Algorithms and Complexity

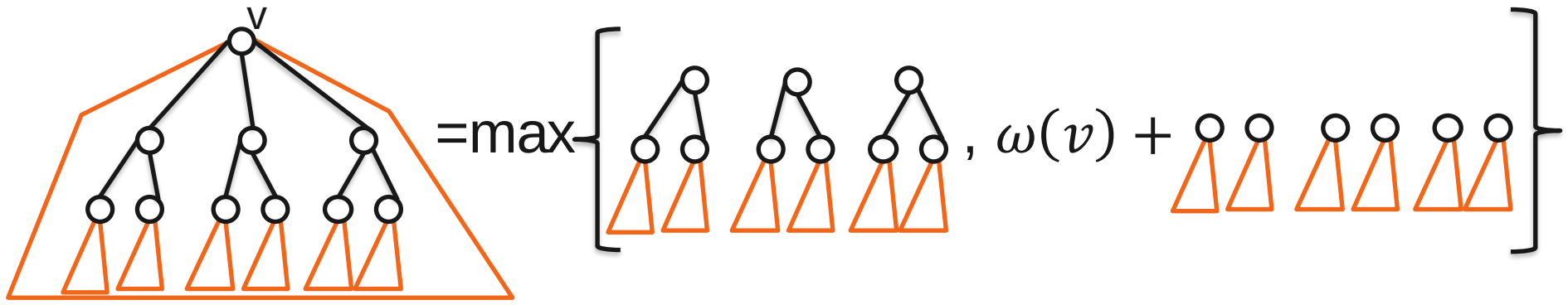
Lecture 8

Introduction to Treewidth

Reminder: Weighted Independent Set in Trees

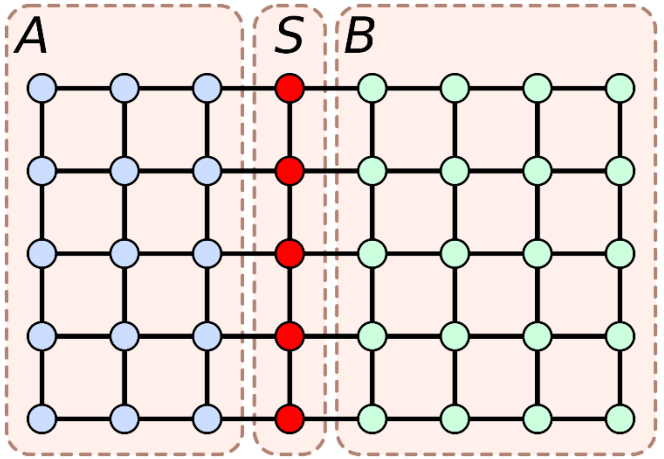
- Let T be a rooted tree, $T[v]$ the subtree rooted at v ∈ V.
- Define $A[v]$ as the max weight of an ind. set of $T[v]$, then

$$A[v] = \begin{cases} \omega(v), & \text{if } T[v] \text{ is a single node,} \\ \max\left\{ \sum_{c \in ch(v)} A[c], \omega(v) + \sum_{c_1 \in ch(v)} \sum_{c_2 \in ch(c_1)} A[c_2] \right\}, & \text{otherwise.} \end{cases}$$



Separators

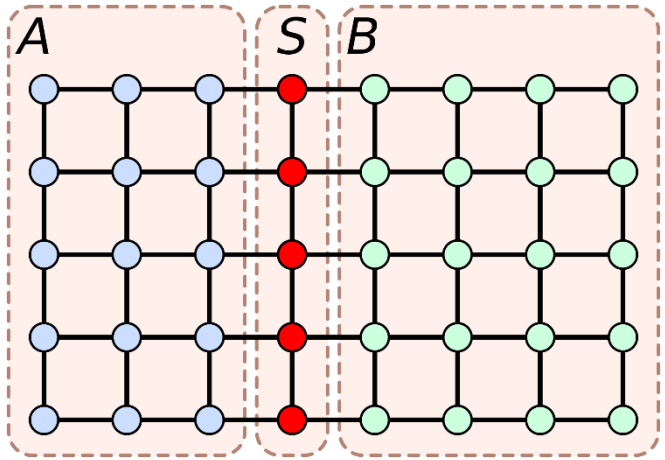
- A vertex set S separates A, B if $V = S \cup A \cup B$ and there are no edges between A and B



- The algorithm for independent sets in trees uses separators
- Since A, B do not interact (except through S), we can solve the problems independently and combine solutions

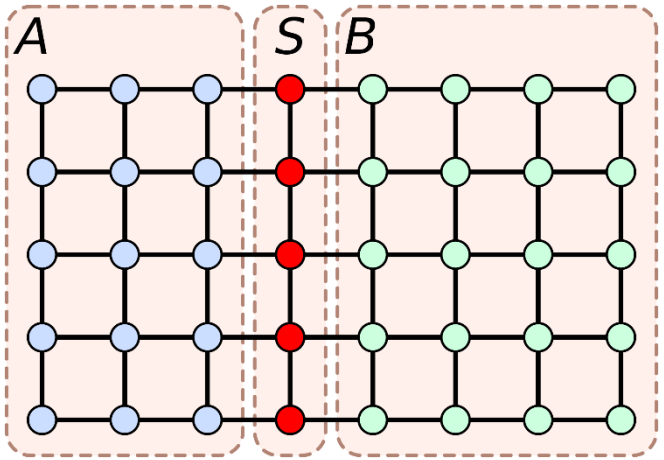
Planar separator theorem

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with
  - $|S| \leq 3\sqrt{n}$
  - $|A|, |B| \leq 2/3\ n$



Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$

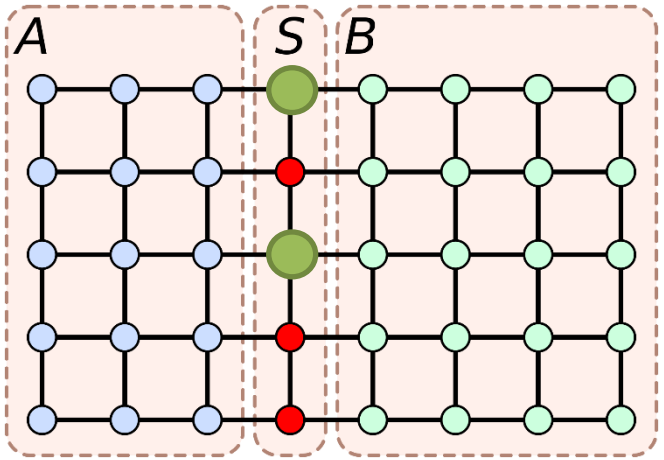


Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$

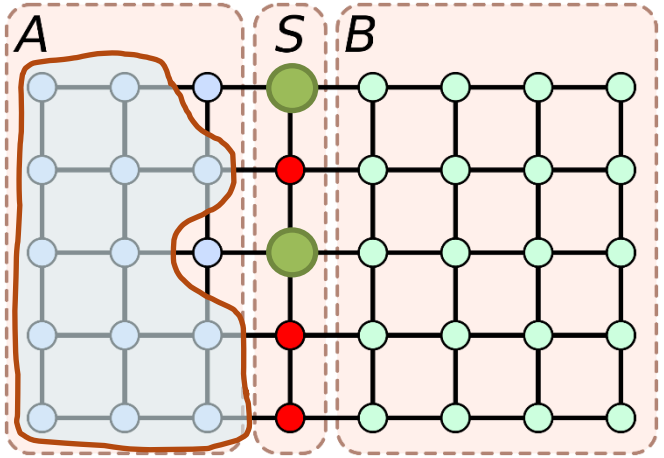


Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$

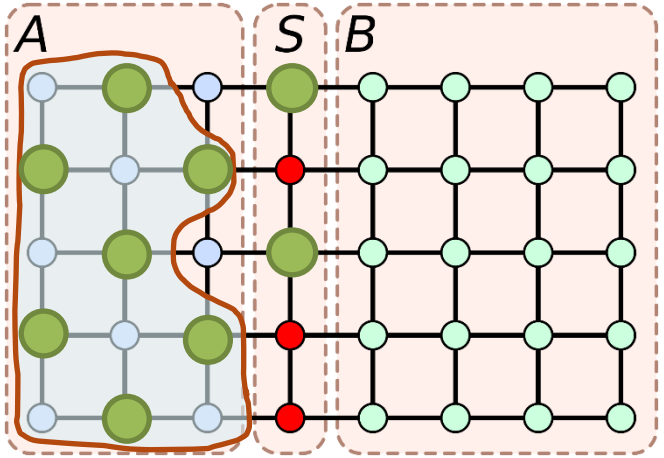


Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$

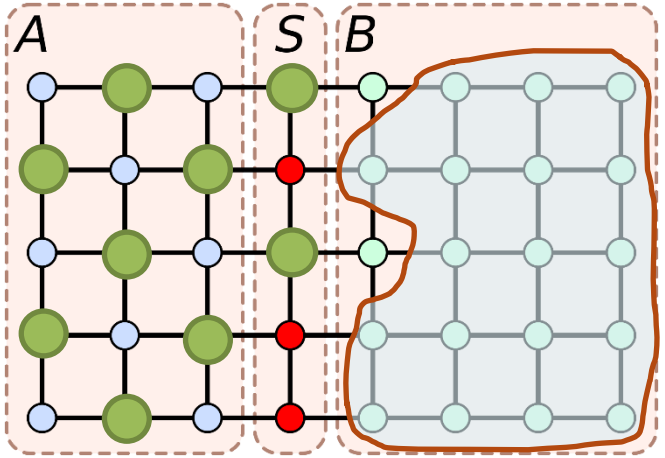


Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$

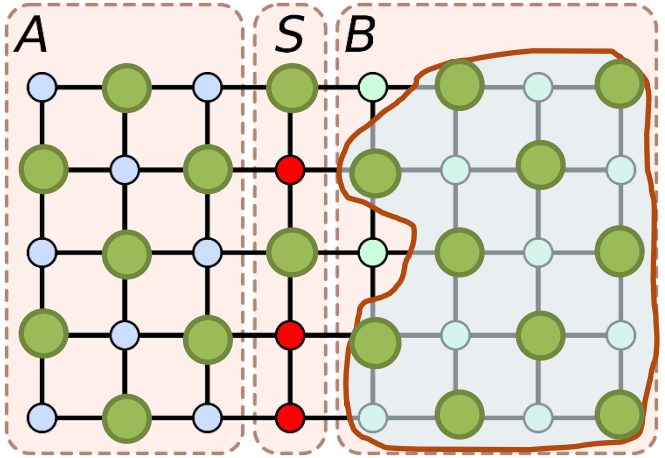


Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$

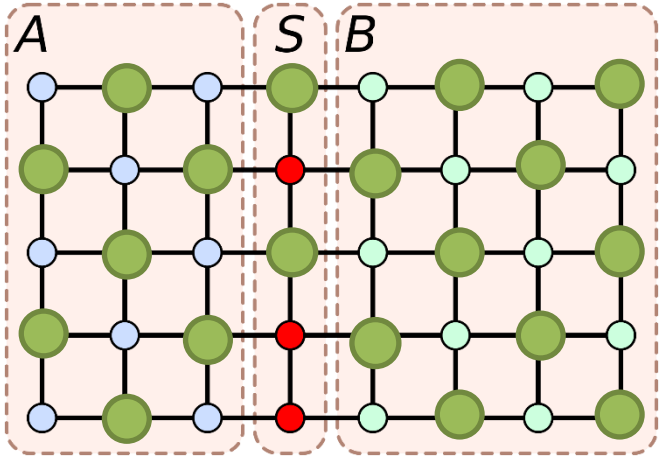


Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$



Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$
- $T(n) \leq 2^{3\sqrt{n}}\big(T(2n/3) + T(2n/3)\big)$

Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

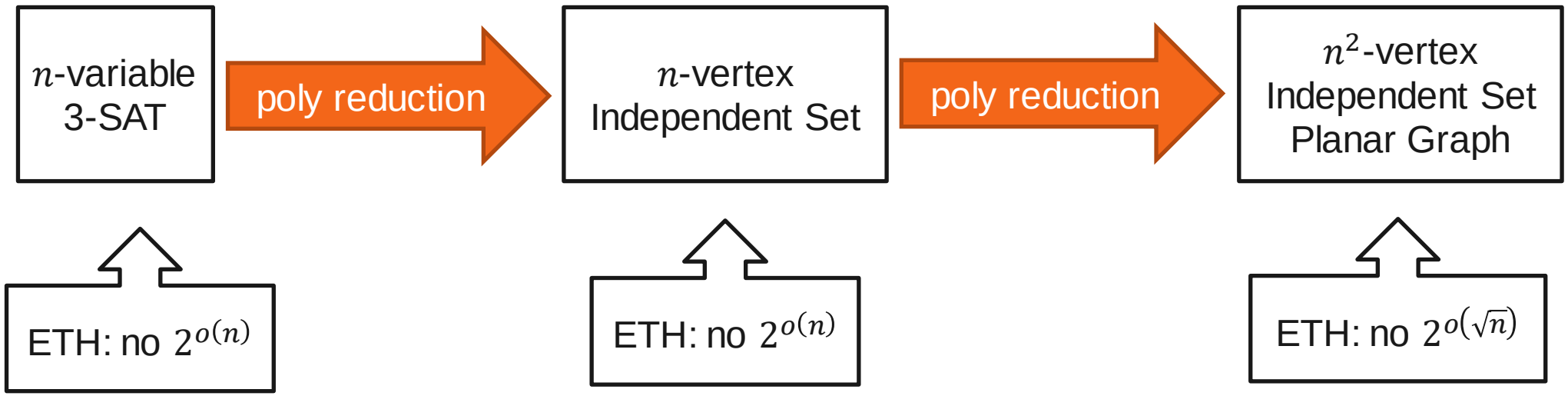
- Let G be an n-vertex planar graph, then there exists a separator (S, A, B) with $|S| \leq 3\sqrt{n}$ and $|A|, |B| \leq 2/3\, n$
- $T(n) \leq 2^{3\sqrt{n}}\left(T(2n/3) + T(2n/3)\right) \quad \rightarrow T(n) = 2^{O(\sqrt{n})}$

Algorithm `planaris`$(G = (V, E))$, Assumes G planar
Output: The maximum weight of an independent set of G.

1: Find a separator (S, A, B) of G using the planar separator theorem
2: Set $max = -\infty$.
3: **for all** $X \subseteq S$ **do**
4: **if** X is an independent set of $G[S]$ **then**
For $G = (V, E)$ and $X \subseteq V$, $G[X]$ denotes the graph $(X, \{e \in E : e \subseteq X\})$.
$Nb[X]$ denotes the set of vertices adjacent to some vertex in X.
4: Set $weight = w(S) +$ `planaris`$(G[A \setminus Nb(S)]) +$ `planaris`$(G[B \setminus Nb(S)])$
5: **if** $weight > max$ **then** set $max = weight$
6: **return** max

Independent Set in planar graphs

- $T(n) \leq 2^{3\sqrt{n}}\left(T(2n/3) + T(2n/3)\right) \quad \rightarrow T(n) = 2^{O(\sqrt{n})}$
- Optimal under Exponential Time Hypothesis



Treewidth

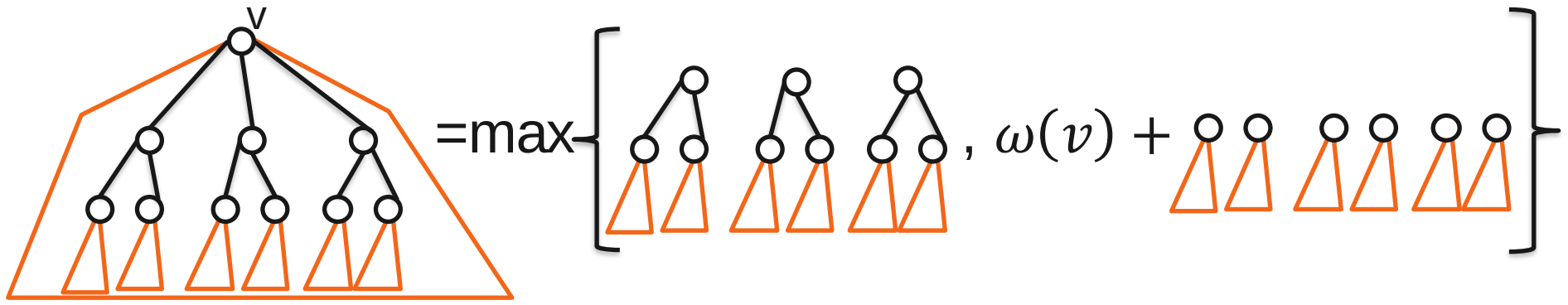
- A measure of how `treelike' a graph is
- Recursive decomposition of a graph by separators

- On a high level, very useful for two reasons:
  - very many NP-hard graph problems can be solved fast on graphs of small treewidth;
  - many graphs of interest have small treewidth.

Weighted Independent Set in Trees

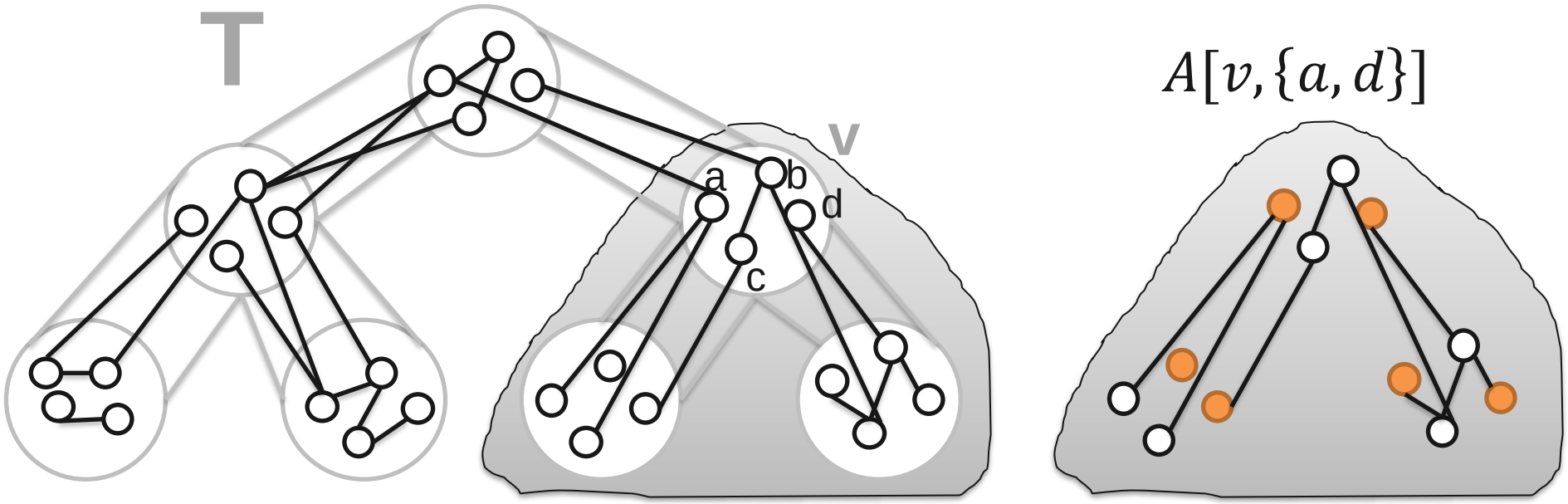
- Let T be a rooted tree, $T[v]$ the subtree rooted at v ∈ V.
- Define $A[v]$ as the max weight of an ind. set of $T[v]$, then

$$A[v] = \begin{cases} \omega(v), & \text{if } T[v] \text{ is a single node,} \\ \max\left\{ \sum_{c \in ch(v)} A[c], \omega(v) + \sum_{c_1 \in ch(v)} \sum_{c_2 \in ch(c_1)} A[c_2] \right\}, & \text{otherwise.} \end{cases}$$



Weighted Independent Set in Trees

- Let T be a rooted tree, $T[v]$ the subtree rooted at $v \in V$.
- ~~Define $A[v]$ as the max weight of an ind. set of $T[v]$, then~~
- For each bag v in T and subsets X of the vertices in it, store the optimal IS of $T[v]$ containing that set



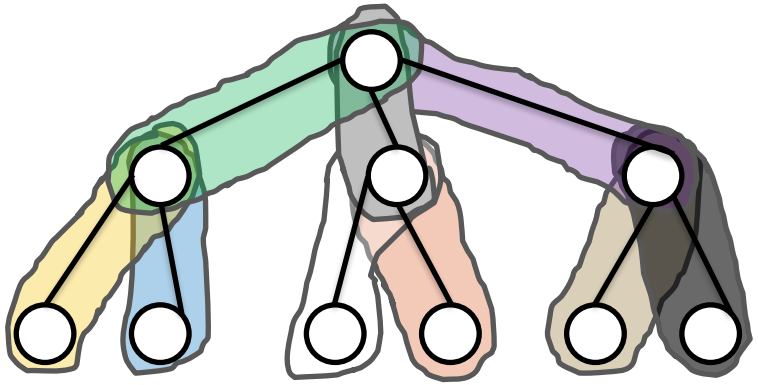
Treewidth

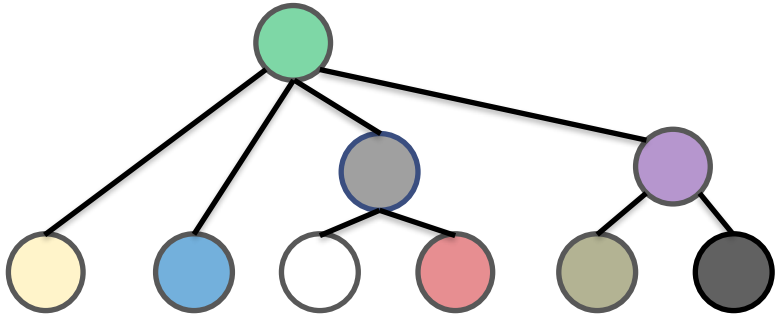
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





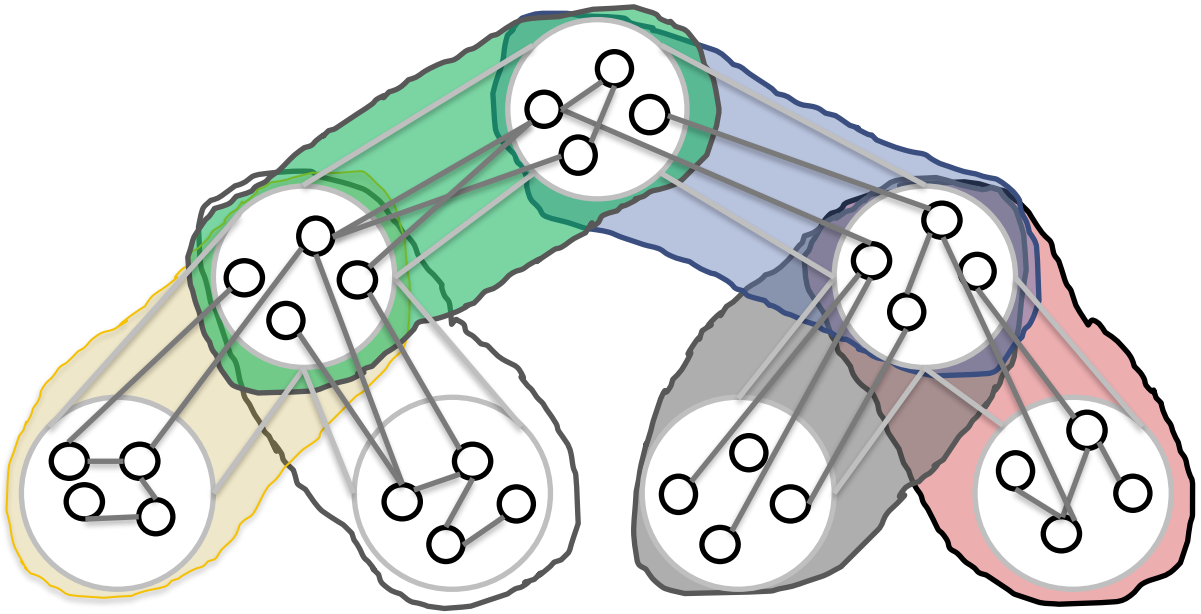
Treewidth

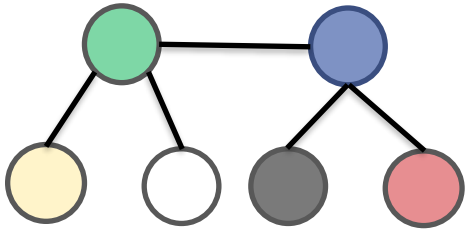
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





