I618 Advanced Computer Science II (Part II)

12/21 11:00-12:30
1/ 7 15:10-16:40
1/ 9 9:20-10:50
1/11 11:00-12:30
○1/16 9:20-10:50

Ryuhei Uehara
uehara@jaist.ac.jp
http://www.jaist.ac.jp/~uehara

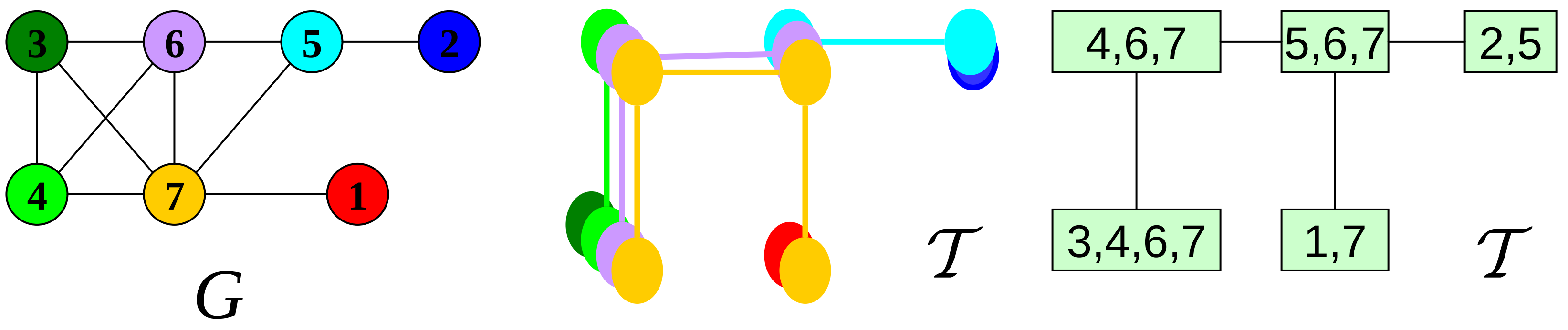
I give you some report problems TODAY with deadline 1/31

Algorithms on Interval/Chordal Graphs

- Some efficient algorithms on the graphs
  - Compact interval representation of an interval graph
    - which can be used for recognition and graph isomorphism.
    - each "cut" at integer point gives us a *maximal clique* which allows us to solve many problems efficiently.
  - "*Compact*" clique tree of a chordal graph
    - which *cannot* be used for graph isomorphism, but,
    - each "cut" at a node of the tree gives us a *maximal clique* which allows us to use dynamic programming and we can solve many problems efficiently.
  - the ideas can be extended to general graphs…
    - k-tree and partial k-tree
    - path decomposition and tree decomposition

