CSE 462

Computing Treewidth

Presented By:

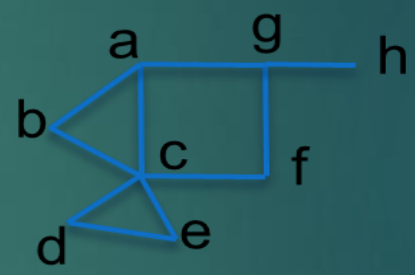
FAIYAZ RASHID NABIL (1405069)

What is Treewidth?

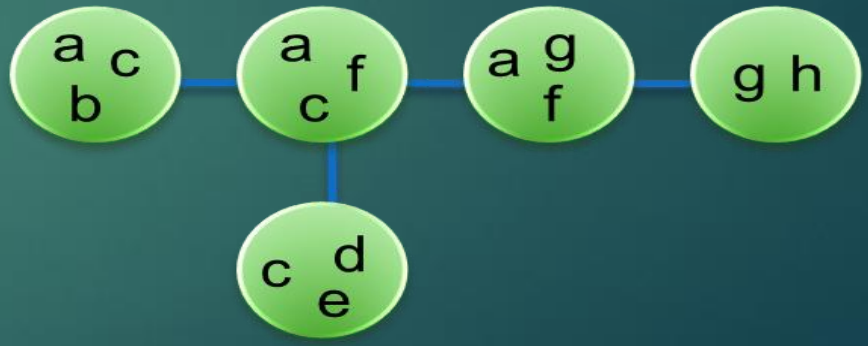
- The treewidth of an undirected graph is an integer number which specifies how far the graph is from being a tree.
- Treewidth captures how similar a graph is to a tree
- The smaller the treewidth, the more tree-like the graph is.
- Treewidth is a useful tool to solve graph problems for several special instances, both for theory and for practice.
- Theoretical results with additional ideas give practical algorithms that work well in many cases.

Tree decomposition

- Given a graph G = (V, E), a tree decomposition is a pair (X, T), where X = {X_1, ..., X_n} is a family of subsets of V, and T is a tree whose nodes are the subsets X_i, satisfying the following properties:

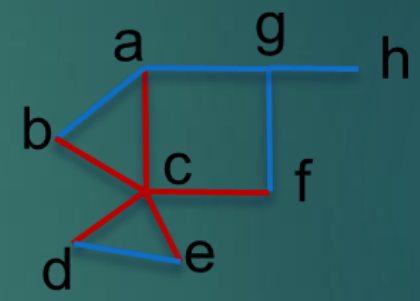


- **Node Coverage:**
  each graph vertex is associated with at least one tree node called **bag**
- **Edge Coverage:**
  for all edges {v,w}: there is a set containing both v and w.
- **Coherence:**
  for every v: the nodes that contain v form a connected subtree.

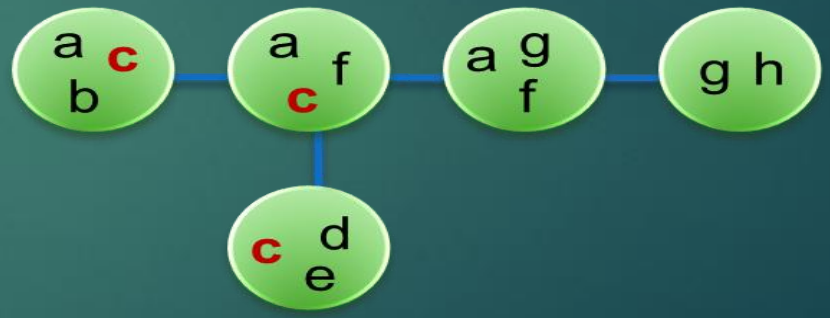


Tree decomposition

- Given a graph G = (V, E), a tree decomposition is a pair (X, T), where X = {X_1, ..., X_n} is a family of subsets of V, and T is a tree whose nodes are the subsets X_i, satisfying the following properties:

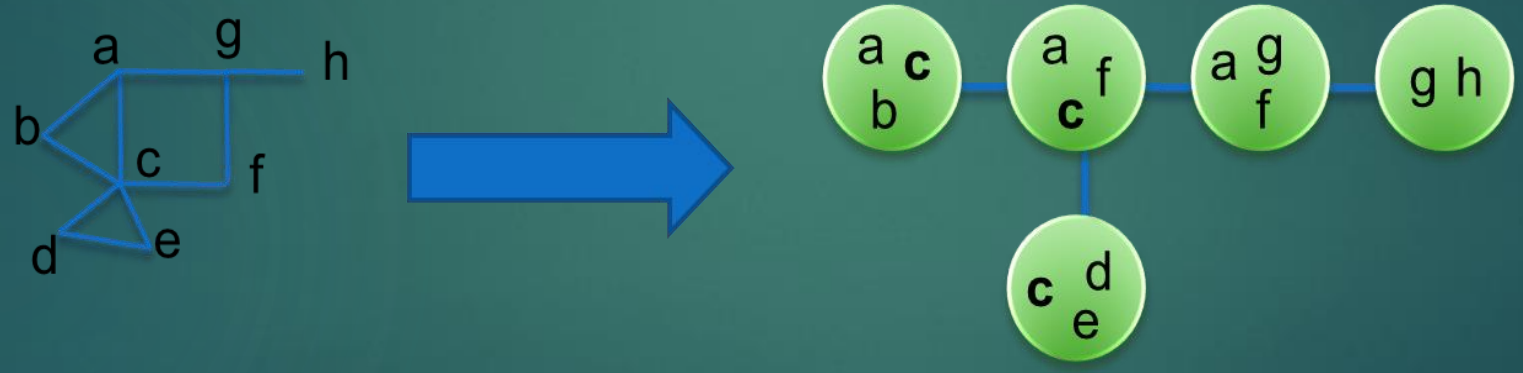


- **Node Coverage:**
  each graph vertex is associated with at least one tree node called **bag**
- **Edge Coverage:**
  for all edges {v,w}: there is a set containing both v and w.
- **Coherence:**
  for every v: the nodes that contain v form a connected subtree.