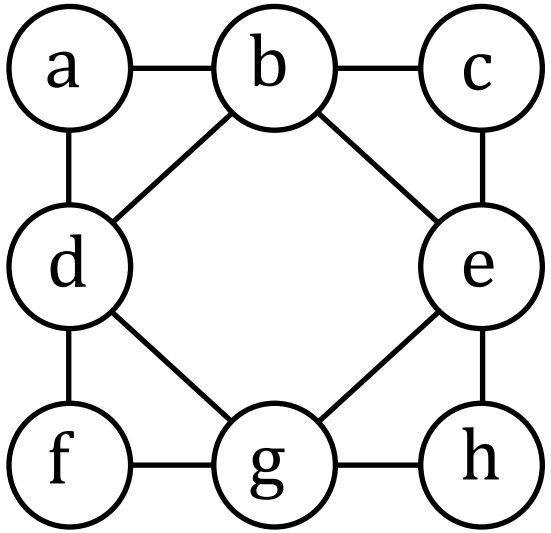
Treewidth

Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.



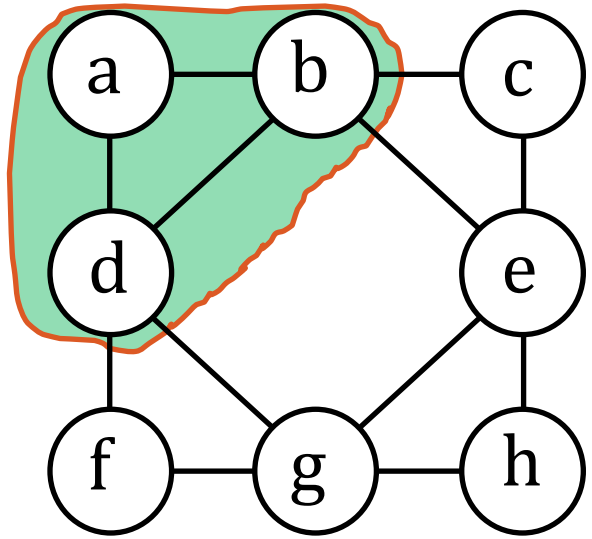
Treewidth

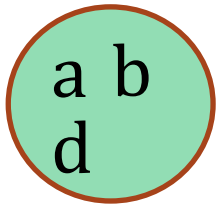
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





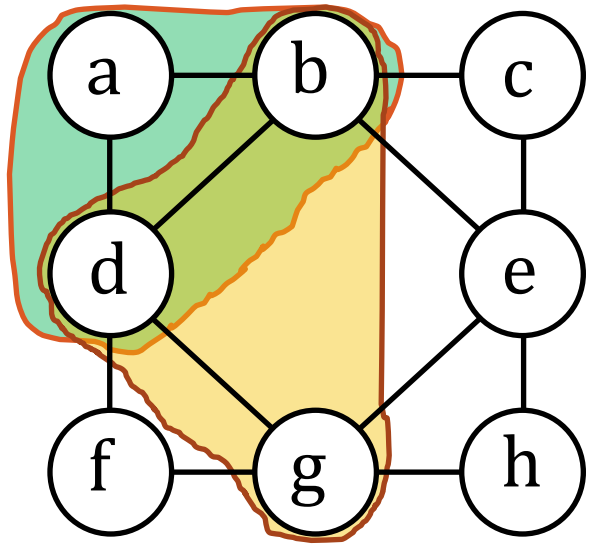
Treewidth

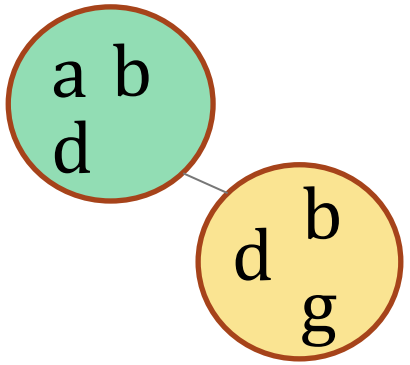
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





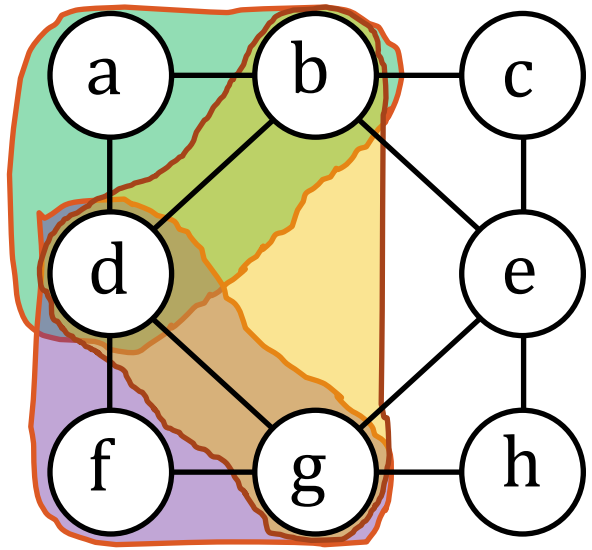
Treewidth

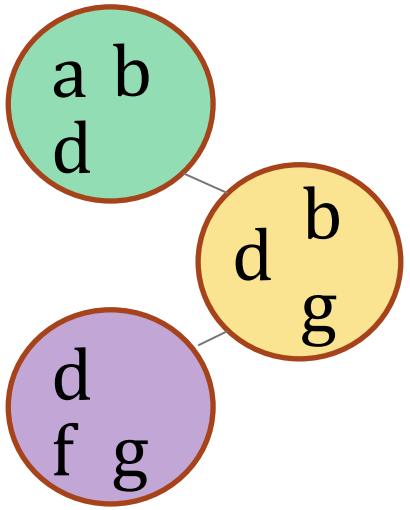
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





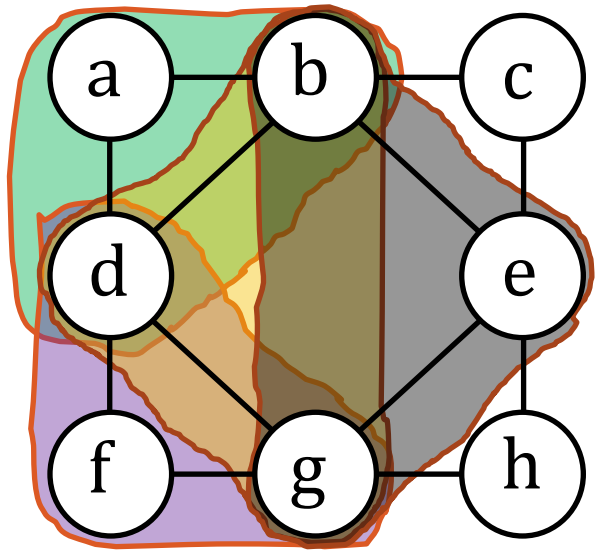
Treewidth

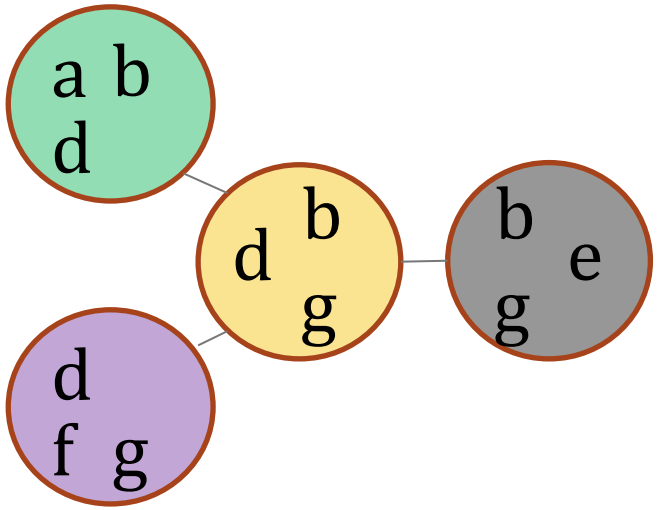
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





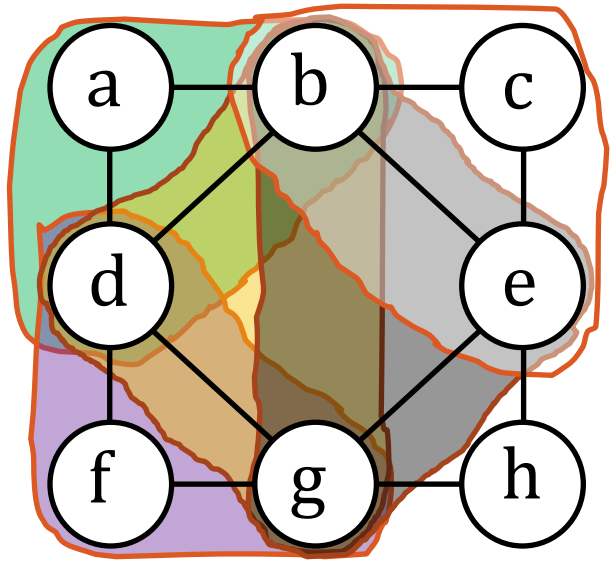
Treewidth

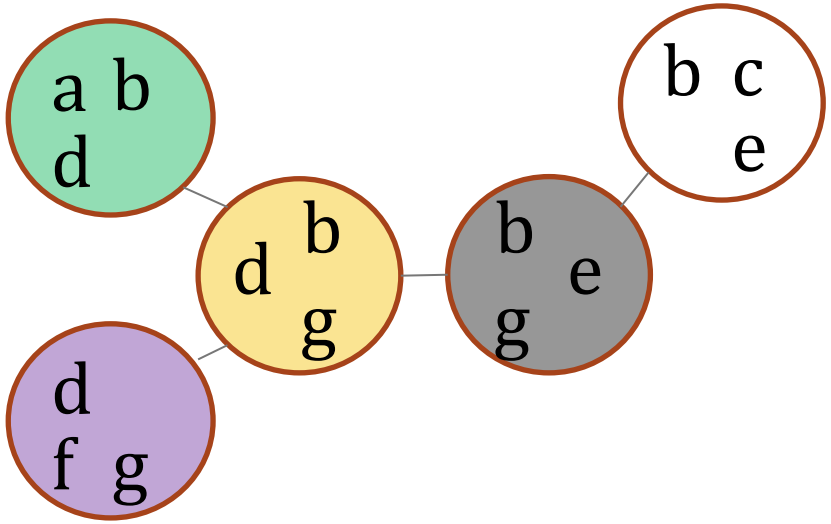
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