Algorithms on Interval/Chordal Graphs

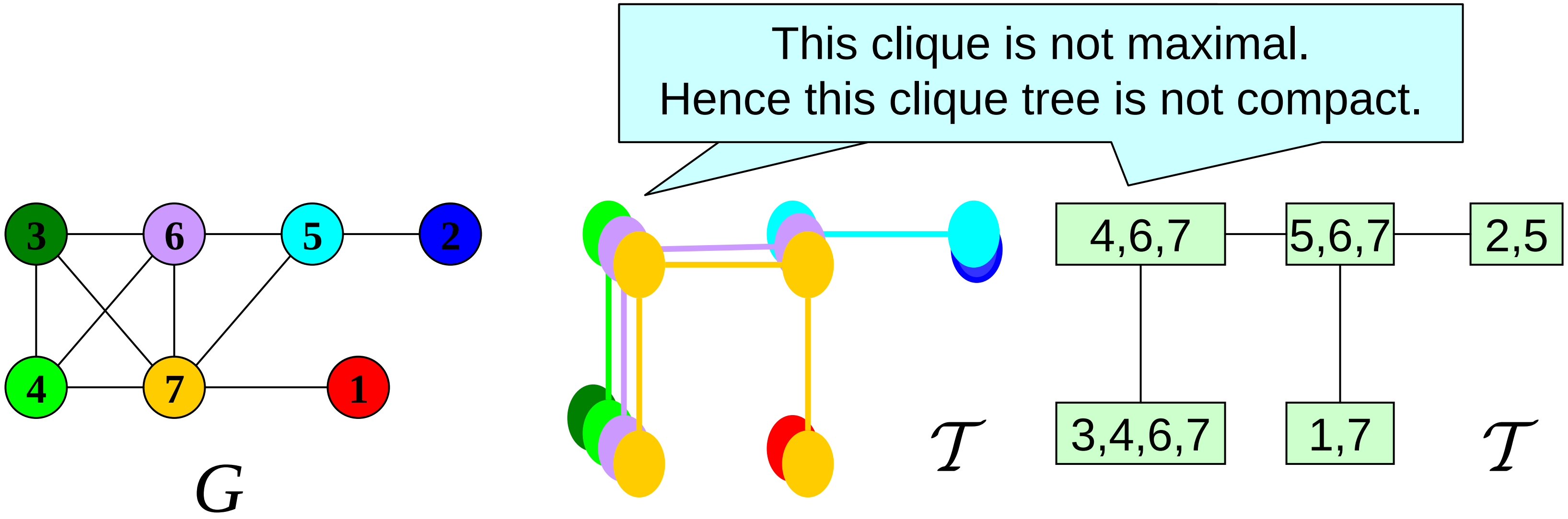
- “*Compact*” clique tree of a chordal graph
  - a clique tree $\mathcal{T}=(\mathcal{C},\mathcal{E})$ of a chordal graph $G=(V,E)$ is…
    - each vertex v corresponds to subtree T_v of $\mathcal{T}$
    - G is an intersection graph of T_vs of $\mathcal{T}$
    - for each node x in T ;
      - let $C(x)$ be the set of subtree T_vs that contains the node x.
      - clearly, the vertices in $C(x)$ induces a clique in G.
      - that is, each node x in $\mathcal{T}$ corresponds to a clique $C(x)$ in G.



Algorithms on Interval/Chordal Graphs

- "*Compact*" clique tree of a chordal graph
  - a clique tree $\mathcal{T}=(\mathcal{C},\mathcal{E})$ of a chordal graph $G=(V,E)$ is…
    - clearly, the vertices in $N(x)$ induces a clique in G.

[Definition 10] A clique tree is *compact* if and only if for each node x in $\mathcal{T}$, $C(x)$ induces a *maximal* clique.



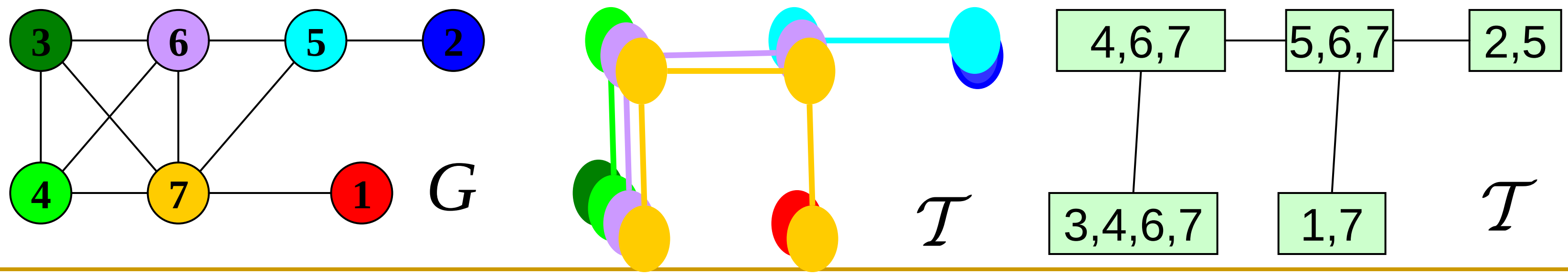
Algorithms on Interval/Chordal Graphs

- "*Compact*" clique tree of a chordal graph

[Lemma 3] If a node x in $\mathcal{T}$ does not corresponds to a maximal clique, $C(x) \subseteq C(y)$ for some y which is a neighbor of x.

