communicate with the outside through $X(v)$

Subproblems for nodes of the decomposition



Observation.

If r is the root of T, then the maximum weight of an independent set in the entire graph is $\max_{I \subseteq X(r)} M[r, I]$

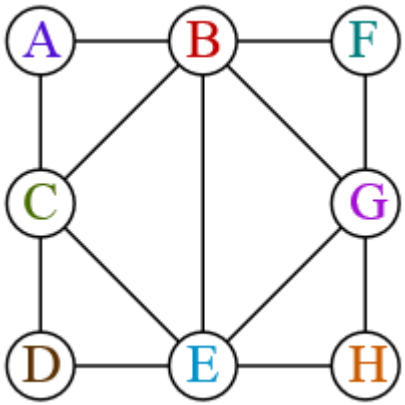
For $v \in V(T)$ and $I \subseteq X(v)$, define:

$M[v, I]$:= the maximum weight of an independent set S in G_v that satisfies $S \cap X(v) = I$

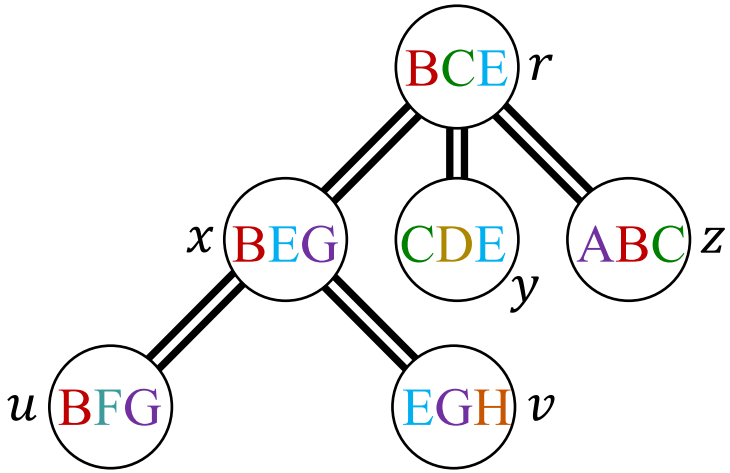
$M[v, I] := -\infty$ if I is not an independent set

Nice tree decompositions

Graph G



Rooted tree decomposition (T, X)

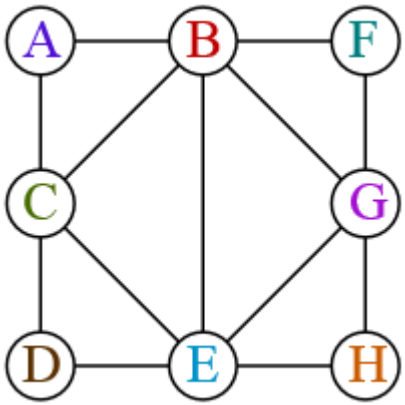


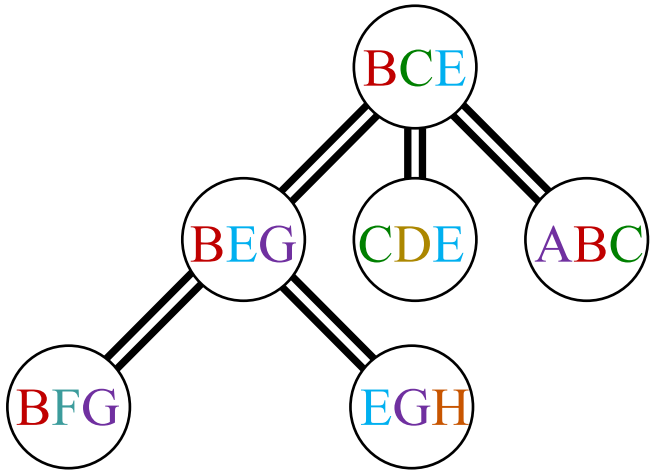
Easier to develop a recurrence for $M[\cdot,\cdot]$ values if (T, X) is nice:

1. Nodes have at most 2 children

Making a tree decomposition nice

Graph G



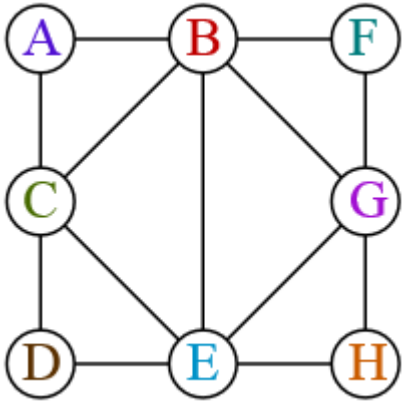


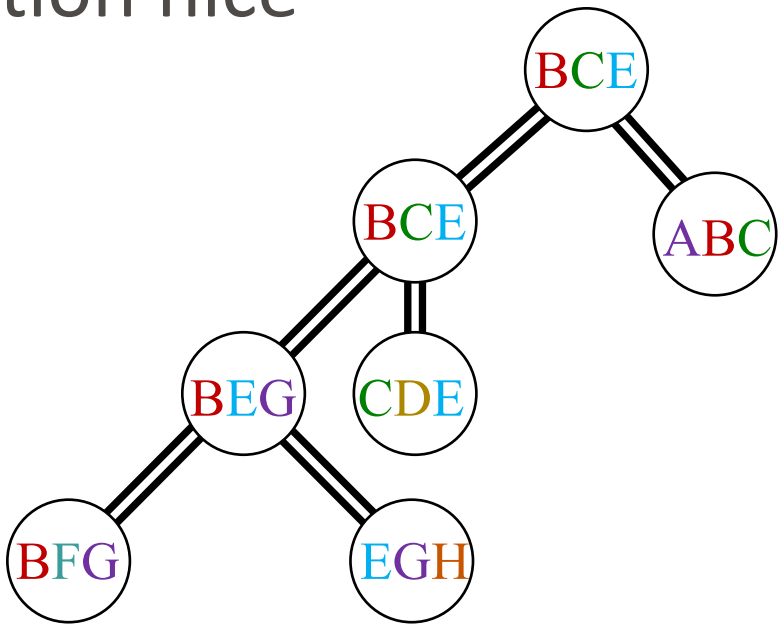
Easier to develop a recurrence for $M[\cdot,\cdot]$ values if (T, X) is nice:

1. Nodes have at most 2 children

Making a tree decomposition nice

Graph G



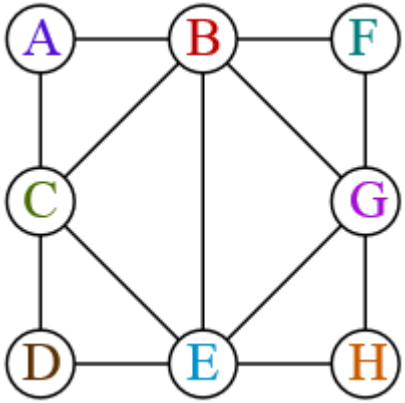


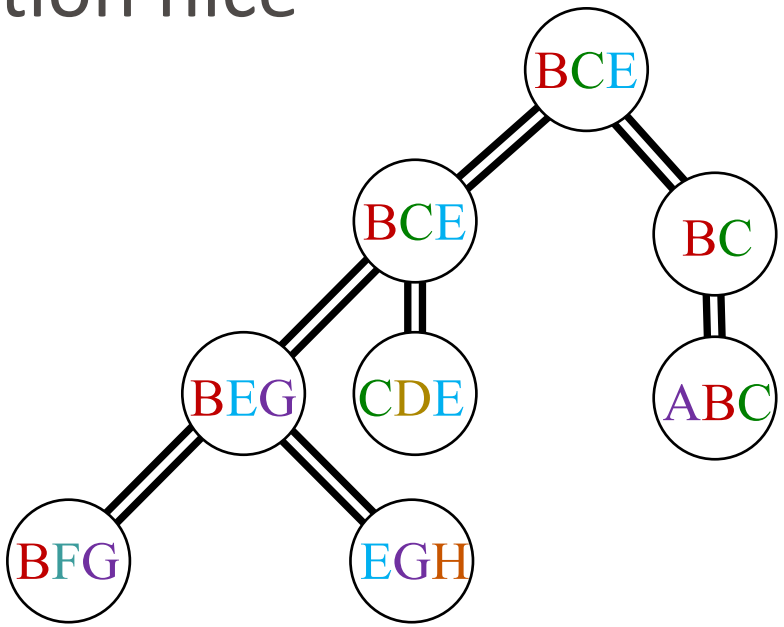
Easier to develop a recurrence for $M[\cdot,\cdot]$ values if (T, X) is nice:

1. Nodes have at most 2 children
2. If a node p has 1 child u, then $X(p)$ and $X(u)$ differ by ≤ 1 vertex

Making a tree decomposition nice

Graph G



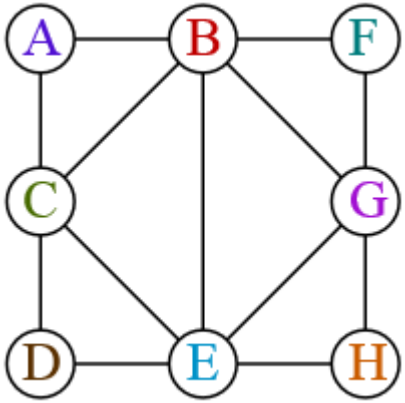


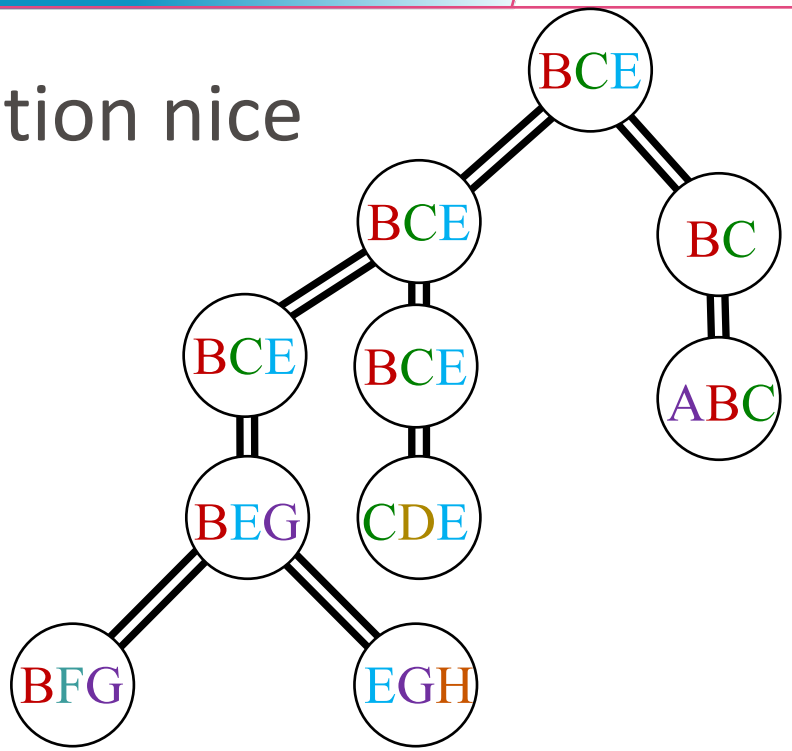
Easier to develop a recurrence for $M[\cdot,\cdot]$ values if (T, X) is nice:

1. Nodes have at most 2 children
2. If a node p has 1 child u, then $X(p)$ and $X(u)$ differ by ≤ 1 vertex
3. If a node p has 2 children u and v, then $X(p) = X(u) = X(v)$

