CSC791/495

Dr. Blair D. Sullivan

September 22, 2017

Reminders

① Week-5 homework now due Tuesday 9/26 (at ~~9am~~ midnight ☺)

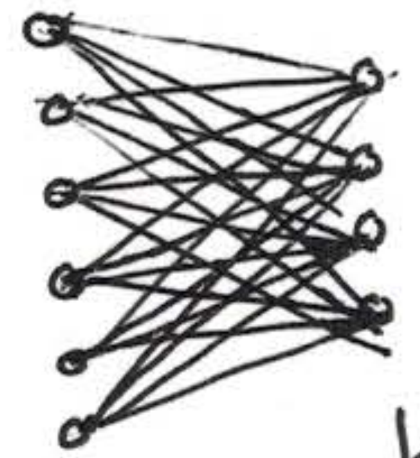
② Proof Review exercise due 9/29 – don't wait too long to start. (esp. 791 students! Correcting proofs is hard).

③ Opportunity Identification Project posted

- report due 10/6
- posters 10/13
- new slack channel #opportunityID

④ Exemplars posted in #problem-solution for Weeks 2,3,4

Treewidth, revisited



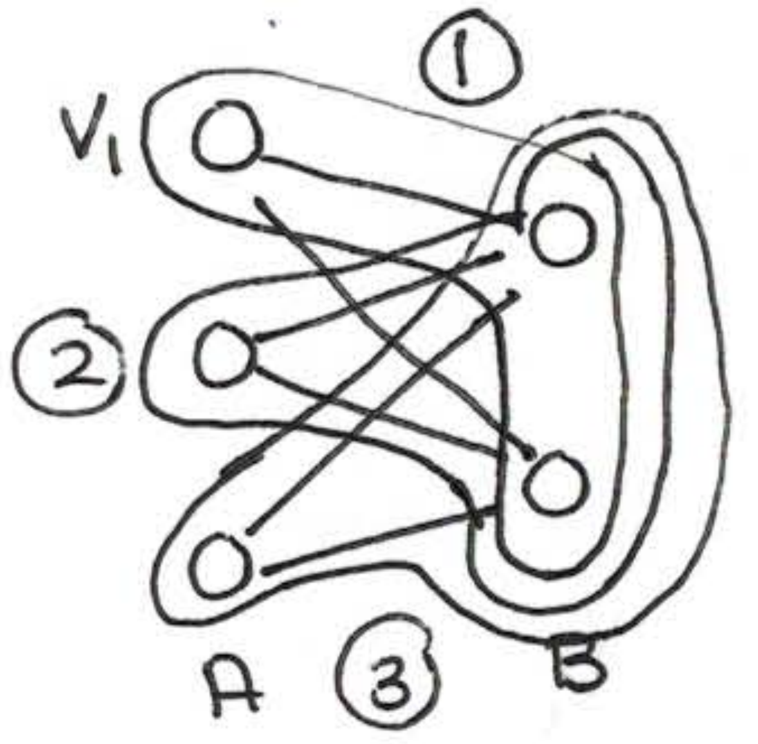
$K_{6,4}$

$H = K_{m,n}$, a complete bipartite graph on $V = A \cup B$ with $|A| = m$, $|B| = n$.

$tw(K_{m,n}) \le \min(m,n)$

- Give a tree decomposition of H

width $= \min(m,n)$



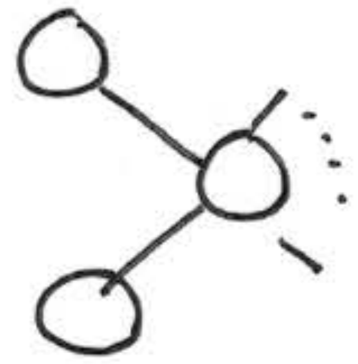
1. pick smaller side, say B
2. make a bag w/ B + one vertex of A (v_1)
3. drop v_1, pick up v_2
4. repeat until A is covered

$B \cup \{v_1\}$

$B \cup \{v_2\}$

$B \cup \{v_3\}$

or



- Prove your decomposition has minimum width

$tw(K_{m,n}) \ge \min(m,n)$. Cops & Robbers: $\min(m,n)$ $\Rightarrow$ if robber wins, $tw > \min(m,n) - 1$

what's the strategy for the robber to win?

always able to run to a node on small side w/o a cop OR

remain on big side (if all cops go to small side)

- Why must every decomposition have either a bag containing A or one containing B?