(Proof (Sketch)) If $C(x)$ is not a maximal clique, there is a maximal clique M that contains $C(x)$. Then, $\mathcal{T}$ has to have a node corresponding to M. The path on $\mathcal{T}$ between $C(x)$ and M has to linearly ordered for inclusion since each vertex corresponds to a connected tree. $\square$

(C.f.) Similar idea can be found in [Def. 3](3).

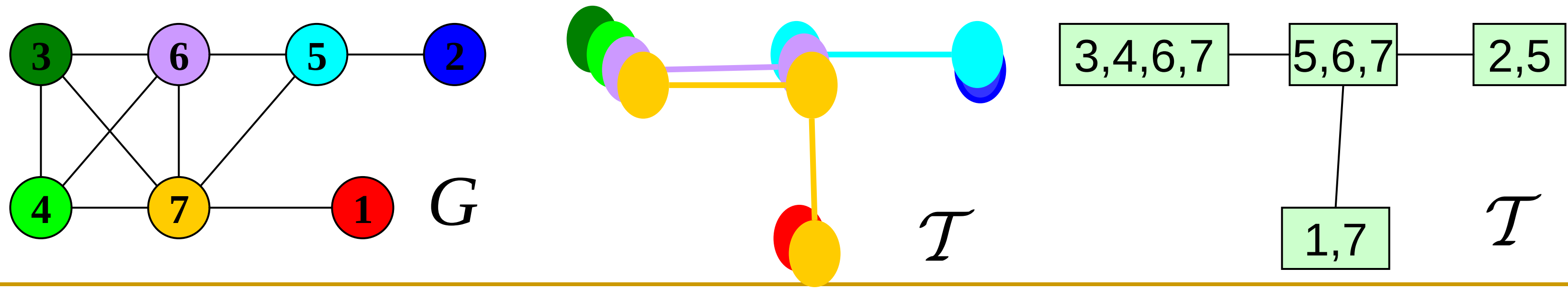


Algorithms on Interval/Chordal Graphs

- "*Compact*" clique tree of a chordal graph

[Lemma 4] For any clique tree $\mathcal{T}$ of a chordal graph G, we can construct a compact clique tree $\mathcal{T}$' in linear time of $|\mathcal{T}|$.

(Proof (Sketch)) If $C(x)$ is not a maximal clique, there is a neighbor y of x with $C(x) \subseteq C(y)$. We then contract x and y. Repeating this process, we finally have a compact clique tree. □