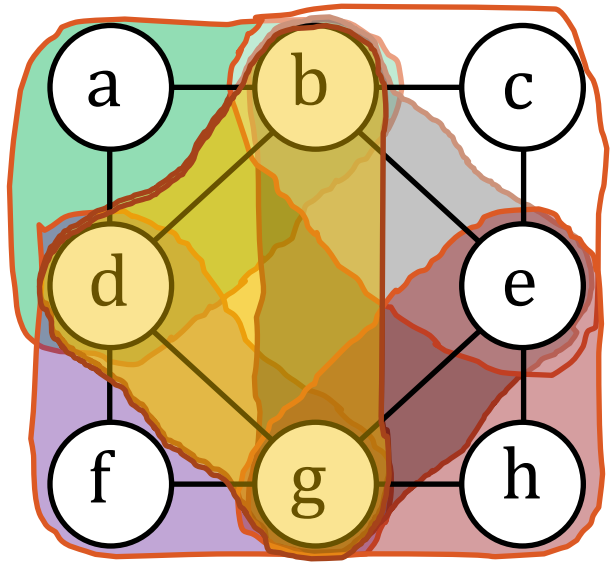
Treewidth

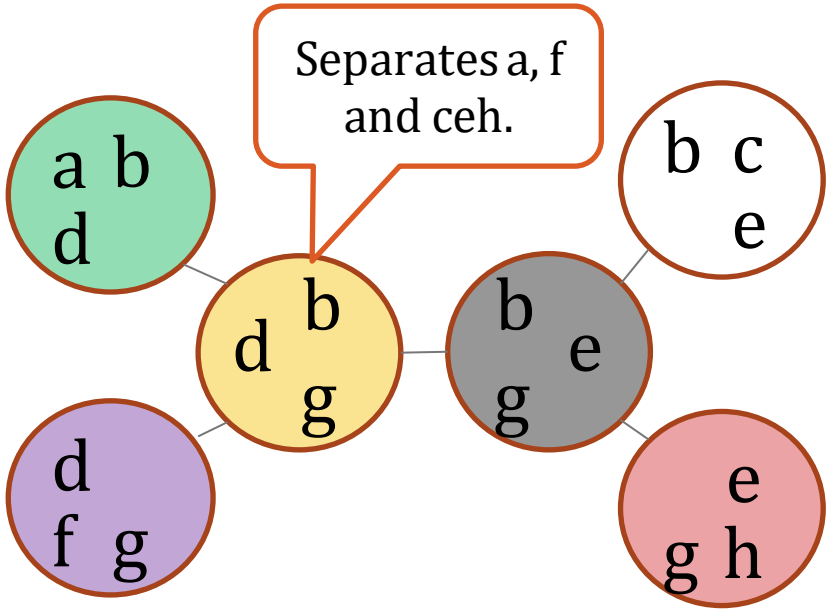
Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \ldots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.





Treewidth

Refer to X_i as a **bag.**

Definition

A **tree decomposition** of graph $G = (V, E)$ is a pair (X, T) where $X = \{X_1, \dots, X_l\}$ with $X_i \subseteq V$ and T a tree with vertex set X such that

1. $\bigcup_{i=1}^{l} X_i = V$,
2. $E \subseteq \bigcup_{i=1}^{l} X_i \times X_i$,
3. $\forall v \in V$: all X_i containing v induce connected subtree if T.

Definition

The **width** of a tree decomposition is $\max_{i=1}^{n} |X_i| - 1$. The **treewidth** of a graph is the minimum width among all possible tree decompositions of G.

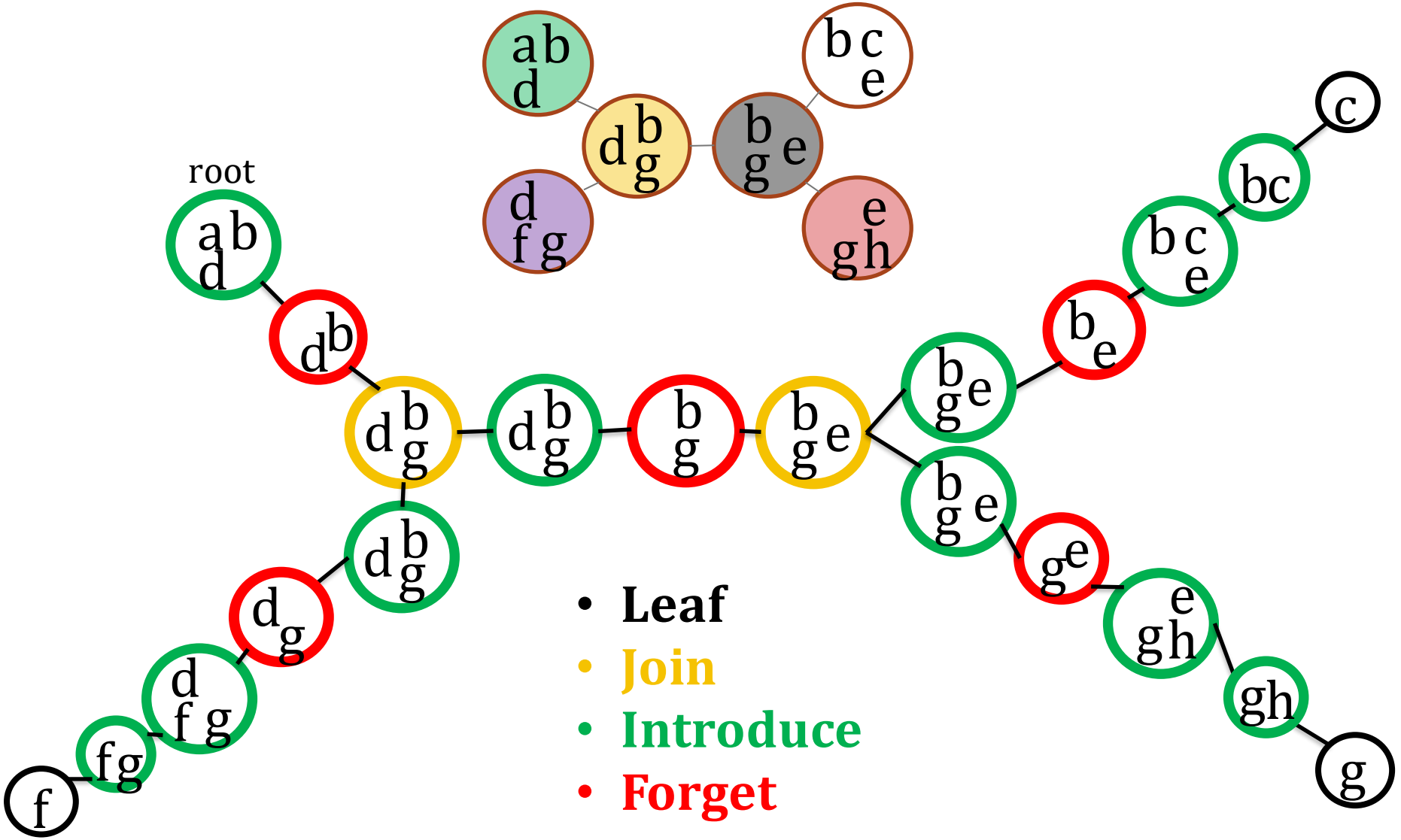
Finding Good Tree Decompositions

- Treewidth problem: Given G and integer w, does G have treewidth w?
- Treewidth param'd by w is NP-hard, but also FPT:
  - Idea is to use iterative compression:
    - A given tree decomposition of width w + 1 can be used to determine the treewidth
    - Full algorithm is technical and beyond the scope of the course
- There are algorithms that given graph of TW w,
  - return TD of width w in $w^{O(w^3)}n$ time.
  - return TD of width $4w + 1$ in $O(3^{3w}wn^2)$ time
  - return TD of width $O(w\sqrt{\lg w})$ in polynomial time

Nice tree decompositions

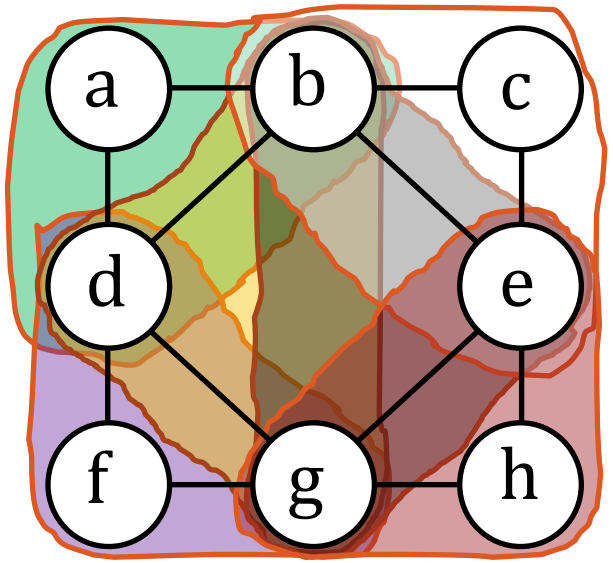
- (T, X) where T is a rooted tree, and four types of bags i:
  - ***Leaf*:** leaf of tree with $|X_i| = 1$
  - ***Join*:** bag with two children j, j' with $X_i = X_j = X_{j'}$,
  - ***Introduce*:** bag with one child j with $X_i = X_j \cup v$ for some vertex v
  - ***Forget*:** bag with one child j with $X_i = X_j \setminus v$ for some vertex v
- Given any TD, we can compute a nice TD of same width in polynomial time.

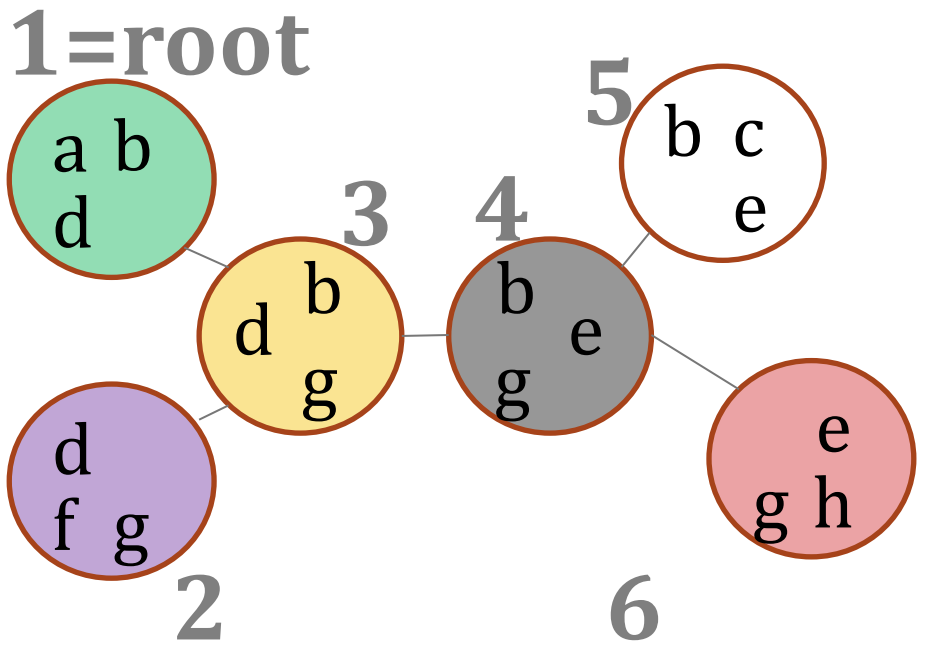
Nice tree decompositions



Max weight IS with given TD in $O^*(2^w)$

- Use given TD to compute a nice TD of width k.
- For a bag i, let $G_i = (V_i, E_i)$ be the subgraph of G formed by all vertices in sets X_j, with $j = i$ or j a descendant of i in the tree.
  - $A[i, Y]$ = maximum weight of an independent set W of G_i with $W \cap X_i = Y$ (which is $-\infty$ if such W does not exist)