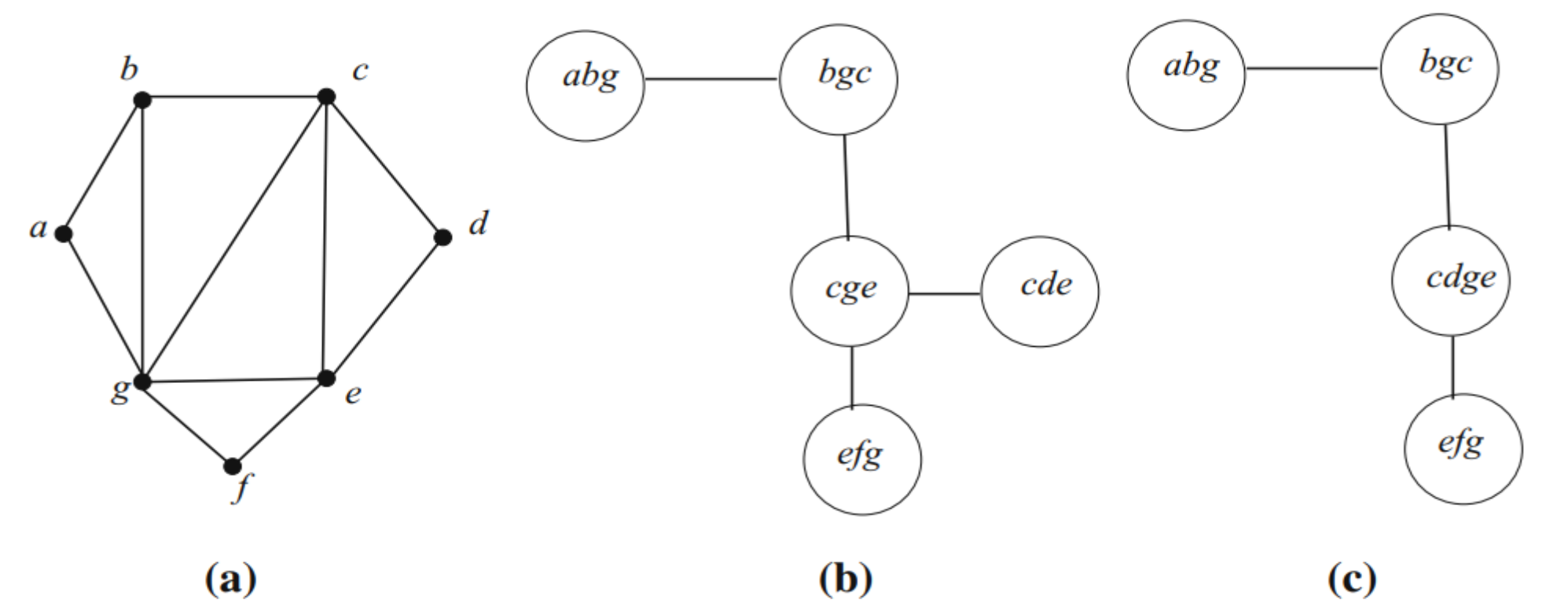


Tree decomposition

- Tree decomposition : mapping of a graph into a tree.
- In machine learning, it is also called junction tree, clique tree, or join tree.



Tree decomposition



- Figures (b) and (c) illustrates two tree decompositions of the graph in Figure (a).
- (a) is a chordal graph (b) is a clique tree of the graph (c) is not a clique tree.

Definition of treewidth

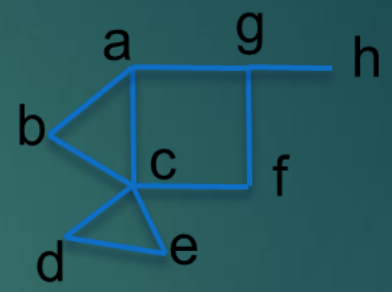
- **Width of tree decomposition:**

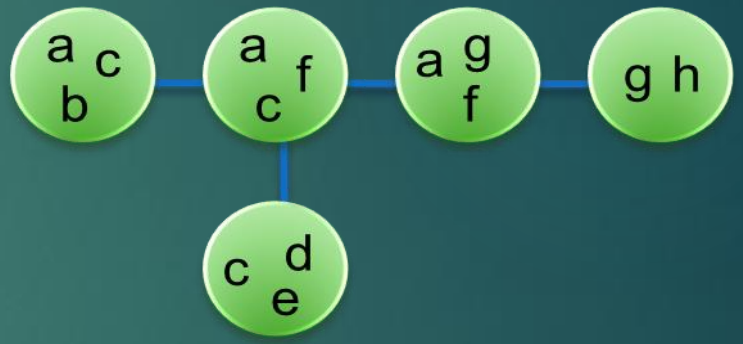
$$\max_{i \in I} |X_i| - 1$$

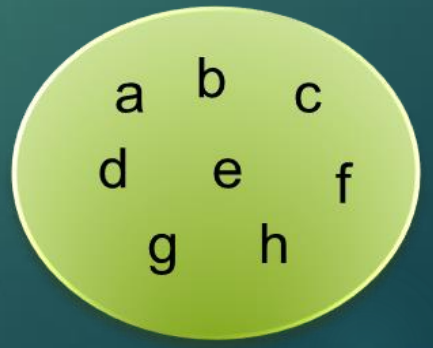
"the size of the largest bag minus one."

- **Treewidth:**

tw(*G*) = minimum width over all tree decompositions of *G*.







Why do we need to compute treewidth?

- Due to the roles the treewidth plays in an enormous number of fields, different practical and theoretical algorithms computing the treewidth of a graph were developed.

- Depending on the application on hand, one can prefer better approximation ratio, or better dependence in the running time from the size of the input or the treewidth.