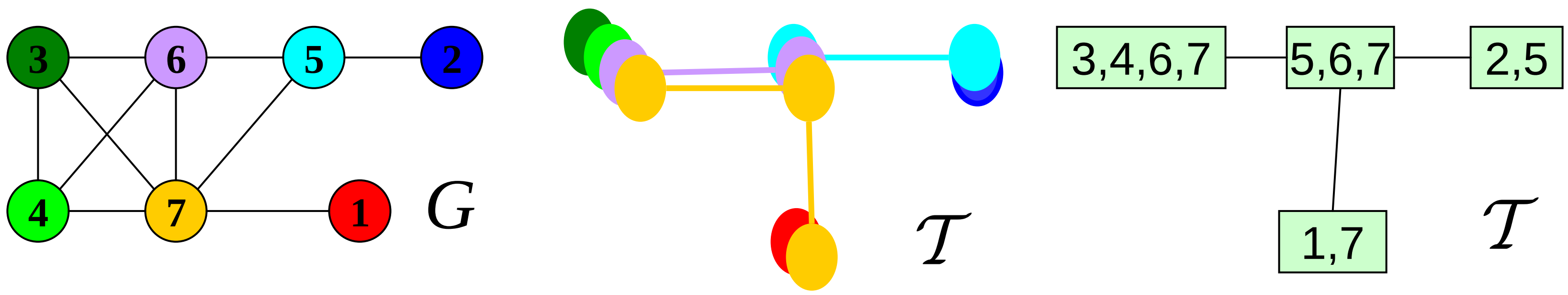
$\mathcal{T}$ $\mathcal{T}$

Algorithms on Interval/Chordal Graphs

- "*Compact*" clique tree of a chordal graph

[Theorem 17] A graph G is a chordal graph if and only if G has a compact clique tree representation.

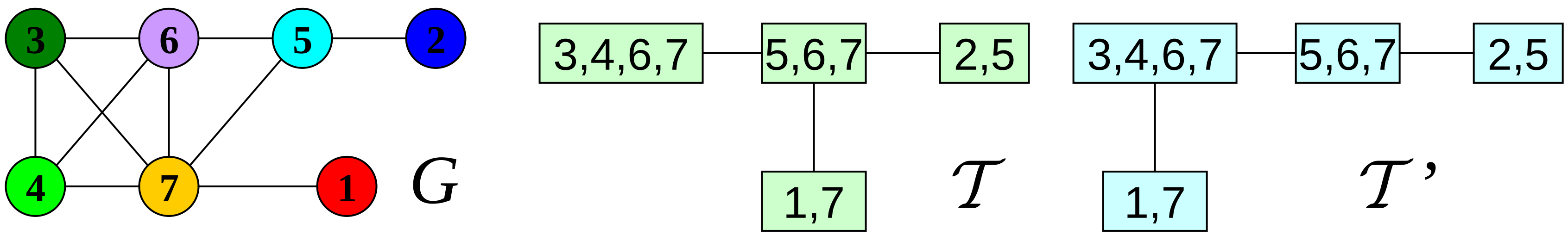
[Corollary 4] For a compact clique tree $\mathcal{T}$ of a chordal graph $G=(V,E)$, $\mathcal{T}$ has at most $|V|$ nodes. For each pair of nodes x and y in $\mathcal{T}$, we have $C(x) \not\subset C(y)$ and $C(y) \not\subset C(x)$.



[Useful property] Each leaf x in $\mathcal{T}$ has simplicial vertices in $C(x)$-$C(y)$, where y is the neighbor of x.

Algorithms on Interval/Chordal Graphs

- Two notes on compact clique tree
  1. Why the graph isomorphism is still hard for chordal graphs? … which is equivalent to "isn't the compact clique tree canonical up to isomorphism?"
     - The answer is, unfortunately, "No".
     - For a chordal graph, there are *__several distinct compact clique trees__* in general.



[Open Problem] As far as I know, there are no known $O(2^n)$ time algorithms that solve the GI problem on chordal graphs.

Algorithms on Interval/Chordal Graphs

- Two notes on compact clique tree
  2. As far as I know, most efficient and simple algorithm that constructs a compact clique tree is given in chapter 15 of "*Efficient Graph Representation*," J.P. Spinrad, 2003.
     - the outline of the algorithm is as follows;
       1. compute a PEO by LexBFS or MCS;
       2. for each $v_n, v_{n-1}, \ldots, v_1$, grow the tree as follows;
          1. if v_i gives a new maximal clique, grow the tree, or
          2. add v_i into the current maximal clique.
     - it is easy to implement the algorithm to run in linear time.

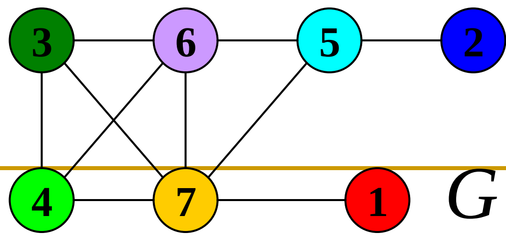
[Corollary 5] For a chordal graph, its compact clique tree can be constructed in linear time.