- $A[6,\{h\}]=1, A[4,\{e,g\}] = -\infty, A[1,\{b\}] = 3, A[3,\{g\}] = 2$

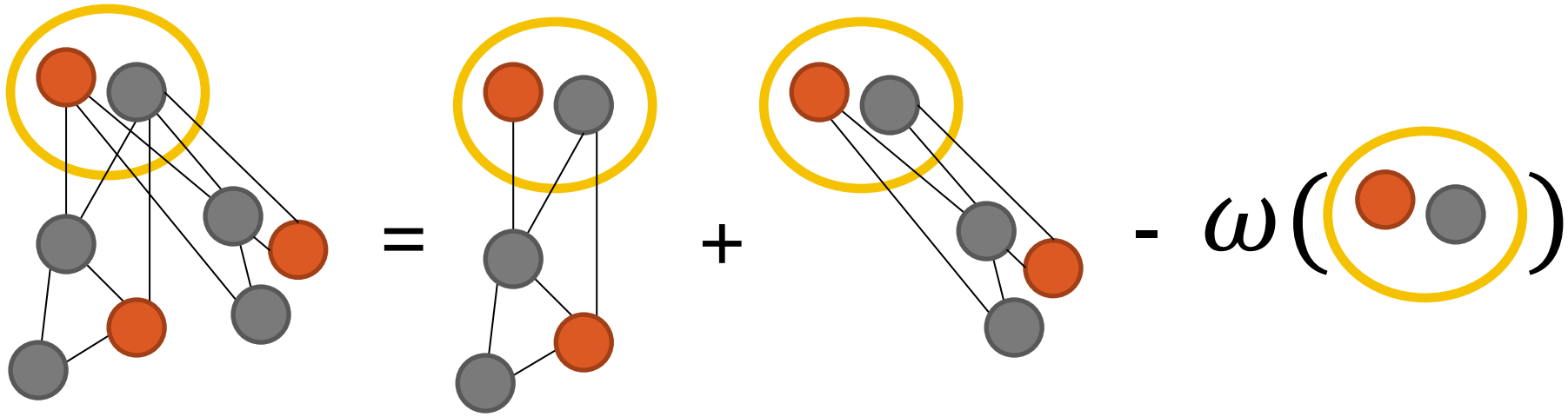
Leaf bags

- Let i be a leaf bag. Say $X_i = \{v\}$
- $A[i, \{v\}] = \omega(v)$
- $A[i, \emptyset] = 0$

G_i is a graph with one vertex

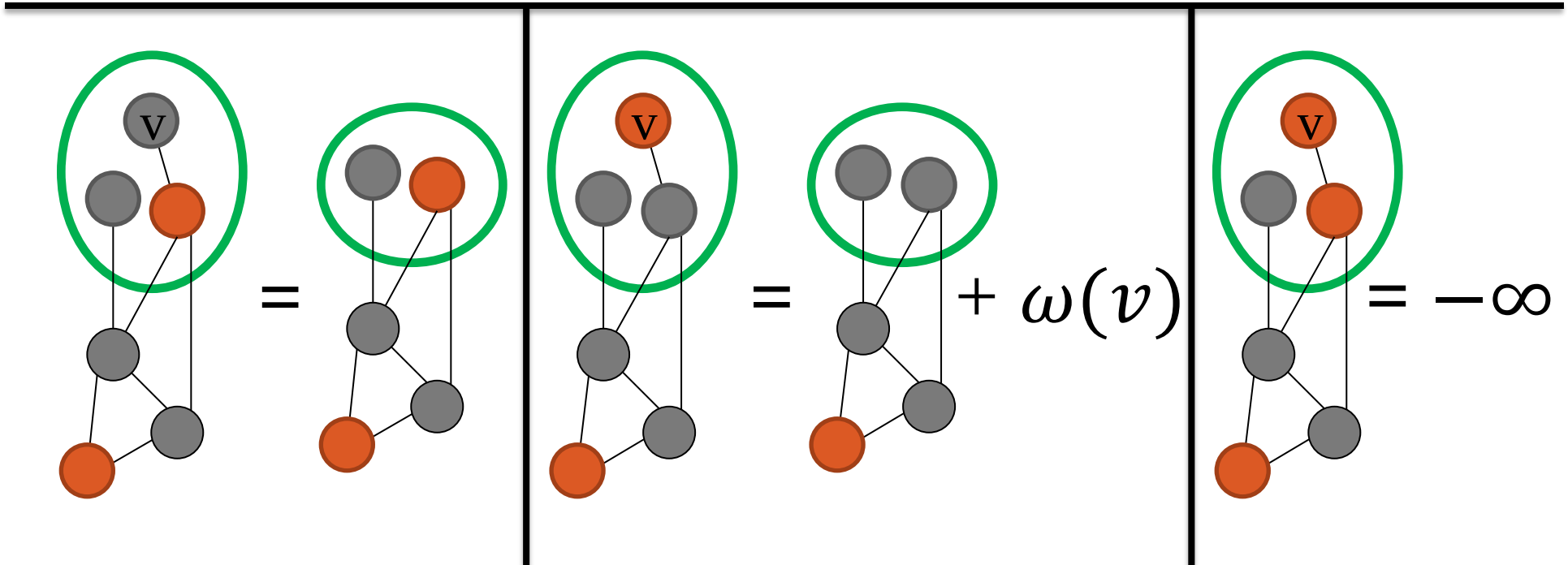
Join bags

- Let i be a join bag with children j_1, j_2.
- $A[i, Y] = A[j_1, Y] + A[j_2, Y] - \sum_{v \in Y} \omega(v)$



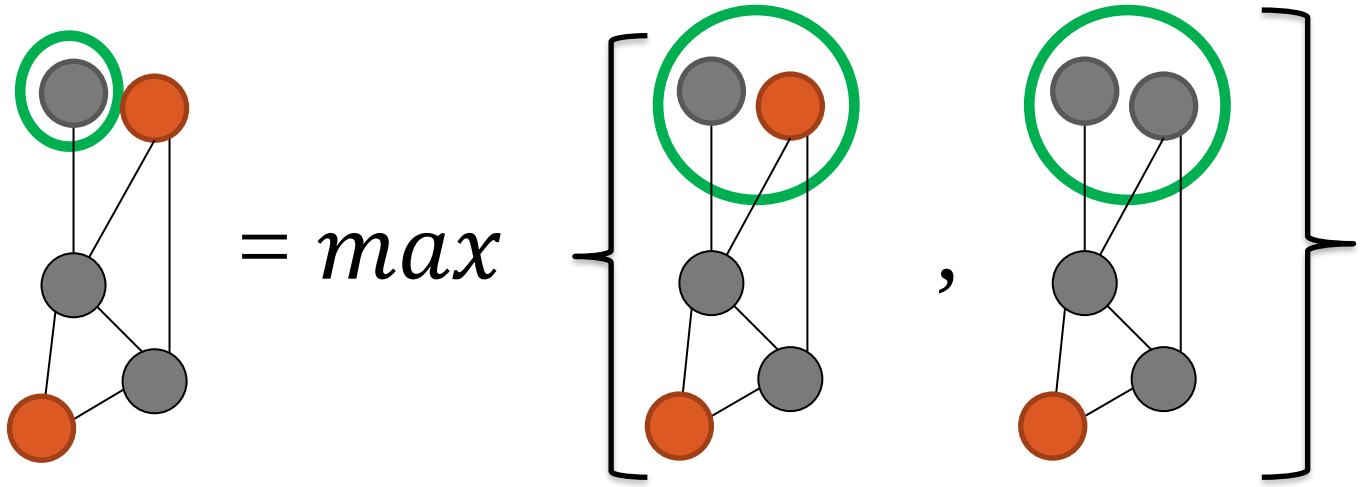
Introduce bags

- Let i be an introduce bag with child j, $X_i = X_j \cup \{v\}$
  1. $A[i,S] = A[j,S]$ for $v \notin S$
  2. $A[i,S \cup \{v\}] = A[j,S] + \omega(v)$, if $N(v) \cap S = \emptyset$
  3. $A[i,S \cup \{v\}] = -\infty$ if $N(v) \cap S \neq \emptyset$



Forget bags

- Let i be a forget bag with child j, $X_i = X_j \setminus \{v\}$
- $A[i, S] = \max\{ A[j, S], A[j, S \cup v]\}$

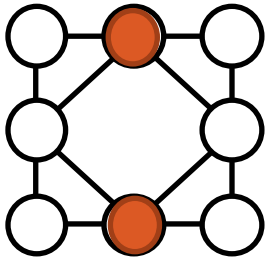


Maximum weighted independent set on graphs with treewidth k

- Compute all table entries in bottom up fashion (starting at the leaves of T) in the usual way.
- Gives $2^w w^{O(1)} n = O^*(2^w)$-time algorithm for max weight ind. set if TD of width w is given.
- FPT algorithms for *very many*, NP-hard graph problems when parameterized by treewidth.

Weighted Dominating Set in Trees

- Dominating set: $X \subseteq V$ such that each vertex is in X or is a neighbor of some vertex in X.

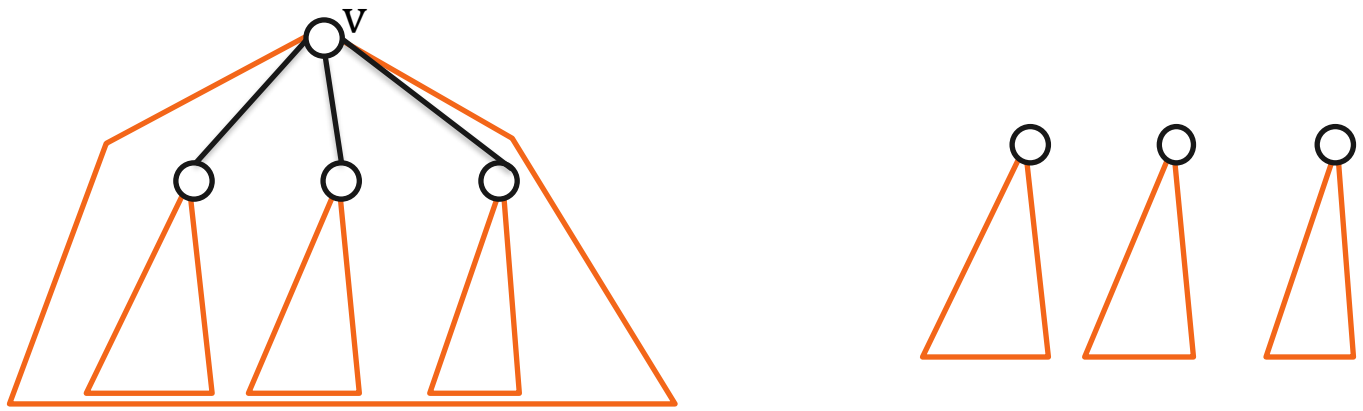


Has a dominating set of size 2

- Can also be solved in $O(n)$ time on trees, and $3^w w^{O(1)} n$ time if a tree decomposition of width w is given ($9^w w^{O(1)} n$ in ex).