Applications of treewidth

- Graph minor theory (Robertson and Seymour)
- Optimization
- Probabilistic networks
- Telecommunication Network Design
- VLSI-design
- Natural Language Processing
- Compilers
- Maximum independent set
- Hamiltonian circuit
- Vertex colour problem
- Edge colour problem
- Graph Isomorphism

Basic Terminology

- **Fixed-Parameter Tractable (FPT)- S**ome problems can be solved by algorithms that are exponential only in the size of a fixed parameter while polynomial in the size of the input. Such an algorithm is called a fixed-parameter tractable (fpt) algorithm, because the problem can be solved efficiently for small values of the fixed parameter.
- **Minor of a graph-** An undirected graph *H* is called a minor of the graph *G* if *H* can be formed from *G* by deleting edges and vertices and by contracting edges.
- **Minor-Monotone Parameter-** A graph parameter f is minor-monotone if f(H) ≤ f(G) whenever H is a minor of a graph G.

Basic Terminology

- **Minor-Closed Graph-** A class of graphs G' is minor-closed if for every G ∈ G,' every minor H of G belongs to G'.
- **Proper Minor-** A proper minor of a graph G is a minor unequal to G.
- **Forbidden Minor-** A graph H is a forbidden minor of the class of graphs $\mathcal{G}$, if $H \notin \mathcal{G}$ but every proper minor of H is.
- **Balanced Separators-** A balanced separator breaks the graph into pieces whose weights constitute only a constant fraction (½ or 2/3) of the original weight of the graph.

Computing Treewidth

- Previously, we assumed a tree decomposition of small width is given as part of the input.
- How fast can we find such a decomposition?
- It is NP-hard !
- So, we need to resort to FPT or approximation algorithms.

Treewidth Problem

- **Input-** a graph G and an integer k.
- **Output-** to determine whether the treewidth of G is at most k.
- Practical algorithms...

  Heuristics (works often well)

  Upper bound.

  Lower bound.

  Pre processing

  Transform your input into a smaller equivalent input

Graph Minor Theorem

Corollary 6.11. *For every minor-closed graph class $\mathcal{G}$, there exists a* finite *set* $\mathrm{Forb}(\mathcal{G})$ *of graphs with the following property: for every graph G, graph G belongs to $\mathcal{G}$ if and only if there does not exist a minor of G that is isomorphic to a member of* $\mathrm{Forb}(\mathcal{G})$.

- The **graph minor theorem** (also called the **Robertson–Seymour theorem**) states that the undirected graphs, partially ordered by the graph minor relationship, form a ordering such that any infinite sequence of elements $x_0, x_1, x_2, \ldots$ from X contains an increasing pair $x_i \leq x_j$ with $i < j$
- Every family of graphs that is closed under minors can be defined by a finite set of forbidden minors.

Computing Treewidth

- Following from the results of the Graph Minor theorem, we can say treewidth is fixed-parameter tractable.
- More precisely, treewidth is a minor-monotone parameter in the following sense:

  for every graph G and its minor H, it holds that $tw(H) \leq tw(G)$
- Hence if we define G_k to be the class of graphs of treewidth at most k, then G_k is closed under taking minors (minor-closed graph)
- There exists a set of forbidden minors $Forb(G_k)$, which depends on k only, such that a graph has treewidth k if and only if it does not contain any graph from $Forb(G_k)$ as a minor.

Computing Treewidth

Theorem 6.12 (Robertson and Seymour). *There exists a computable function f and an algorithm that, for given graphs H and G, checks in time $f(H)|V(G)|^3$ whether $H \leq_m G$.*

- Every minor-closed class can be recognized in polynomial time.
- We can apply this algorithm to each H ∈ Forb(G_k) separately.
- We can verify in f(H) .| V (G) | ^{3} time whether it is contained in G as a minor.

Computing Treewidth

- Since Forb(G_k) depends only on k, both in terms of its cardinality and the maximum size of a member, this algorithm runs in $g(k).|V(G)|^3$ for some function g.
- This gives only a nonuniform FPT algorithm-

Because graph minor theorem does not provide us any means of constructing the family G_k, or even bounding its size.

Alternative Approach

- One could approach the Treewidth problem using the graph searching game.
- This direction quite easily leads to an algorithm with running time $n^{O(k)}$

Exact Algorithm

- It is possible to obtain a uniform, constructive algorithm for the Treewidth problem.
- But this requires much more involved techniques and technical effort.
- We can show that Treewidth is FPT(fixed-parameter tractable) using Bodlaender algorithm.

Theorem 7.17 ([45]). *There exists an algorithm that, given an n-vertex graph G and integer k, runs in time $k^{\mathcal{O}(k^3)} \cdot n$ and either constructs a tree decomposition of G of width at most k, or concludes that $\mathrm{tw}(G) > k$.*

- It is the fastest known exact parameterized algorithm, meaning that it computes the exact value of the treewidth.

Why do we need approximation?

- Here, $k^{\mathcal{O}(k^3)}$ is a dominating factor in most applications.
- We would like to obtain better parameter dependence than it.
- Most other works follow the direction of approximating treewidth .
- We relax the requirement that the returned decomposition has width at most k by allowing the algorithm to output a decomposition with a slightly larger width (> k)

Approximation Algorithm

- The classical approximation algorithm for treewidth that originates in the work of Robertson and Seymour.

Theorem 7.18. *There exists an algorithm that, given an n-vertex graph G and integer k, runs in time $\mathcal{O}(8^k k^2 \cdot n^2)$ and either constructs a tree decomposition of G of width at most $4k+4$, or concludes that $\mathrm{tw}(G) > k$.*

- The running time of algorithm of this theorem depends quadratically on the size of the graph, whereas for the previous theorem this dependence is linear.

Issues with this theorem

- Since the running time of most dynamic-programming algorithms on tree decompositions depends linearly on the graph size, this particular difference(quadratic/linear) is important in applications.
- Here, the n^2 factor is a bottleneck. $\mathcal{O}(8^k k^2 \cdot n^2)$
- To cope with this issue, very recently

a 5-approximation algorithm running in time $2^{O(k)} \cdot n$ has been developed with treewidth at most 5k + 4 (Bodlaender et al.) in 2016

a 2-approximation algorithm running in time $2^{O(k)}. n$ has been developed with treewidth at most 2k + 1 (Korhonen) in 2021

- The techniques needed to obtain such a result are beyond the scope of this book.