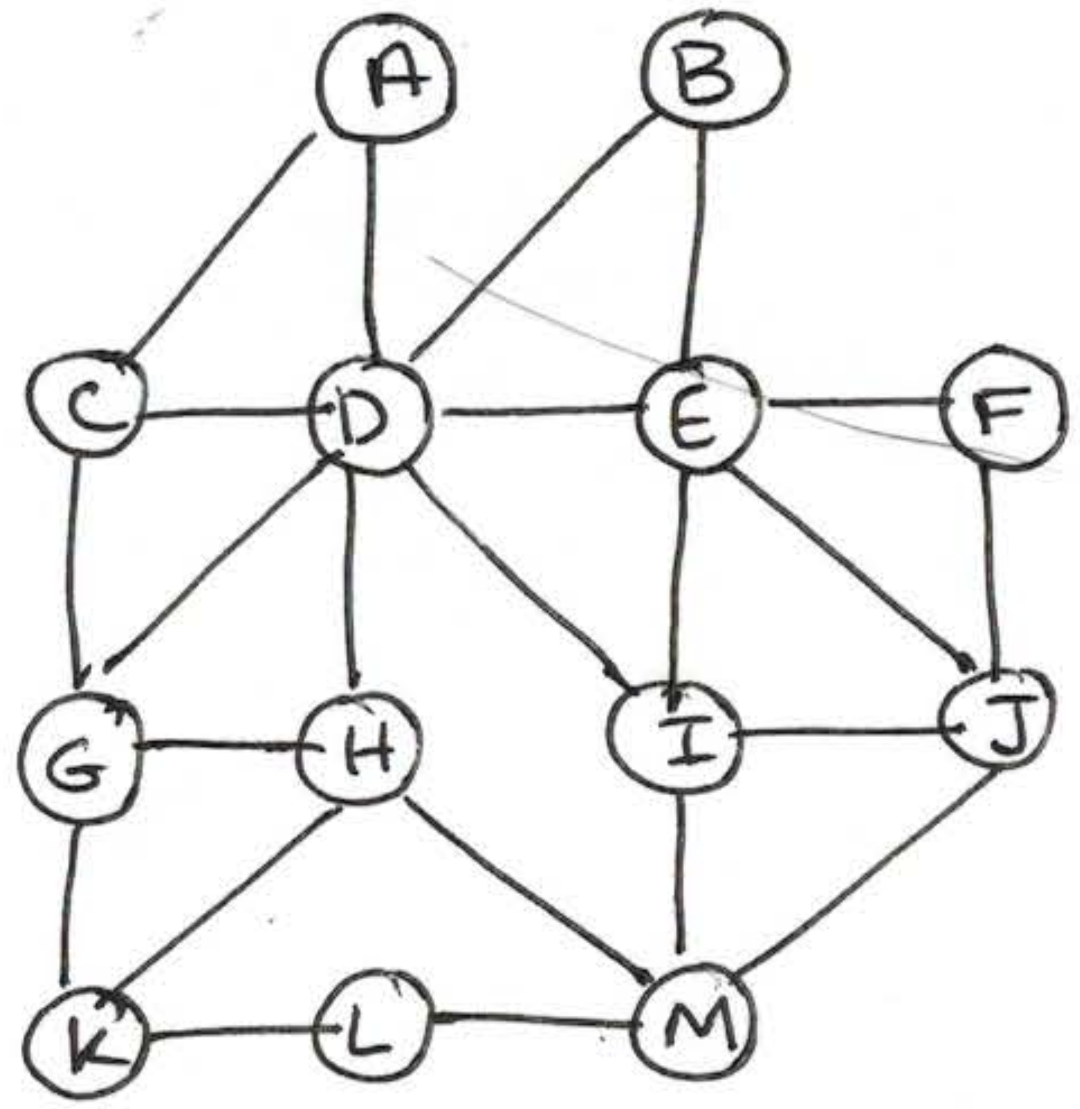
Suppose no bag contains A (WLOG). Consider a bag containing a vertex of A, v.

v is adjacent to every node in B. Then some bag has ~~node~~ vertex b_1. $\Rightarrow$?