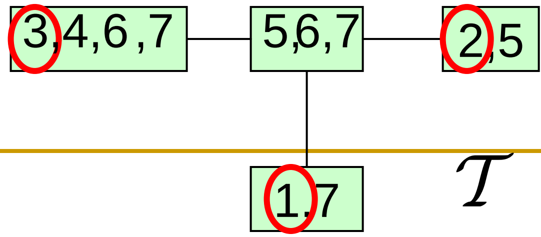
Algorithms on Interval/Chordal Graphs

- Efficient algorithms on interval/chordal graphs;
  - useful property of compact representations;
    - interval graphs;
      - by Lemma 1, on a compact representation, there is a (simplicial) vertex that corresponds to the interval [0,0].
    - chordal graphs; similar property of interval graphs;
      - on a compact representation,
        1. there is a simplicial vertex that corresponds to a leaf
        2. each leaf contains such a simplicial vertex
  - problems related to {clique/independent set} are easy to solve;
    1. maximum clique; find the maximum one in the compact representation.

Algorithms on Interval/Chordal Graphs

- Efficient algorithms on interval/chordal graphs;
  - problems related to {clique/independent set} are easy to solve;
    2. maximum independent set;
       1. $S := \phi$;
       2. on the clique tree,
          1. pick up a simplicial vertex v corresponding to a leaf in the tree;
          2. $S := S \cup \{v\}$;
       3. remove v and its neighbors from the graph;
       4. if the graph is not empty, go to step 2.



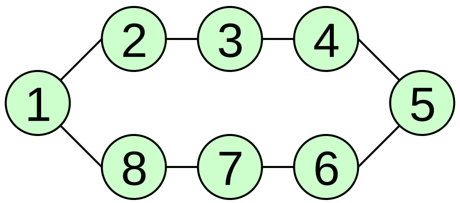


Algorithms on General Graphs

- Extensions…
  - We admit to lack of edges on each representation;

two vertices u, v are adjacent	⇔	corresponding trees/intervals share a common node
two vertices u, v are adjacent	⇒	corresponding trees/intervals share a common node

[Example 3] Long cycle, which is not a chordal graph.



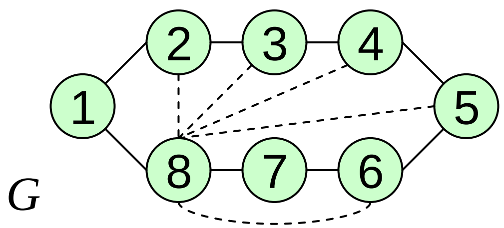
G

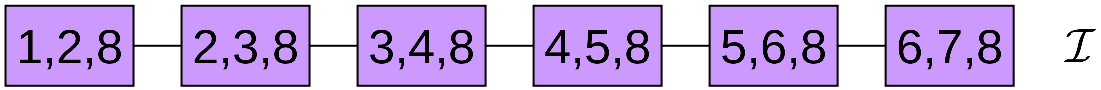
1,2,8 — 2,3,8 — 3,4,8 — 4,5,8 — 5,6,8 — 6,7,8 $\mathcal{I}$

1. Each edge in G appears in a node in $\mathcal{I}$,
2. each vertex in G appears consecutively in $\mathcal{I}$, and
3. *not* all pairs in a node in $\mathcal{I}$ produce edges in G.

Algorithms on General Graphs

- Extensions…
  - We admit to lack of edges; on each representation,
    - **From the viewpoint of efficient algorithms**, the representation is enough to solve many problems; each node in the representation is still a separator.
    - **From the graph theoretical point of view**, finding the representation of $G=(V,E)$ corresponds to finding a super graph $G'=(V,E')$ such that $E \subseteq E'$ and G' is chordal.