Importance of this theorem

- It provides an efficient way of finding a reasonable tree decomposition of a graph.
- The general strategy employed in this algorithm has been reused multiple times in approximation algorithms for other structural graph parameters.

Crux of this theorem

- It is easy to see that every n-vertex tree T contains a vertex v such that every connected component of T - v has at most n/2 vertices.
- A similar fact can be proved for graphs of bounded treewidth.
- Graphs of treewidth at most k have balanced separators of size at most k + 1
- It is possible to remove k + 1 vertices from the graph so that every connected component left is "significantly" smaller than the original graph.

Crux of this theorem

- The algorithm decomposes the graph recursively.
- At each step we try to decompose some part of a graph that has only ck vertices on its boundary, for some constant c.
- The crucial step is to find a small separator of the currently processed part H with the following property: the separator splits H into two pieces H_1, H_2 so that the boundary of H gets partitioned evenly between H_1 and H_2.

Crux of this theorem

- If for some $a > 1$, each of H_1 and H_2 contains only, say, ck/a vertices of $\partial(H)$, then we have $|\partial(H_1)|, |\partial(H_2)| \leq ck/a + (k+1)$
- The summand $k+1$ comes from the separator.
- If $ck/a + (k+1) \leq ck$, then we can recurse into pieces H_1 and H_2.

Running Time of Different Treewidth Algorithms

Approximation	f(k)	g(n)	reference
exact	$O(1)$	$O(n^{k+2})$	Arnborg, Corneil & Proskurowski (1987)
$4k+3$	$O(3^{3k})$	$O(n^2)$	Robertson & Seymour (1995)
$8k+7$	$2^{O(k \log k)}$	$n \log^2 n$	Reed (1992)
$5k+4$	$2^{O(k \log k)}$	$n \log n$	Lagergren (1996)
exact	$k^{O(k^3)}$	$O(n)$	Bodlaender (1996)
$O(k \cdot \sqrt{\log k})$	$O(1)$	$n^{O(1)}$	Feige, Hajiaghayi & Lee (2008)
$4.5k+4$	2^{3k}	n^2	Amir (2010)
$\frac{11}{3}k+4$	$2^{3.6982k}$	$n^3 \log^4 n$	Amir (2010)
exact	$O(1)$	$O(1.7347^n)$	Fomin, Todinca & Villanger (2015)
$3k+2$	$2^{O(k)}$	$O(n \log n)$	Bodlaender et al. (2016)
$5k+4$	$2^{O(k)}$	$O(n)$	Bodlaender et al. (2016)
k^2	$O(k^7)$	$O(n \log n)$	Fomin et al. (2018)
$5k+4$	$2^{8.765k}$	$O(n \log n)$	Belbasi & Fürer (2021a)
$2k+1$	$2^{O(k)}$	$O(n)$	Korhonen (2021)
$5k+4$	$2^{6.755k}$	$O(n \log n)$	Belbasi & Fürer (2021b)

Here,
k is the treewidth

n is the number of vertices of an input graph G.

Each of the algorithms outputs in time f(k).g(n).

A decomposition of width given in the Approximation column

Computing treewidth

Maximum Cardinality Search Heuristic

- Simple mechanism to make a permutation of the vertices of an undirected graph
- Let the visited degree of a vertex be its number of visited neighbours
- Pseudocode for MCS:

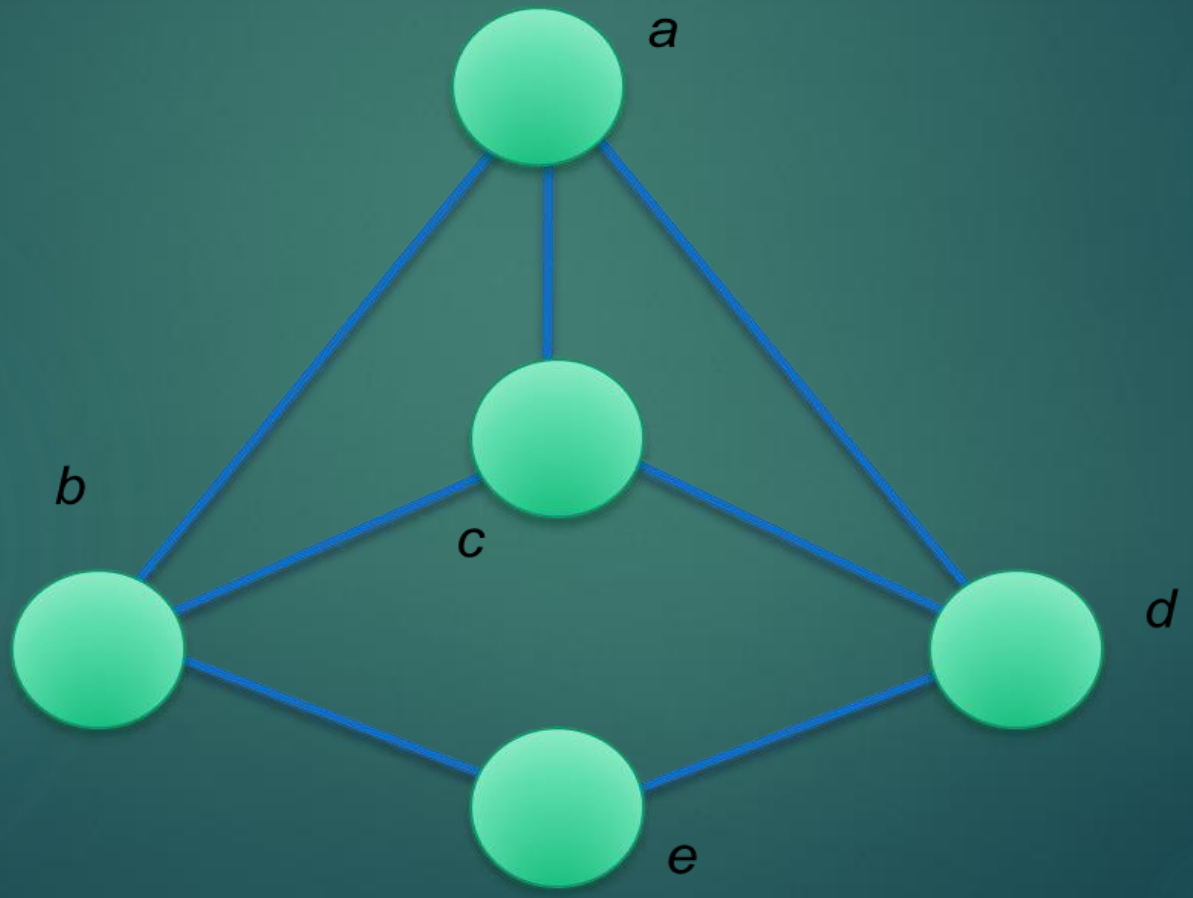
  Repeat

    Visit an unvisited vertex that has the *largest visited degree*

  Until all vertices are visited

Computing treewidth

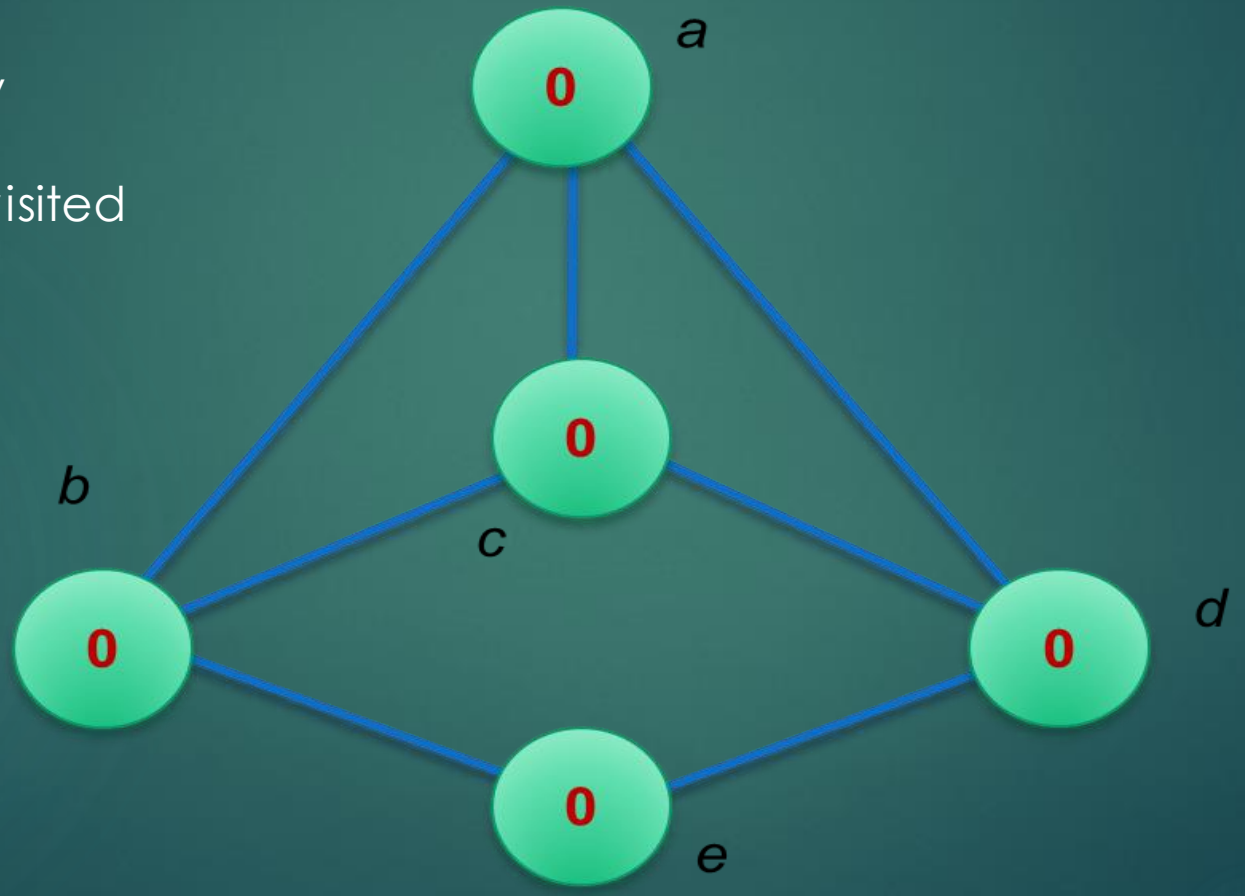
Maximum Cardinality Search: a 4-clique with one subdivision