Try to formalize this proof by contradiction!

Thinking of Trees

edited



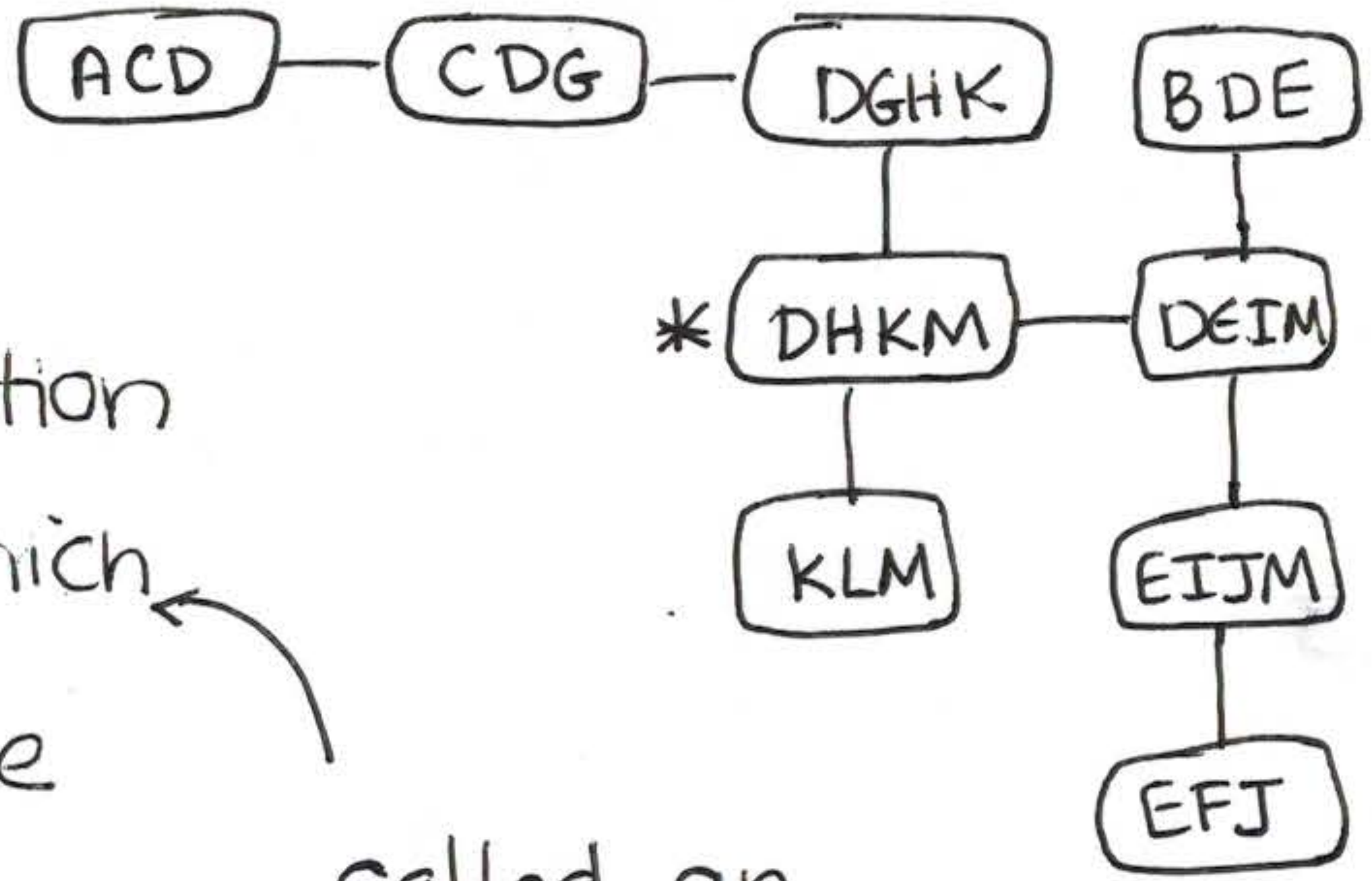
Defn a k-tree is a graph G where either
① $G = K_{k+1}$ or ② $\exists v \in G$, $\deg(v) = k$, $G \setminus v$ a k-tree.