1. Each edge in G appears in a node in $\mathcal{I}$,
2. each vertex in G appears consecutively in $\mathcal{I}$, and
3. *not* all pairs in a node in $\mathcal{I}$ produce edges in G.

[Observation] Any graph has a trivial super chordal graph;

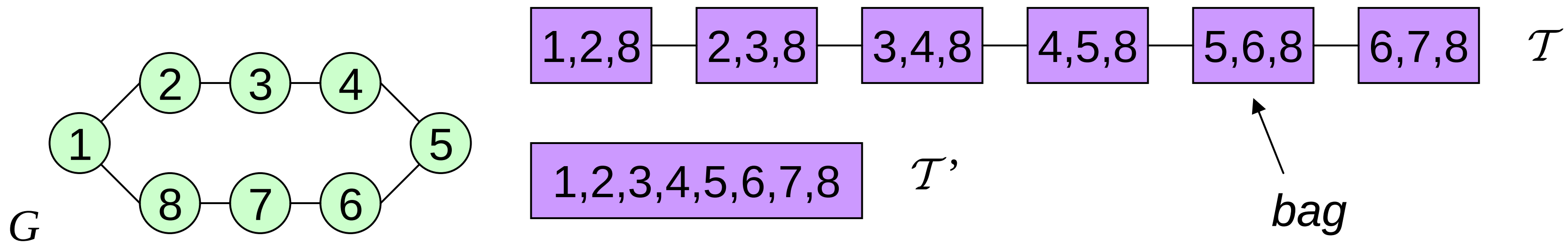
1,2,3,4,5,6,7,8 any graph is a subgraph of K_n.

Algorithms on General Graphs

- Extensions…

[Definition11] A tree decomposition of a graph G=(V,E) is a tree $\mathcal{T}$=($\mathcal{B}$,$\mathcal{E}$) such that

1. each *bag* B in $\mathcal{B}$ is $B \subseteq V$,
2. for each edge e={u,v} in E, there is a bag B with u,v in B,
3. each vertex v in V induce a (connected) subtree T_v of $\mathcal{T}$; precisely, T_v consists of nodes B in $\mathcal{B}$ with v in B.

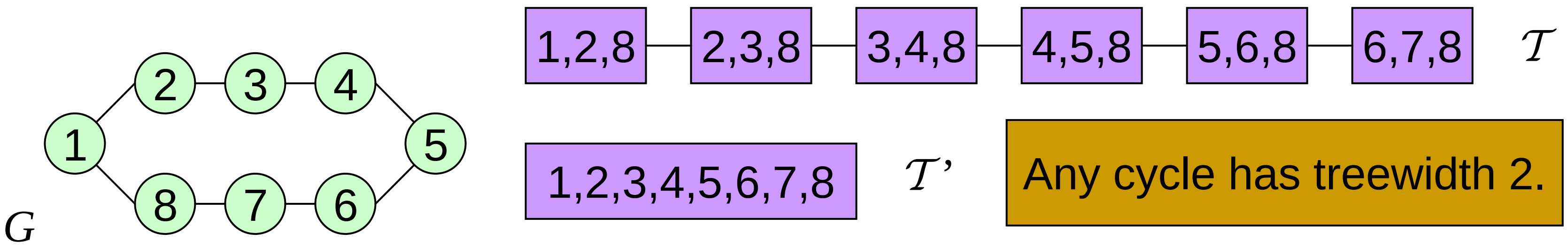


Algorithms on General Graphs

- Extensions…
  - We want to have a *good* tree decomposition
    - "good" = "$\max\{|B|\}$ is small"

[Definition12]

1. The *treewidth* $tw(\mathcal{T})$ of a tree decomposition $\mathcal{T}=(\mathcal{B},\mathcal{E})$ of a graph $G=(V,E)$ is defined by $tw(\mathcal{T})=\max\{|B|\}$ for all B in $\mathcal{B}$.
2. The *treewidth* $tw(G)$ of a graph G is defined by $\underline{\min\{tw(\mathcal{T})\}-1}$ for all tree decompositions $\mathcal{T}$.