Making a tree decomposition nice

Graph G

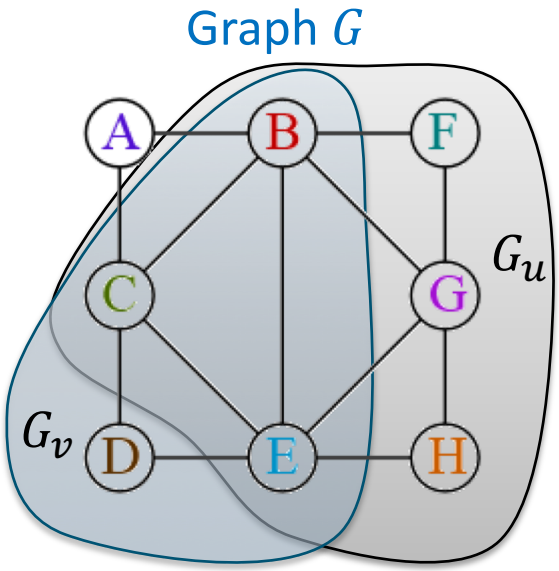


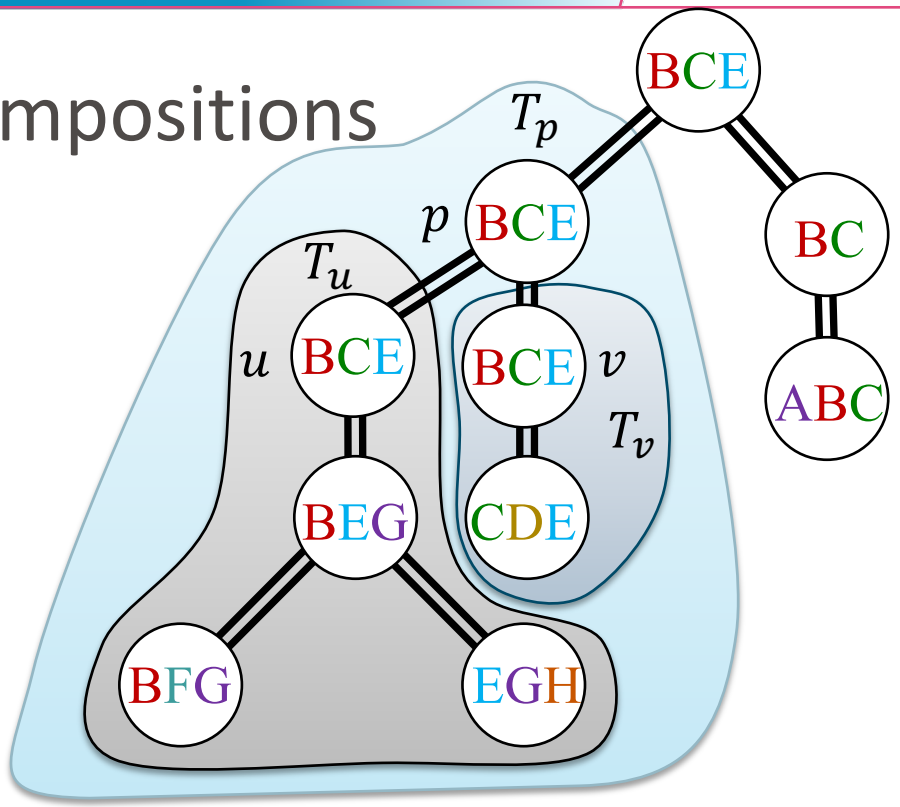


Easier to develop a recurrence for $M[\cdot,\cdot]$ values if (T, X) is nice:

1. Nodes have at most 2 children
2. If a node p has 1 child u, then $X(p)$ and $X(u)$ differ by ≤ 1 vertex
3. If a node p has 2 children u and v, then $X(p) = X(u) = X(v)$

Structure of nice decompositions





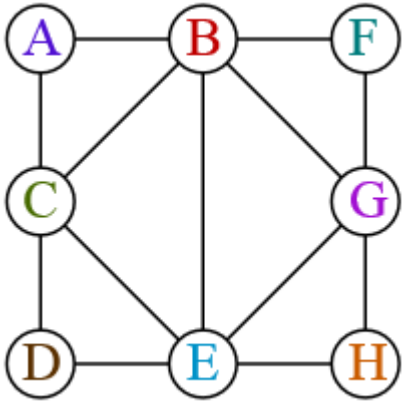
If node p has children u and v with $X(p) = X(u) = X(v)$:

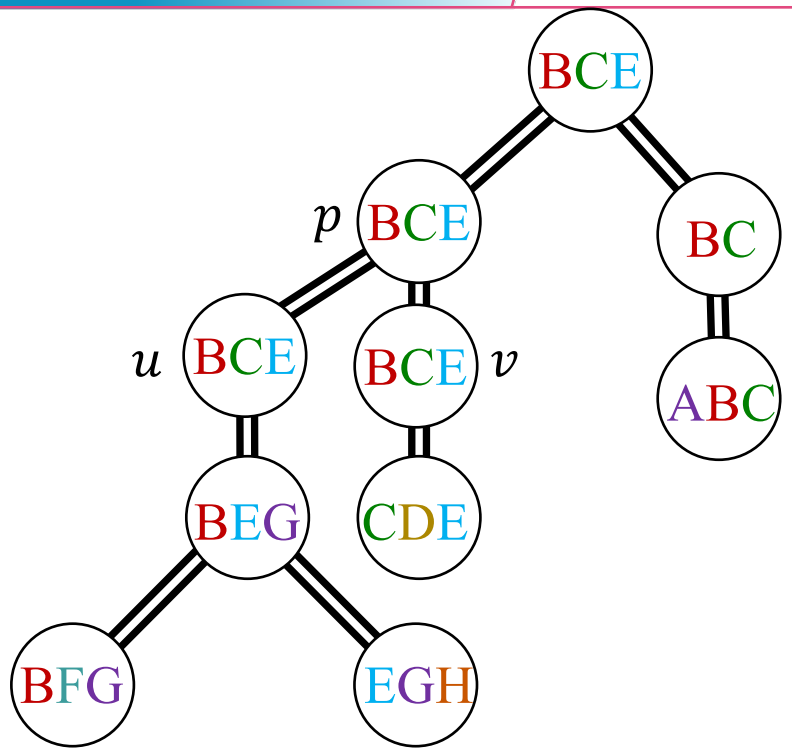
Let S_u be an independent set in G_u, and
S_v be an independent set in G_v,
such that $S_u \cap X(p) = S_v \cap X(p) = I$.

Then $S_u \cup S_v$ is an independent set in G_p of weight $w(S_u) + w(S_v) - w(I)$.

A recurrence for M

Graph G



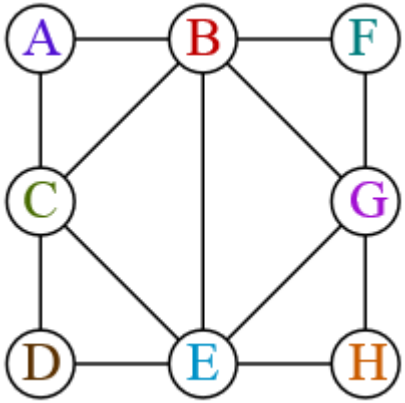


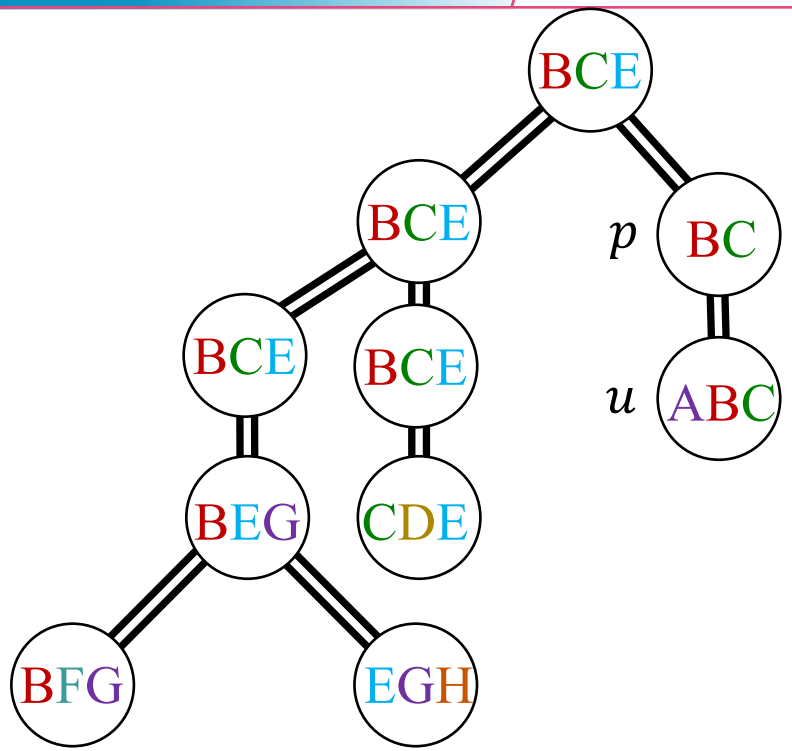
If node p has children u and v with $X(p) = X(u) = X(v)$:

$$M[p, I] = M[u, I] + M[v, I] - \sum_{a \in I} w(a)$$

A recurrence for M

Graph G



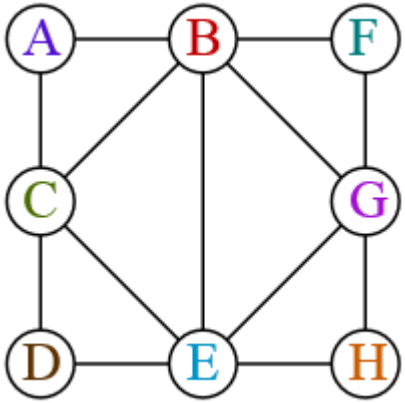


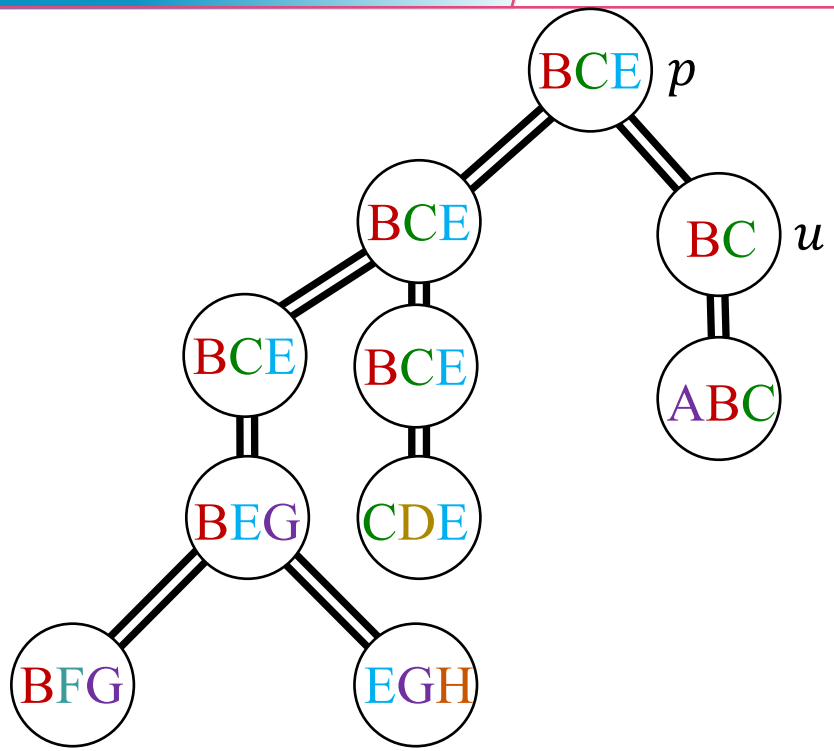
If node p has 1 child u with $X(p) = X(u) \setminus \{a\}$, then:

$$M[p, I] = \begin{cases} \max(M[u, I], M[u, I \cup \{a\}]) & \text{if } I \cup \{a\} \text{ is independent} \\ M[u, I] & \text{otherwise} \end{cases}$$