0-trees: 

1-trees
(trees!)

Defn partial k-trees are subgraphs of k-trees.

Thm G has $tw \leq k \iff G$ is a partial k-tree.



- This observation can be used to form a tree decomposition by unraveling the recursion (vertices of $\deg \leq k$ which are removed). It is important that k-trees are chordal (triangulated) for this algorithm & so one must "fill-in" (triangulate) as you go to calculate correct bags.

called an elimination ordering.

make all higher-indexed neighbors adjacent

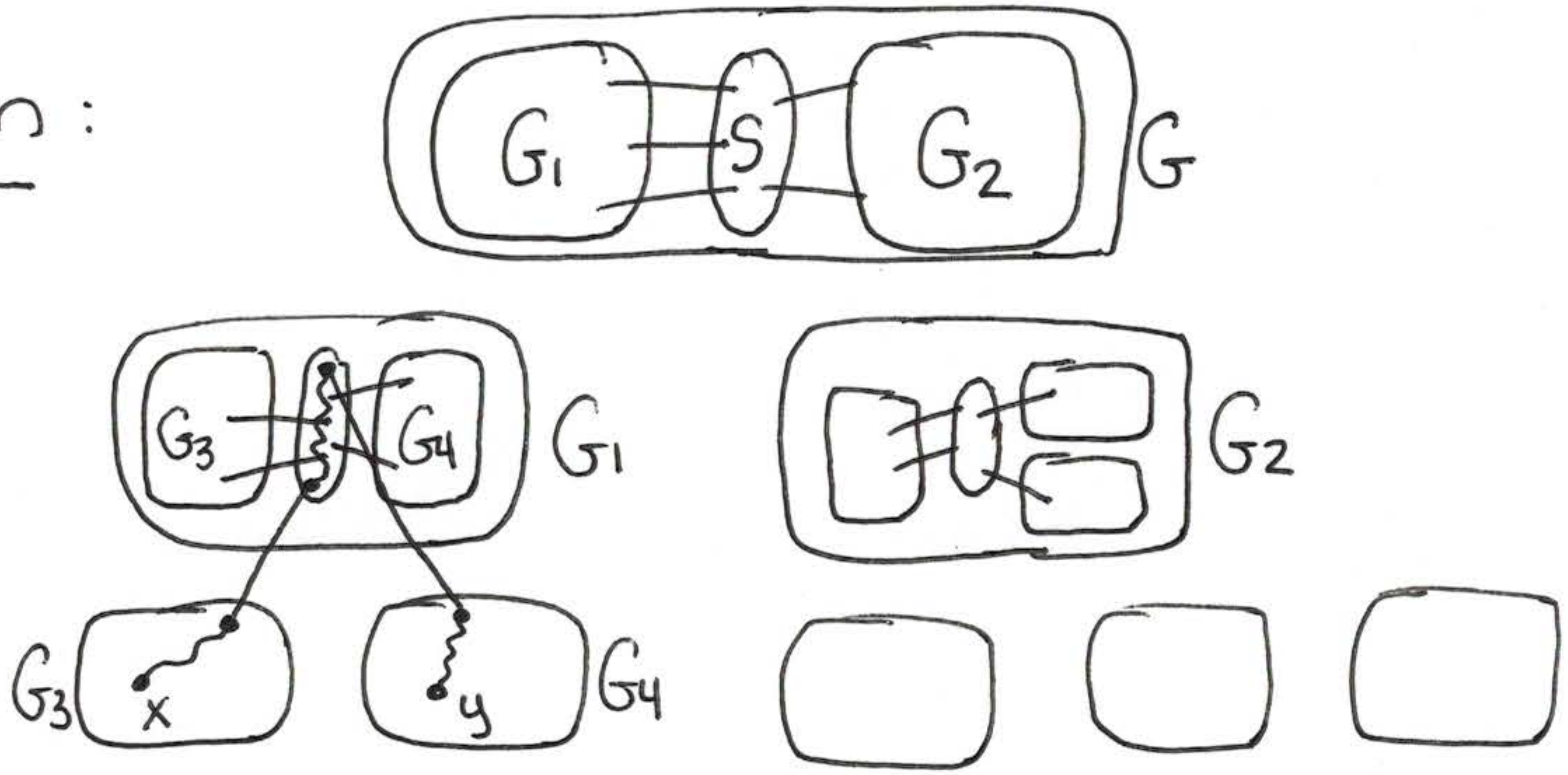
[I made a mess of this in class – my apologies!]