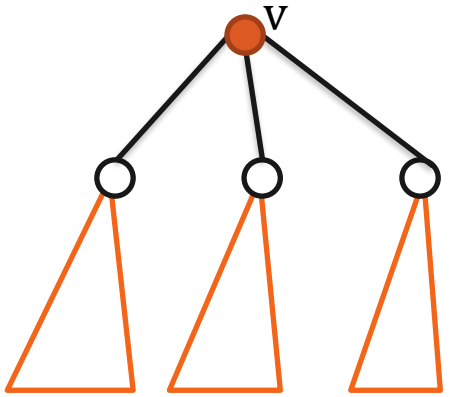
Weighted Dominating Set in Trees

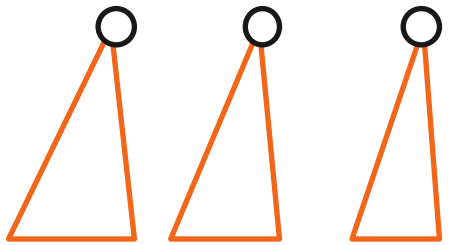
- Dominating set: $X \subseteq V$ such that each vertex is in X or is a neighbor of some vertex in x.
  1. Define $A_D[v]$ as min weight DS of $T[v] \setminus v$
  2. Define $A_I[v]$ as min weight DS of $T[v]$ including v
  3. Define $A_E[v]$ as min weight DS of $T[v]$ excluding v
- If v is leaf, $A_E[v] = \infty, A_D[v] = 0, A_I[v] = \omega(v)$, otherwise:
  - $A_D[v] = \sum_{c \in ch(v)} \min\{A_I[c], A_E[c]\}$



Weighted Dominating Set in Trees

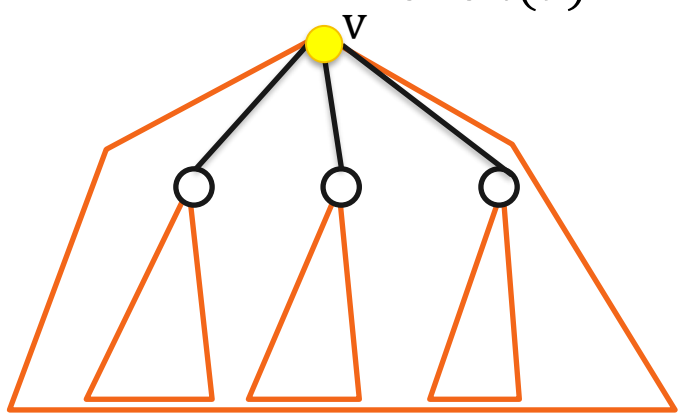
- Dominating set: $X \subseteq V$ such that each vertex is in X or is a neighbor of some vertex in x.
  1. Define $A_D[v]$ as min weight DS of $T[v] \setminus v$
  2. Define $A_I[v]$ as min weight DS of $T[v]$ including v
  3. Define $A_E[v]$ as min weight DS of $T[v]$ excluding v
- If v is leaf, $A_E[v] = \infty, A_D[v] = 0, A_I[v] = \omega(v)$, otherwise:
  - $A_D[v] = \sum_{c \in ch(v)} \min\{A_I[c], A_E[c]\}$
  - $A_I[v] = \omega(v) + \sum_{c \in ch(v)} \min\{A_D[c], A_I[c]\}$

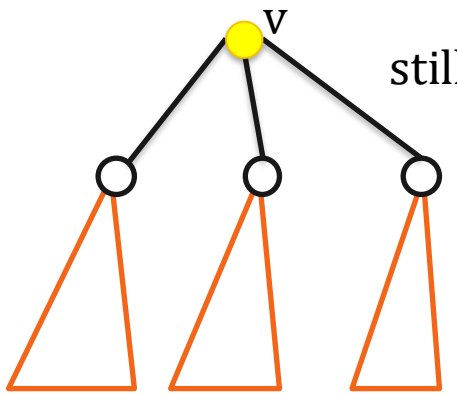




Weighted Dominating Set in Trees

- Dominating set: $X \subseteq V$ such that each vertex is in X or is a neighbor of some vertex in x.
  1. Define $A_D[v]$ as min weight DS of $T[v] \setminus v$
  2. Define $A_I[v]$ as min weight DS of $T[v]$ including v
  3. Define $A_E[v]$ as min weight DS of $T[v]$ excluding v
- If v is leaf, $A_E[v] = \infty, A_D[v] = 0, A_I[v] = \omega(v)$, otherwise:
  - $A_D[v] = \sum_{c \in ch(v)} \min\{A_I[c], A_E[c]\}$
  - $A_I[v] = \omega(v) + \sum_{c \in ch(v)} \min\{A_D[c], A_I[c]\}$
  - $A_E[v] = \min_{c \in ch(v)} \{A_I[c] + \sum_{c' \in ch(v) \setminus c} \min\{A_I[c'], A_E[c']\}$





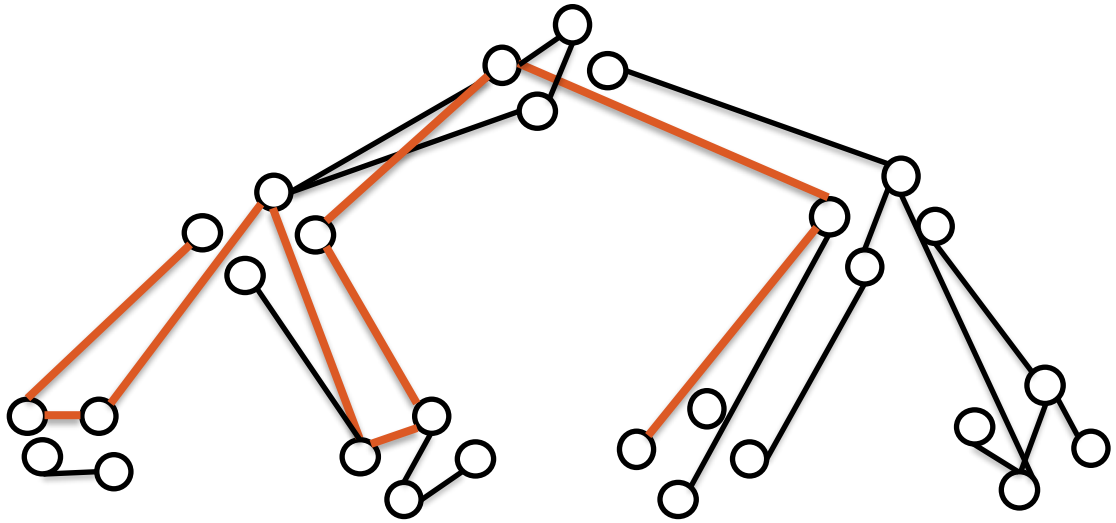
still need to dominate v!

Weighted Dominating Set and Treewidth

- Similar approach as for ind. set.
  - Table $T[i, D, E, I]$ for bag i and partition D,E,I of X_i
  - We need 3 states now for every vertex!
    1. Excluded and dominated already (D)
    2. Excluded and still needs to be dominated
    3. Included (I)
  - Gives $9^w w^{O(1)} n$ time algo, if given TD of width w (exercise, $O^*(3^w)$ table entries, additional time used for join bags)

k-path FPT through treewidth

- In the k-path problem we are given an undirected graph G and need to determine whether there exists a simple path on k vertices.
- NP-complete: $k = n$ is Hamiltonian path
- Similarly as for IS and DS, it can be determined whether a k-path exists in $w^{O(w)}n$ time if a TD of width w if given



DFS /BFS reminder

BFS(G,s)

- Let S be a **queue**
- S.push((ε,s))
- WHILE S not empty
  - (u,v)= S.pop()
  - If v is not labeled `visited'
    - Label v as visited
    - Add (u,v) to T
    - For all neighbors w of v
      - S.push(w)

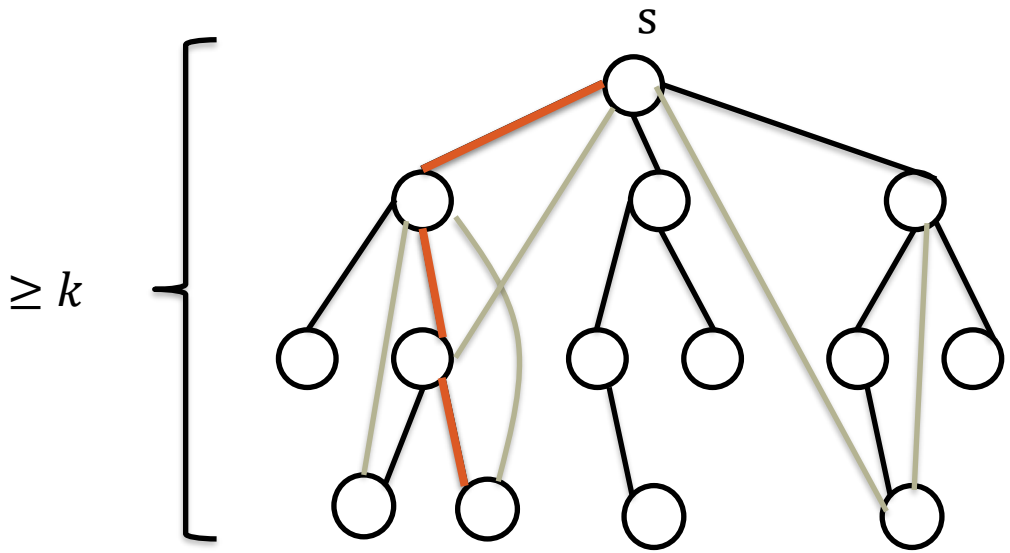
DFS(G,s)

- Let S be a **stack**
- S.push((ε,s))
- WHILE S not empty
  - v= S.pop()
  - If v is not labeled `visited'
    - Label v as visited
    - Add (u,v) to T
    - For all neighbors w of v
      - S.push(w)

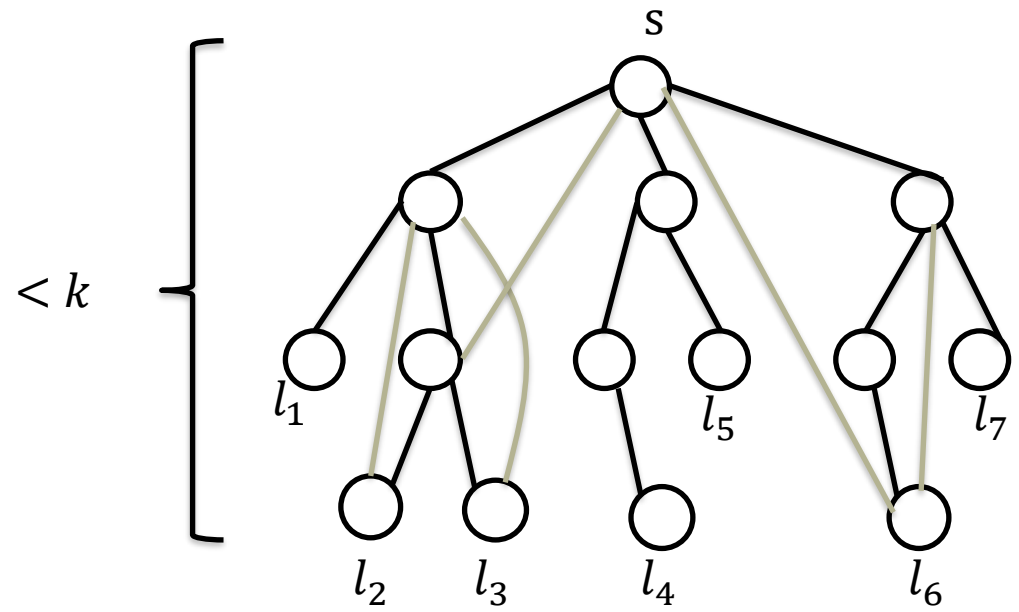
k-path FPT through treewidth

- k-path: given an undirected graph $G = (V, E)$, determine whether there is a simple path on k vertices.
- NP-complete ($k = n$ is Hamiltonian Path).
- Using similar methods as for IS and DS, there is an algorithm determining whether a k-path exists in $w^{O(w)}n$ time, if given TD of width w
- Using this fact we solve k-path in $k^{O(k)}n$ time:
  - Pick $s \in V$ arbitrarily, perform DFS from s. If height of DFS-tree $\geq k$, return YES since path from s to some leaf has length at least k.
  - Otherwise, if the DFS-tree has height $\leq k$, we use it to find a TD of width at most k and we can apply the above algorithm

k-path FPT through treewidth



k-path FPT through treewidth



- Let leaves be $l_1, \ldots, l_p$.
- Create a tree decomposition as follows:
  - T is the path on vertices $X_1, \ldots, X_p$
  - X_i contains all ancestors of l_i in the DFS tree.
- Covers all vertices. Covers all edges since DFS tree.
- Third requirement: T visits children of v consecutively

Bidimensionality

- To solve k-path we exploited that a graph either has:
  1. Low treewidth;
  2. or an easily found k-path.