Computing treewidth

Maximum Cardinality Search: a 4-clique with one subdivision

We can start at any vertex:
each vertex has 0 visited neighbours

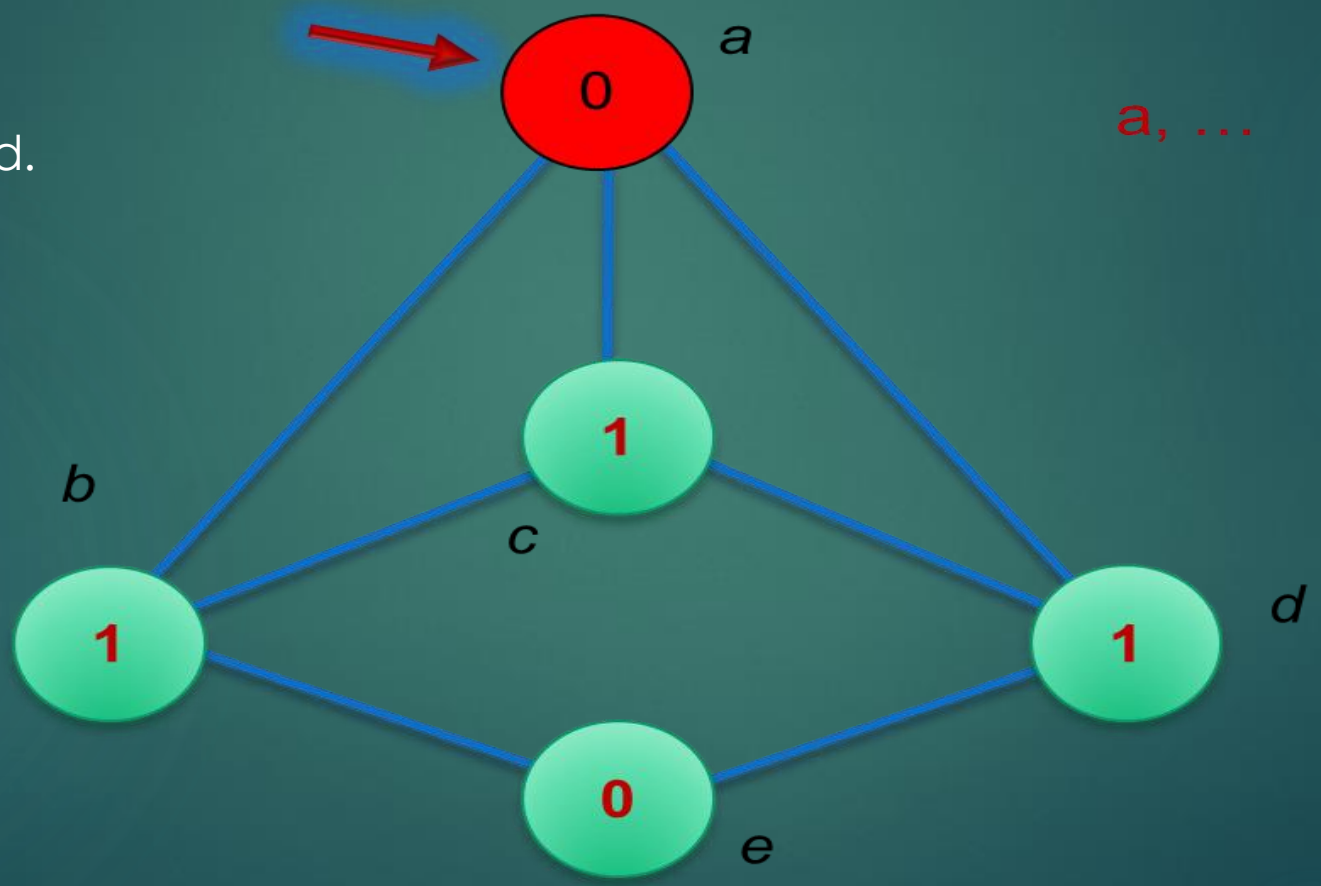


Computing treewidth

Maximum Cardinality Search: a 4-clique with one subdivision

The next vertex must be b, c, or d.

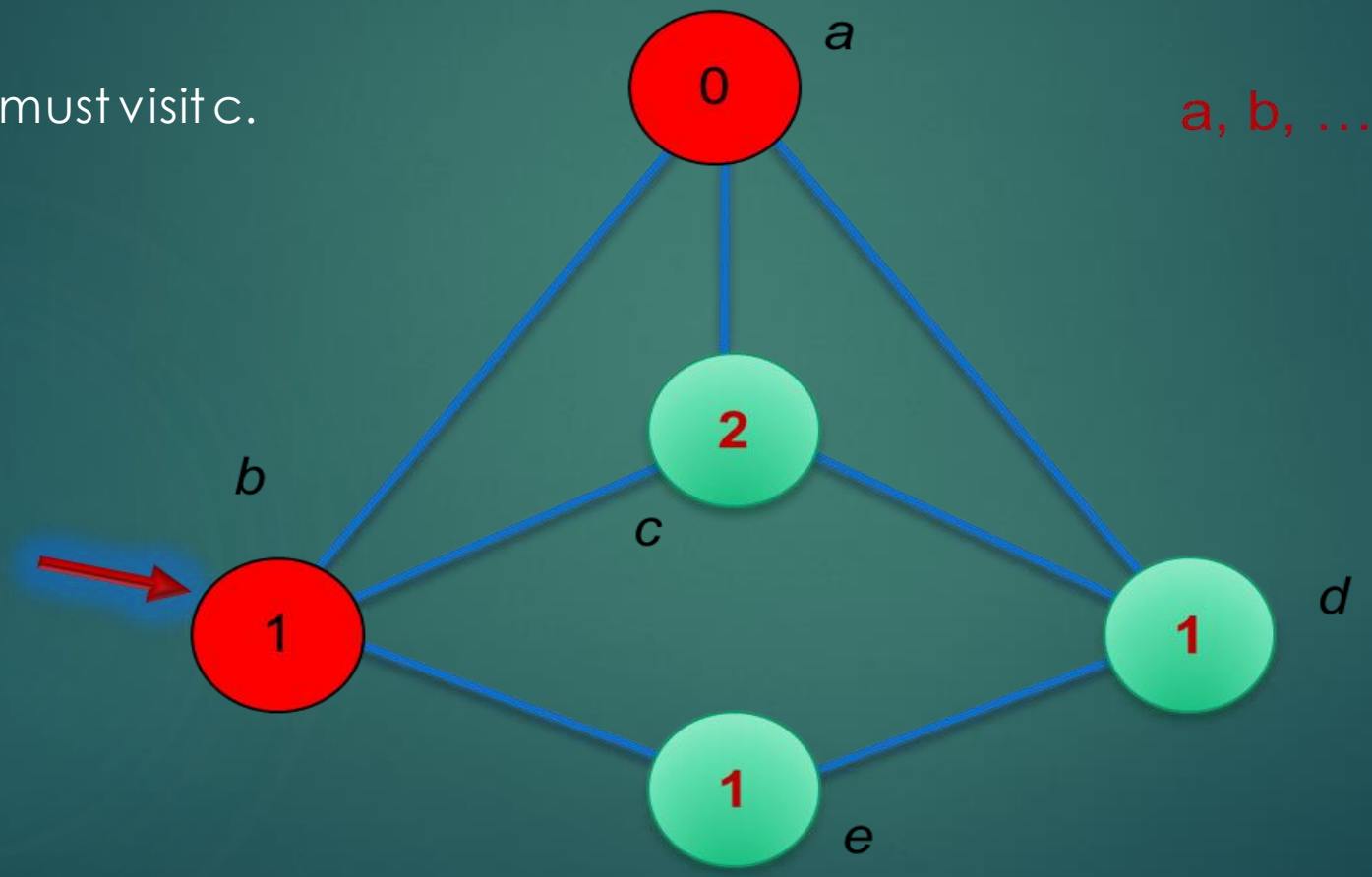
It cannot be e.