A recurrence for M

Graph G



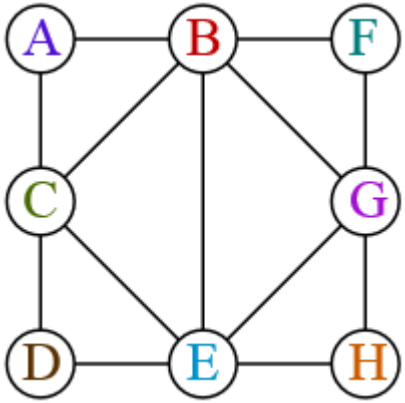


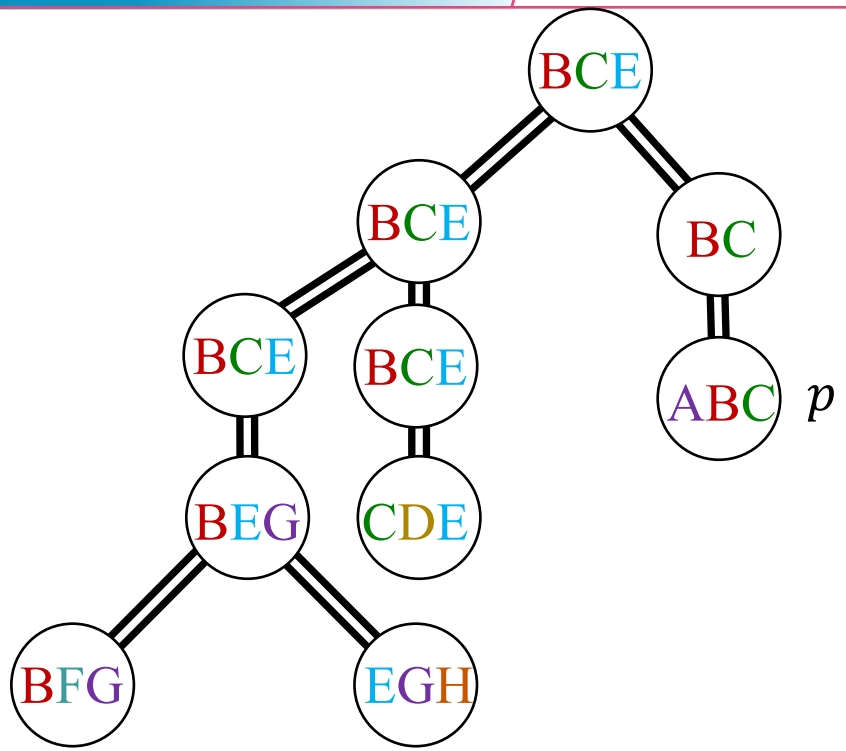
If node p has 1 child u with $X(p) = X(u) \cup \{a\}$, then:

$$M[p, I] = \begin{cases} w(a) + M[u, I \setminus \{a\}] & \text{if } a \in I \wedge I \text{ independent} \\ -\infty & \text{if } a \in I \wedge I \text{ not independent} \\ M[u, I] & \text{if } a \notin I \end{cases}$$

A recurrence for M

Graph G





If node p is a leaf:

$$M[p, I] = \begin{cases} \sum_{a \in I} w(a) & \text{if } I \text{ is independent} \\ -\infty & \text{if } I \text{ is not independent} \end{cases}$$

Solving INDEPENDENT SET using the recurrence

Compute the values of $M[\cdot,\cdot]$ bottom-up in the tree

- For each $u \in V(T)$ there are $2^{|X(u)|}$ subproblems $M[u, I]$
  - On a tree decomposition of width w, bag size is $\leq w + 1$
  - At most $|V(T)| \cdot 2^{w+1}$ subproblems in total
  - n-vertex graph has nice tree decomp's with $O(n)$ nodes
- Using values for children, computation takes $O(1)$ steps
- Answer is given by $\max_{I \subseteq X(r)} M[r, I]$

Theorem. Given a width-w tree decomposition of an n-vertex graph G, a maximum weight independent set in G can be found in $O(2^w \cdot n)$ time.