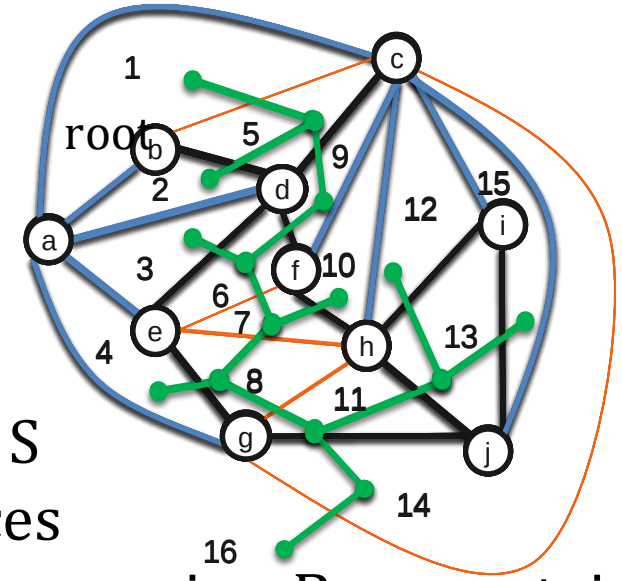
- Similar behavior is called “bidimensionality”
- Seen especially in planar graphs: a planar graph of treewidth $\geq t$ contains (as a minor) a $O(\sqrt{t}) \times O(\sqrt{t})$ grid (Grid Minor Theorem)

TW of planar graphs with shallow spanning tree

Theorem Let G be a planar graph and S be a spanning tree of G of height h. Then given G and S, a TD of G of width at most $3h$ can be found in polynomial time.

Proof idea:

- Triangulate
- Bag for each face
- Edges in T for faces sharing edge not in S
- Bag contains all verts in face and ancestors in S
- Covers all edges and faces



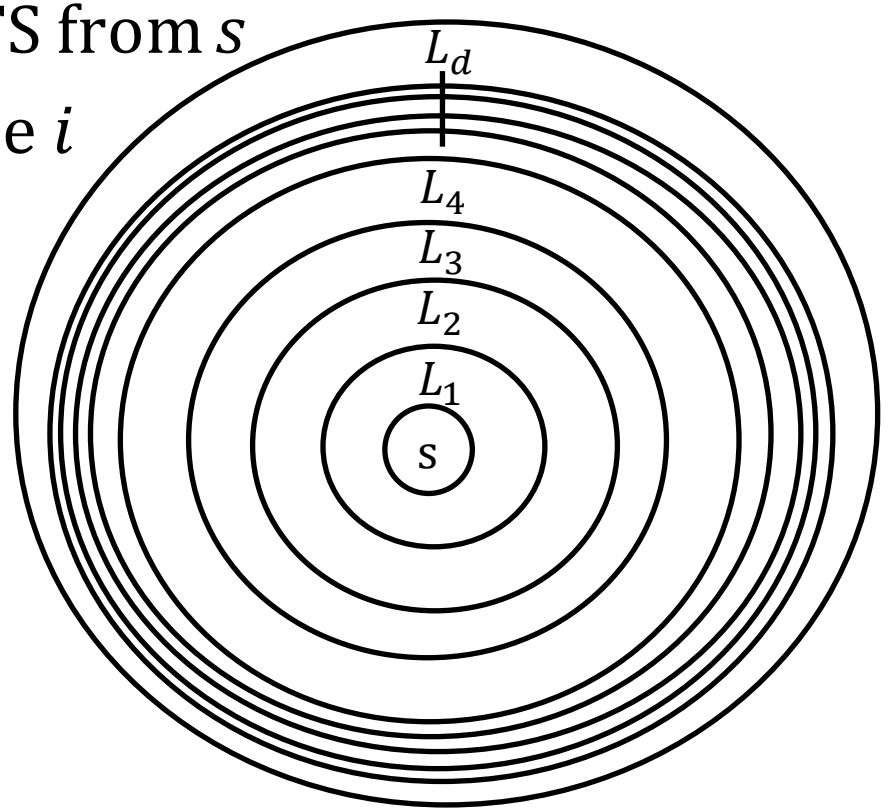
X_1={abc} X_9 ={acdf}
X_2={abd} X_{10} ={acfh}
X_3={ade} X_{11} ={acghj}
X_4={aeg} X_{12} ={achi}
X_5={abcd} X_{13} ={achij}
X_6={acdef} X_{14} ={agcj}
X_7={acefh} X_{15} ={acij}
X_8={acegh} X_{16} ={acg}

- T has no cycle since S is spanning. Bags containing vertex v are connected since we can walk around faces adjacent to $S[v]$
- T connected by previous property since all bags contain root.
- Each bag $\leq 3(h+1)$ since all 3 vertices of face $\leq h$ ancestors.

Baker's approach for approximation in planar graphs

- Given planar graph G, $\omega: V \to \mathbb{N}$ and $\epsilon > 0$. Find IS of weight at least $(1-\epsilon)OPT$ in $O^*(2^{O(1/\epsilon)})$ time

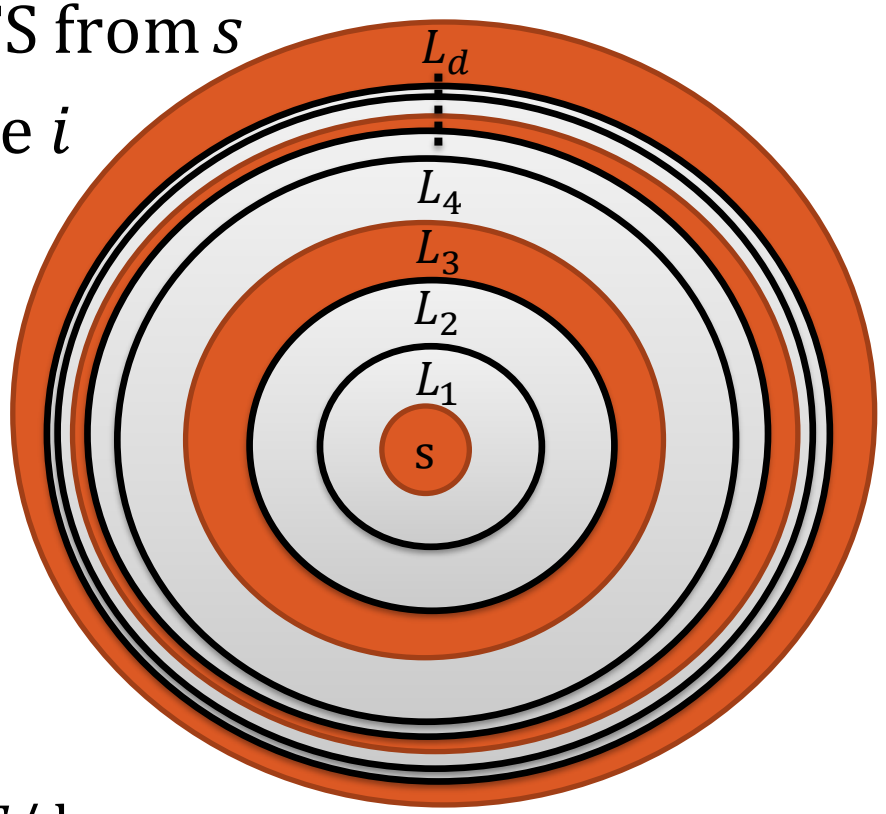
- Pick vertex s arbitrarily. Do BFS from s
- Let L_i be all vertices at distance i
- Let $k = 1/\epsilon$
- Let $V_i = \bigcup_{j \neq i \ (mod\ k)} L_j$
- For $i = 1, \ldots, k$
  - Find max weight IS of $G[V_i]$



Baker's approach for approximation in planar graphs

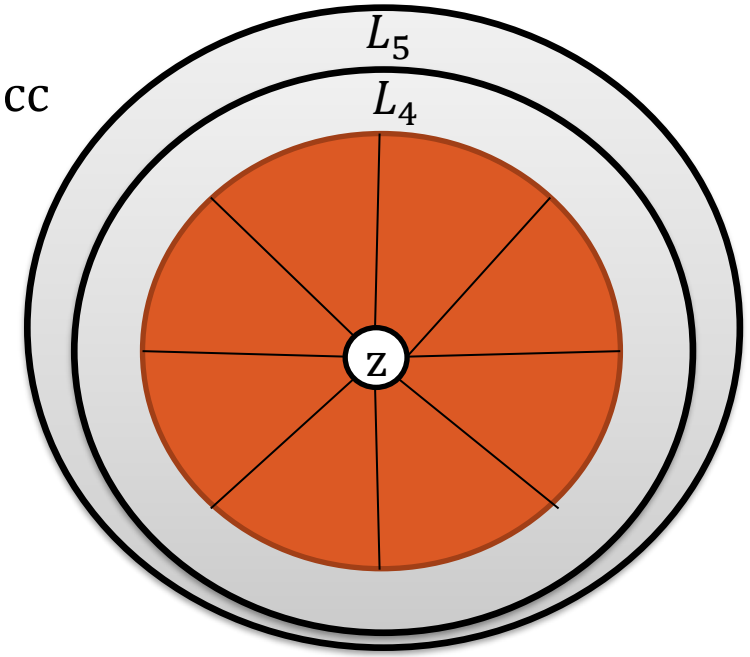
- Given planar graph G, $\omega: V \to \mathbb{N}$ and $\epsilon > 0$. Find IS of weight at least $(1-\epsilon)OPT$ in $O^*(2^{O(1/\epsilon)})$ time

- Pick vertex s arbitrarily. Do BFS from s
- Let L_i be all vertices at distance i
- Let $k = 1/\epsilon$
- Let $V_i = \bigcup_{j \neq i \ (mod\ k)} L_j$
- For $i = 1, \ldots, k$
  - Find max weight IS of $G[V_i]$:
- Return max weight found
- Let $\overline{V_i} = V \setminus V_i$. $\overline{V_i}$ partition V.
  - If I is MWIS, $\exists i$: $\omega(I \setminus V_i) \leq OPT/\ \mathrm{k}$
  - Then MWIS of $G[V_i] \geq OPT - OPT/k = OPT(1-\epsilon)$



Baker's approach for approximation in planar graphs

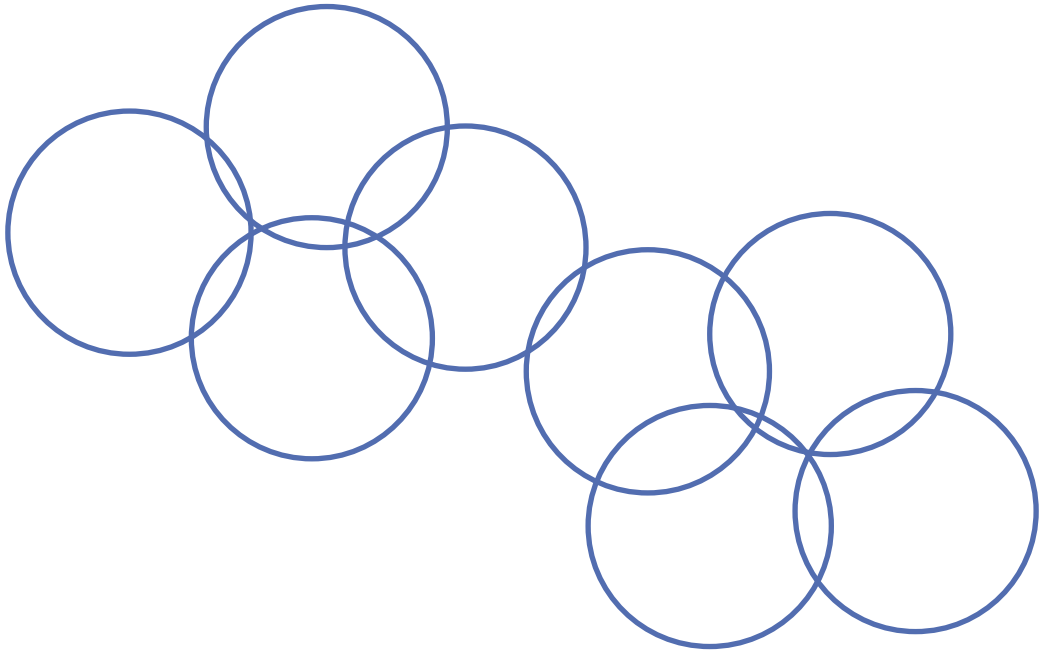
- For a $<$ b $<$ c, L_b separates L_a from L_c by distance property.
- $G[V_i]$ splits in cc's $L_1 \dots l_{j-1}, L_{j+1} \dots L_{j+k-1}, L_{j+k+1} \dots L_{j+2k-1}, \dots$
- Find TD of each cc (can later obtain TD of $G[V_i]$ from this by adding empty bag to connect the tree)
- To find TD of each cc:
  - Let a be the min. int such that L_a is in cc
  - Add a vertex z adjacent to every vertex of L_a
  - This new graph is still planar since we can put z at s's place at the embedding and the BFS tree from s has paths from s to L_a avoiding vertices from L_b with $b > a$
  - Perform BFS from z in this graph.
  - Gives tree of height at most k by definition of layers. Apply previous thm.