Computing treewidth

Maximum Cardinality Search: a 4-clique with one subdivision

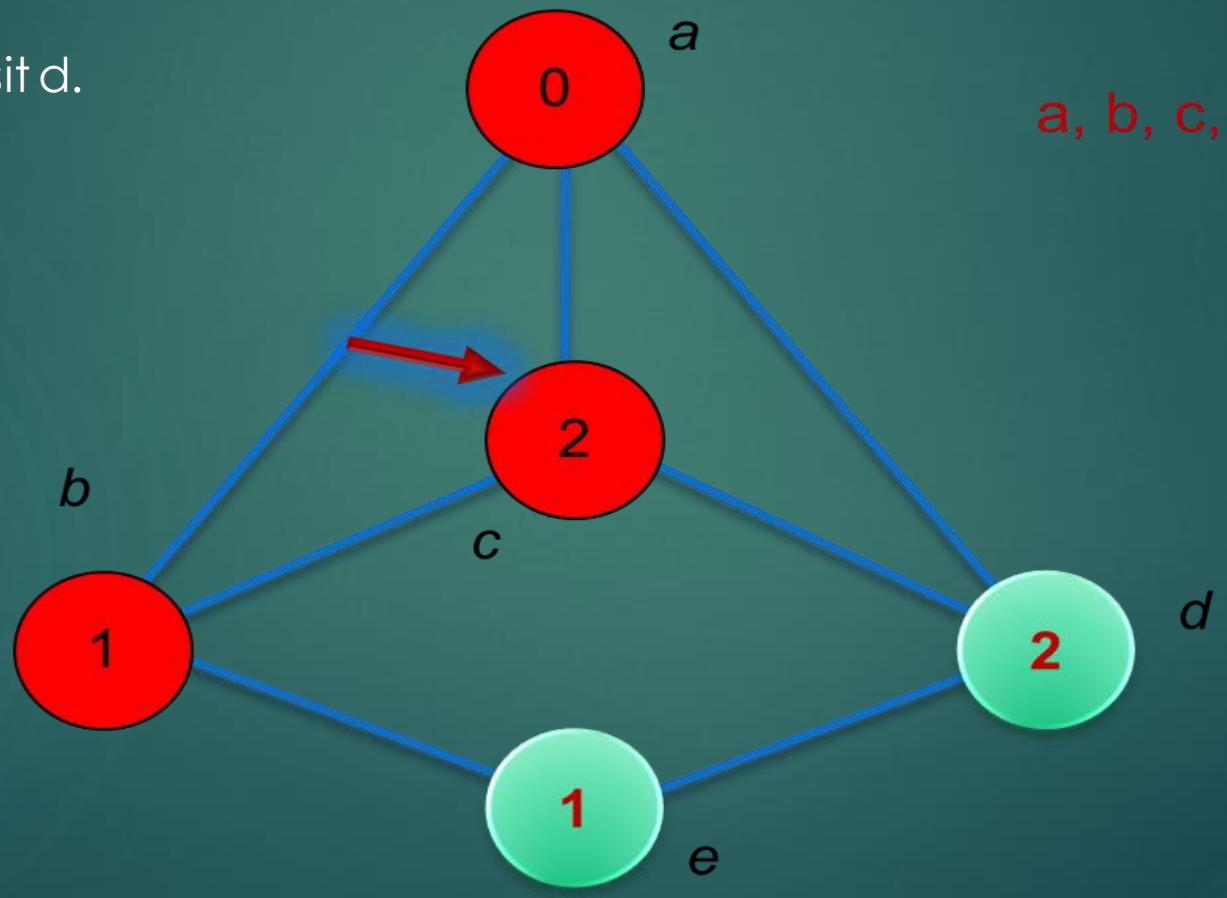
After b, we must visit c.