A Separate Topic?

<u>Defn</u> $S \subseteq V$ is a <u>vertex separator</u> if $G \setminus S$ has at least two connected components. S is a <u>balanced</u> separator if every component has $\geqslant \frac{2}{3}|V(G)|$ vertices.

<u>Trees</u>: have balanced separators of size 1! Furthermore, we can do this repeatedly until no separators left (edges/isolates).

$\Rightarrow$ <u>Recursive Decomposition</u>:



How does treewidth come into the picture?

① Graphs of tw k have balanced separators of size $\leq k$

② tree decompositions <u>are</u> recursive separator decompositions.

→ pick a root & let $D_i = \bigcup_{j \text{ desc of } i} X_j$, $A_i = \bigcup_{j \text{ proper anc. of } i} X_j$, $B_i = A_i \cap D_i$. Then X_i separates $G[D_i] \setminus \{(u,v) \mid u,v \in B_i\}$.

In-Class Exercise

Let G be a graph and $(T=(I,F), \{X_i\})$ a tree decomposition of width $\leq k$. Prove that if there are at least $k+1$ vertex-disjoint paths between vertices x and y, some bag contains both x and y.

My Approach

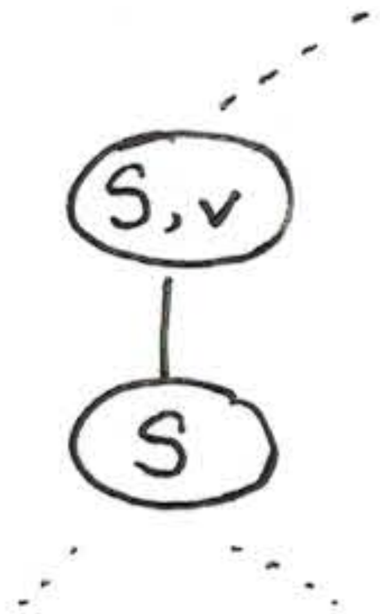
- First, prove that $X_i \cap X_j$ is a vertex separator for any edge ij of T (a TD).
- Use this to argue that if x & y don't co-occur in a bag, they live on opposite sides of a $(\leq k)$-separator
- Observe that $k+1$ disjoint paths can't be routed through such a separator.

Could it get any nicer?

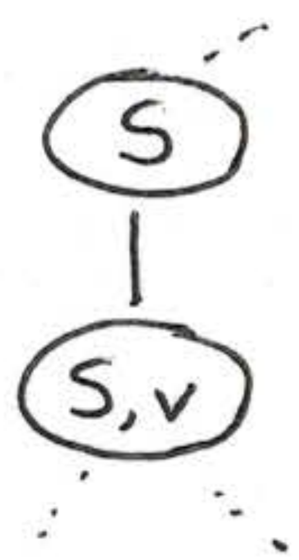
Definition A (rooted) tree decomposition is nice if every node X_i is of one of the following types:

1. root / leaf $|X_i| = 0$ 

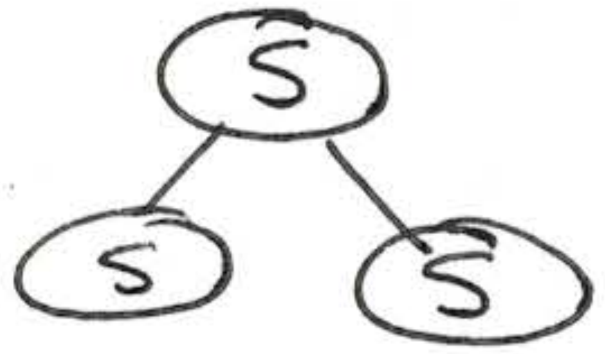
2. introduce: one child X_j: $X_i = X_j \cup \{v\}$ $v \notin X_j$



3. forget: one child X_j: $X_i = X_j \setminus \{v\}$ $v \in X_j$



4. join: two children X_j, X_k: $X_i = X_j = X_k$



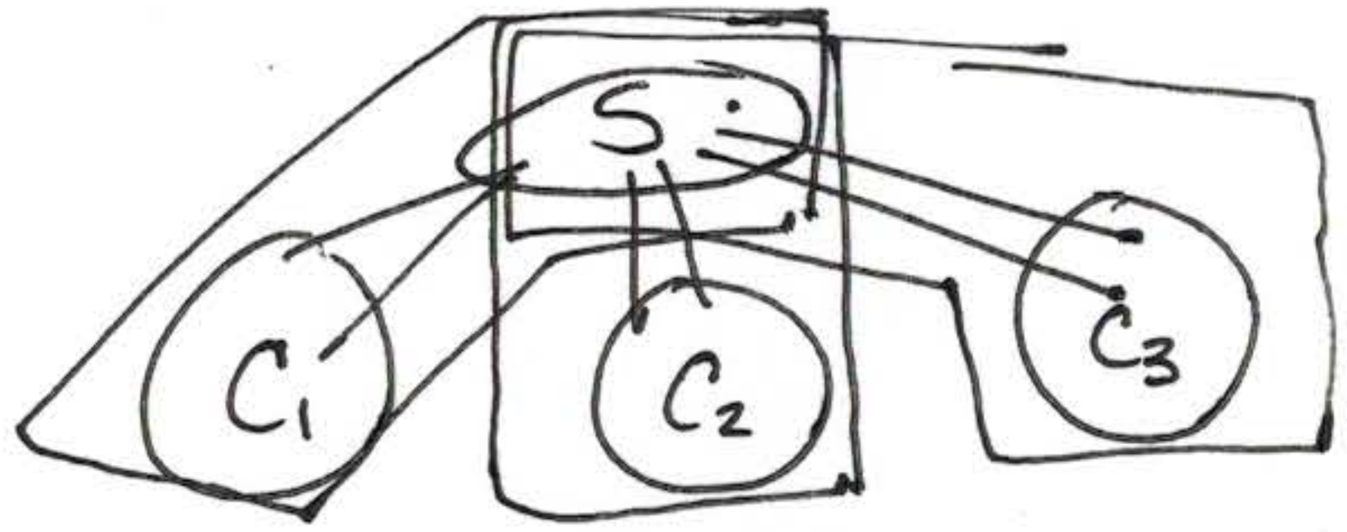
Thm G has tw $\le k \Rightarrow G$ has a nice TD of width $\le k$ and $O(kn)$ nodes.

MWIS parameterized by treewidth

Problem Given a graph G of treewidth $\leq k$ and a nice tree decomposition $(T, \{X_i\})$ of G (of width k), find the max. weighted independent set in G under weight function $\omega: V(G) \to \{\text{non-negative reals}\}$.

need to remember $M[\overset{i}{\cancel{x}}, S]$ = max weight of an indep. set in T_x with $\underline{I \cap X_i = S}$.

T_x ← tree below x (D_i)



how can an independent set behave?