Algorithms on General Graphs

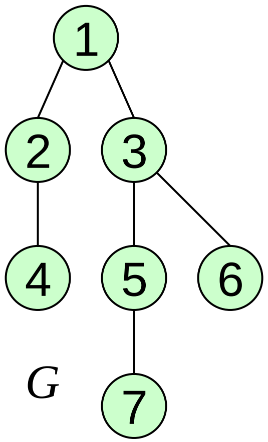
- Extensions…

[Definition12]

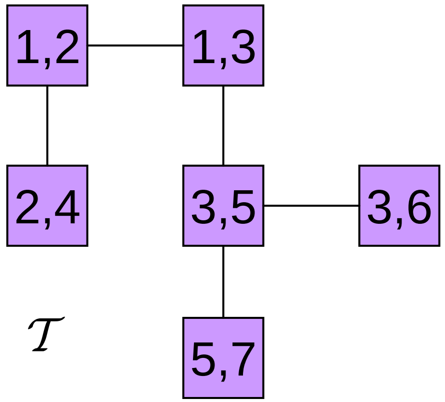
1. The *treewidth* $tw(\mathcal{T})$ of a tree decomposition $\mathcal{T}=(\mathcal{B},\mathcal{E})$ of a graph $G=(V,E)$ is defined by $tw(\mathcal{T})=\max\{|B|\}$ for all B in $\mathcal{B}$.
2. The *treewidth* $tw(G)$ of a graph G is defined by $\underline{\min\{tw(\mathcal{T})\}-1}$ for all tree decompositions $\mathcal{T}$.

Any cycle has treewidth 2.

Any tree has treewidth 1.



G



$\mathcal{T}$

Algorithms on General Graphs

- ## Known Results
  - Computing the treewidth of a graph is $\mathcal{NP}$-complete in general.
    - there are several approximation algorithms for some graph classes.
  - But, the treewidth is *fixed parameter tractable*; that is,
    - if the graph $G=(V,E)$ has treewidth k, we can compute the tree decomposition of treewidth k in $O(f(k)\ \mathrm{poly}(n))$ time, where $n=|V|$, even if we do not know the value of k; here, $f(k)$ is not polynomial of k, but $f(k)$ is independent from n.
    - actually, for fixed k, there is a linear time algorithm that computes the treewidth. [Bodlaender 1996]
  - Thus, in a practical sense, treewidth can be computed
    - … for graphs having small treewidth.

Algorithms on General Graphs

- Graphs with small treewidth k;
  - "A graph has treewidth k" means the graph can be decomposed into small graphs (with vertices of constant number) by using the separators of size at most k.
  - Typical "*dynamic programming*" on the tree decomposition works; for each merging process of nodes, the algorithm may maintain some tables of size $O(2^k)$, which is polynomial of n (or constant).
    - in some areas like bioinformatics, treewidth of graphs is bounded by 3.

Algorithms on General Graphs

- Graphs with small treewidth k;
  - Bodlaender wrote two survey papers;
    - "A partial k-arboretum of graphs with bounded treewidth," *Theoretical Computer Science*, 209, p.1-45, 1998.
    - "A tourist guide through treewidth," *Acta Cybernetica*, 11, p.1-21, 1993.

    … and tons of papers can be found at Bodlaender's web page (http://people.cs.uu.nl/hansb/mypapers.html)

  - in the papers, you can find many tractable problems on graphs with small treewidth.

Report

Deadline: January 31 (Thusday), 17:00.

Submit to Uehara at I67b.

1. Prove the Helly property for the set of intervals.
2. Prove that the graph isomorphism can be solved in linear time for trees.
3. Prove that any vertex can be the last vertex of a perfect elimination ordering of a chordal graph.