Algorithms on bounded-treewidth graphs

Two-step approach:

1. Compute a small-width tree decomposition of the input
2. Solve the problem by bottom-up dynamic programming

Task (1) is not easy: TREEWIDTH is NP-complete

- Solvable in ☹(n) for fixed target width w
- Heuristics exist & suboptimal dec's give optimal solutions

Task (2) is sometimes painful, but can be automated

A meta-theorem

Very general statement:

problems that can be stated in a certain logic,
can be solved efficiently on graphs of bounded treewidth

Stated in terms of Monadic Second Order logic

Boolean logic: no quantification over variables

$$\big(a \to (b \vee c)\big) \to (\neg a \vee (b \vee c))$$

First-order logic: quantify over single objects

$$\exists x\colon human(x) \wedge mortal(x)$$

Monadic second-order logic: can also quantify over sets

$$\exists S, \forall x\colon human(x) \to x \in S$$

Monadic second-order logic on graphs

MSO on graphs allows logic formulas consisting of:

- Logical connectives $\{\wedge, \vee, \rightarrow, \leftrightarrow, \neg\}$
- Quantification over objects and sets $\exists x, \forall x, \exists S, \forall S$
- $Vtx(x), Edge(e), Adj(x, y), Inc(x, e), =, \in$

Universe of discourse contains vertices and edges

Examples of MSO expressions for graphs

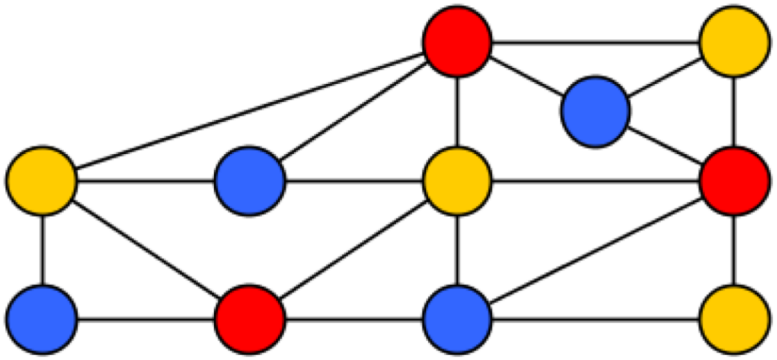
1. “There is a vertex of degree at least three”

$$\exists x \exists u \exists v \exists w : u \neq v \wedge u \neq w \wedge v \neq w \wedge Adj(x,u) \wedge Adj(x,v) \wedge Adj(x,w)$$

Examples of MSO expressions for graphs

2. "The vertices can be colored red, green, blue such that the endpoints of each edge receive different colors"

$$\exists R \exists G \exists B:$$
$$\big(\forall x: Vtx(x) \rightarrow (x \in R \vee x \in G \vee x \in B)\big) \wedge$$
$$(\forall x \forall y: Adj(x,y) \rightarrow$$
$$\neg((x \in R \wedge y \in R) \vee (x \in G \wedge y \in G) \vee (x \in B \vee y \in B)))$$



Examples of MSO expressions for graphs

3. “There is an independent set of 4 vertices”

$$\exists x_1 \dots x_4 : \bigwedge_{i=1}^{4} Vtx(x_i) \wedge \bigwedge_{1 \le i < j \le 4} (x_i \neq x_j \wedge \neg Adj(x_i, x_j))$$

Exercise: Formulating properties in MSO

Which formula ϕ expresses "the graph is connected"?

(*Hint*: negate "G has 2 different connected components")

Courcelle's theorem

Courcelle's theorem. Given a formula ϕ in monadic second-order logic and an n-vertex graph G of treewidth w, one can determine whether $G \models \phi$ in time $f(|\phi|, w) \cdot n$.

Extension to optimization problems:

Courcelle's theorem. Given a formula ϕ in monadic second-order logic and an n-vertex graph G of treewidth w, that contains a free set variable X, one can find a maximum-size X such that $(G, X) \models \phi$ in time $f(|\phi|, w) \cdot n$.

Same holds for cliquewidth if ϕ does not quantify over edge sets

Summary

Graph parameters have algorithmic & combinatorial applications

Solve problems on treelike graphs by dynamic programming

Courcelle's theorem links this to monadic second-order logic

Treewidth shows up in problems where you don't expect it

Use the parameter hierarchy to guide the attack on a problem!