If $I_j \subseteq C_j$ independent $\Rightarrow$ $I_1 \cup I_2 \cup I_3$ is indep.

- If I_1 is indep. in $C_1 \cup S$ $\Rightarrow$ what sets in C_2 can I safely merge it with?
- If nothing is adj to a member of S that's in I_1 $\Rightarrow$ $(I_1 \cap S) \cup I_2$ is indep.

that's all we need

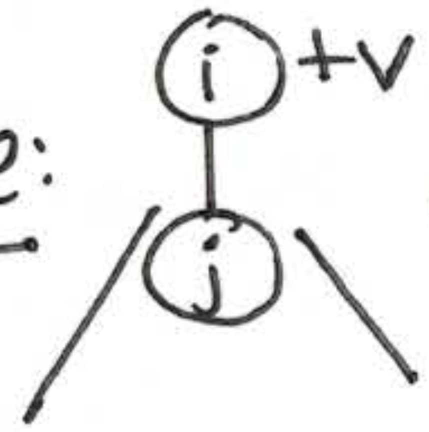
how big is this table?

$\leq 2^{|X_i|}$ sets S

$\leq 2^{tw+1}$

<u>leaf:</u> trivial $M[i, \emptyset] = 0$

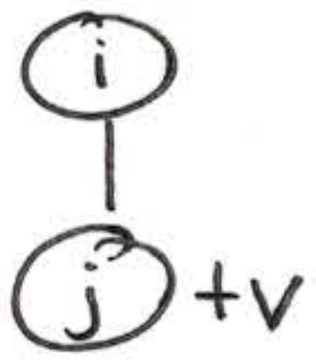
<u>introduce:</u>



$$M[i,S] = \begin{cases} m[j,S] & v \notin S \\ m[j, S \setminus v] + w(v) & v \in S, \text{indep.} \end{cases}$$

could v have a nbr in T_i?
No b/c X_j is a separator.

<u>forget:</u> $M[i,S] = \cancel{\max[j,S] + ma}$



$$\max(m[j,S],\ m[j, S \cup \{v\}])$$

<u>join:</u> ($\wedge$) $m[j_1, S] + m[j, S] - \omega(S)$

time $O(2^k \cdot n)$

3-coloring: an exercise

Thm 3-coloring has a $3^{tw}\, tw^{O(1)}\, n$ algorithm.

Sketch We'll solve this by DP over a nice TD.

- What should we store in the table?

- How can we update at each type of node?

leaf:

introduce:

forget:

join:

- How long does it take?

Problems

① Show OCT is FPT parameterized by treewidth.

② Show SAT is FPT parameterized by the treewidth of (a) its primal graph or (b) its incidence graph.

Defn φ a CNF formula. The primal graph $G_P(\varphi) = (V_P, E_P)$ with $V_P = \{\text{variables}\}$ and $E_P = \{(x,y) \mid x, y \text{ co-occur in a clause of } \varphi\}$. The incidence graph $G_i(\varphi) = (V_i, E_i)$ is the bipartite graph with $V_i = \{\text{variables}\} \cup \{\text{clauses}\}$ and $E_i = \{(x, C) \mid x \text{ is a variable occurring in clause } C\}$.

Courcelle's Theorem (bonus material)

<u>EMSO</u>: extended monadic second order logic (on graphs)

- logical connectives $\wedge, \vee, \rightarrow, \neg, =, \neq$
- quantifiers $\forall, \exists$ over vertex/edge variables
- predicate $adj(u,v)$: vertices u,v are adjacent
- predicate $inc(e,v)$: edge e is incident to vertex v
- $\in, \subseteq$ for vertex/edge sets

Example: $\exists C \subseteq V \; \forall v \in C \; \exists u_1, u_2 \in C \, (u_1 \neq u_2 \wedge adj(u_1, v) \wedge adj(u_2, v))$

<u>Thm</u> If a graph property can be expressed in EMSO with formula φ, there is an FPT algorithm for the property parameterized by treewidth.

<u>WARNING</u>: the constants involved are (very) unfriendly.