Computing treewidth

Maximum Cardinality Search: a 4-clique with one subdivision

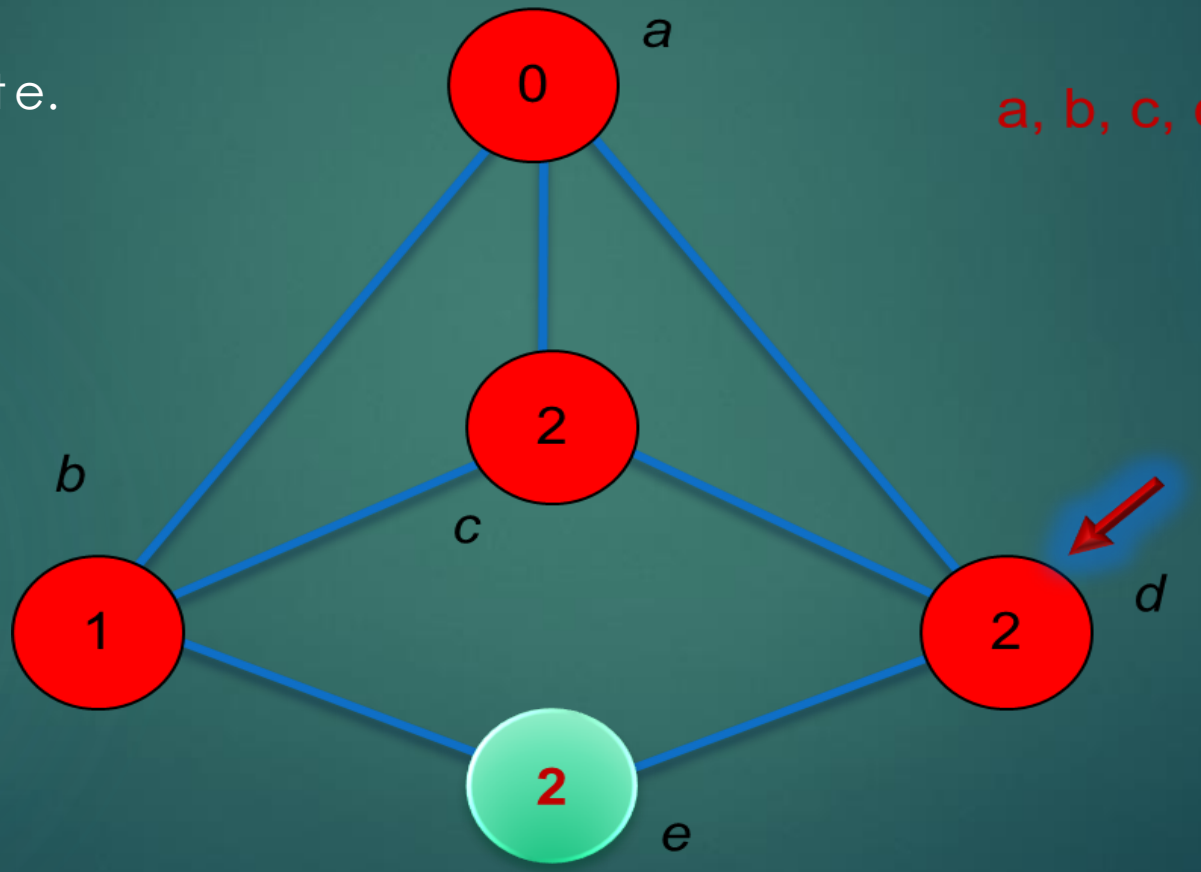
After c, we must visit d.



Computing treewidth

Maximum Cardinality Search: a 4-clique with one subdivision

After d, we must visit e.

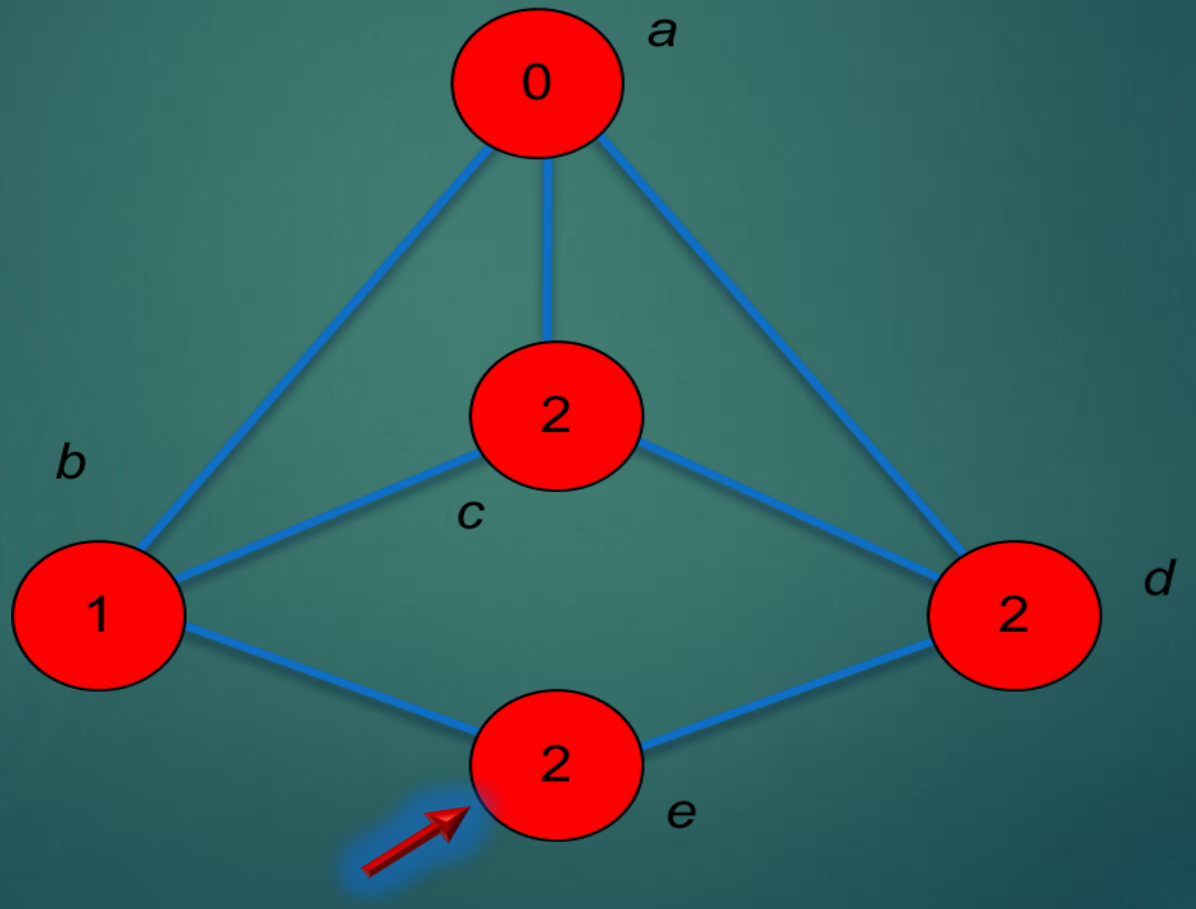


Computing treewidth

Maximum Cardinality Search: a 4-clique with one subdivision

We made an MCS-ordering of the graph:

a, b, c, d, e



Computing treewidth

Maximum Cardinality Search

- Used as an upper bound heuristic for treewidth
- If we have an MCS-ordering of G, and a vertex is visited with visited degree k, then the treewidth of G is at least k. (Lucena's theorem)
- Running (a few times) an MCS and reporting maximum visited degree gives lower bound for treewidth.

Thank You