- Use the $O^*(2^w)$ time algorithm for MWIS.

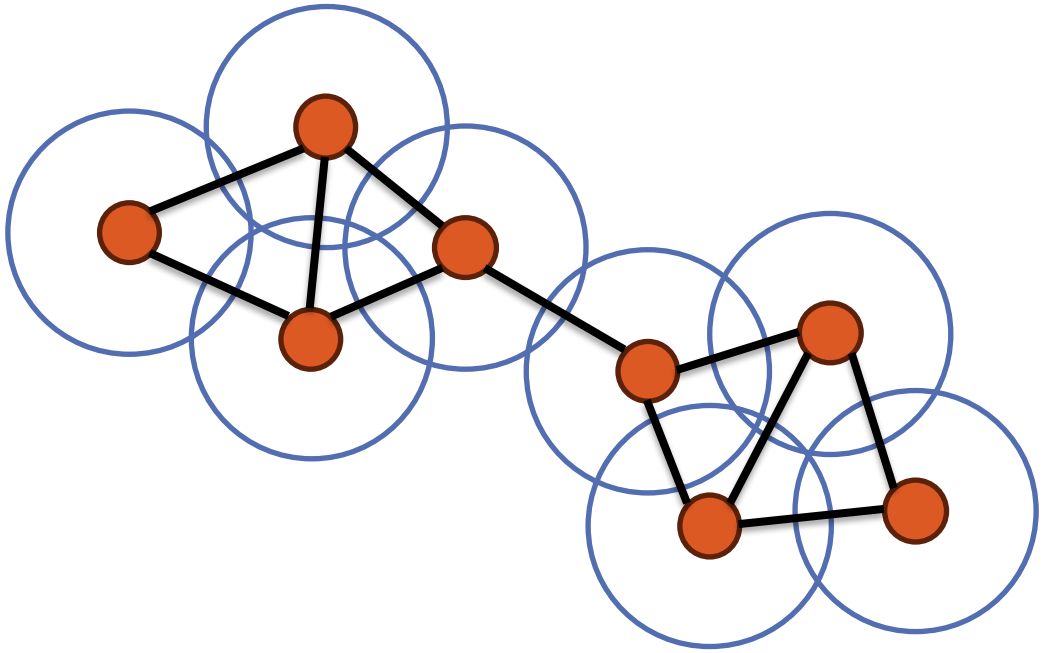
Clique-partitioned Treewidth

- Some graphs are based on (intersections of) geometric objects
- Example: Unit Disk Graphs (radius 1)



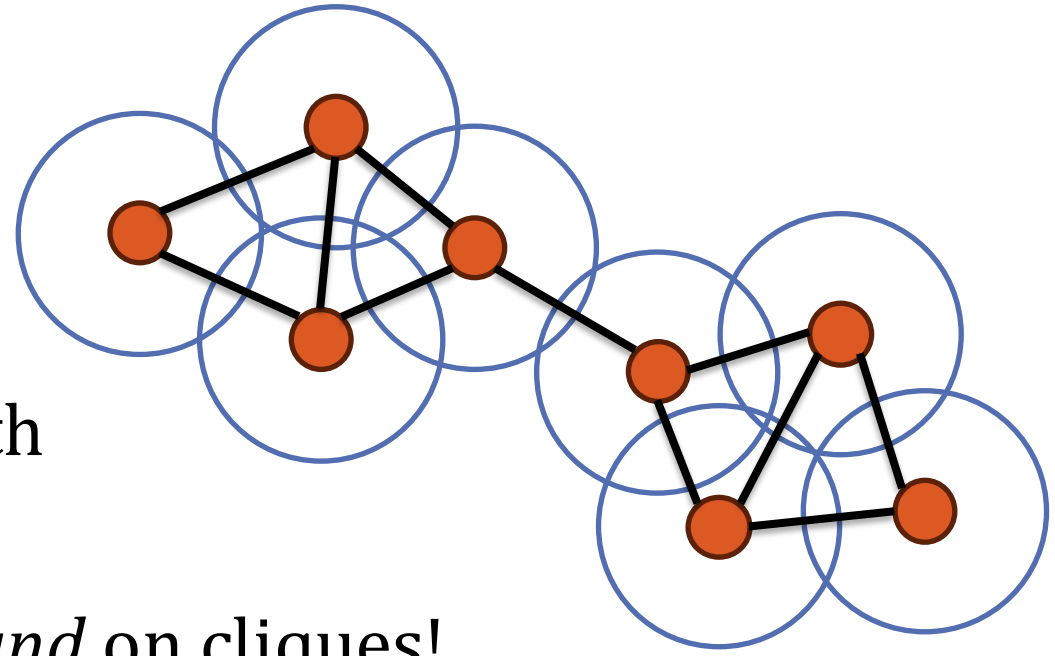
Clique-partitioned Treewidth

- Some graphs are based on (intersections of) geometric objects
- Example: Unit Disk Graphs (radius 1)



Clique-partitioned Treewidth

- Some graphs are based on (intersections of) geometric objects
- Example: Unit Disk Graphs (radius 1)

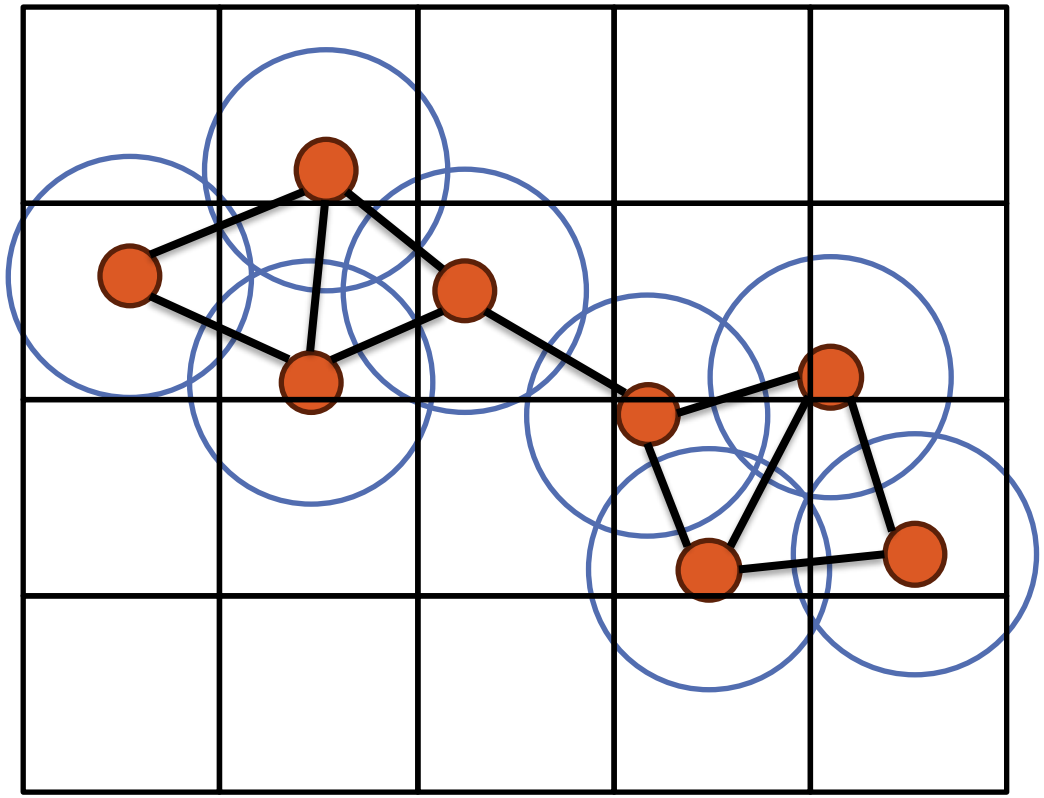


- Independent Set ~ radio towers that do not interfere
- Can have large cliques → large treewidth

- …but Independent Set is easy on trees *and* on cliques!

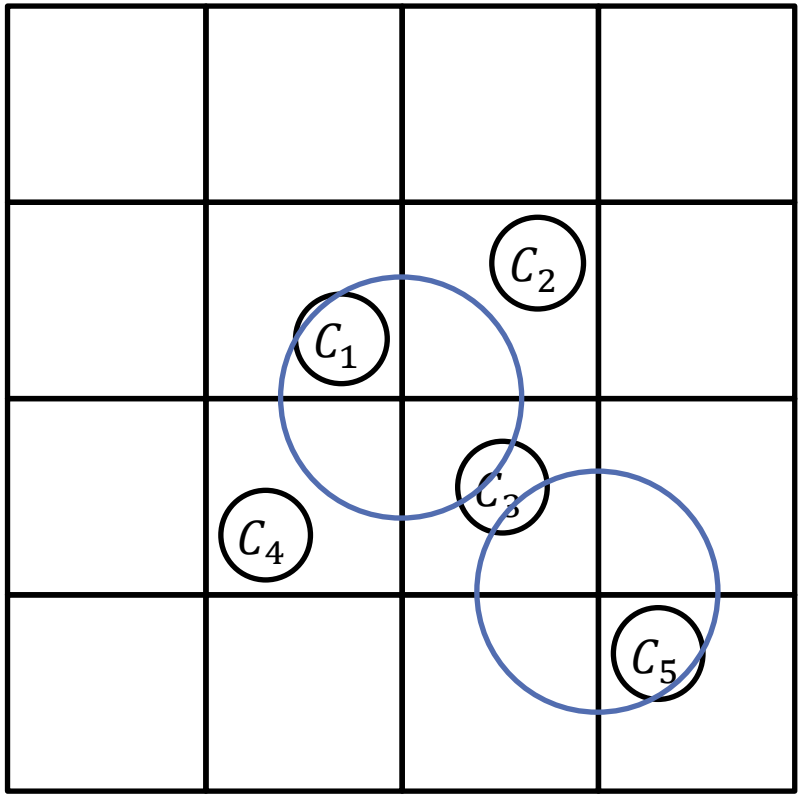
Clique-partition

- Consider a grid with side length $\sqrt{2}$
  - Vertices in the same grid cell form a clique



Clique-partition

- Consider a grid with side length $\sqrt{2}$
  - Vertices in the same grid cell form a clique

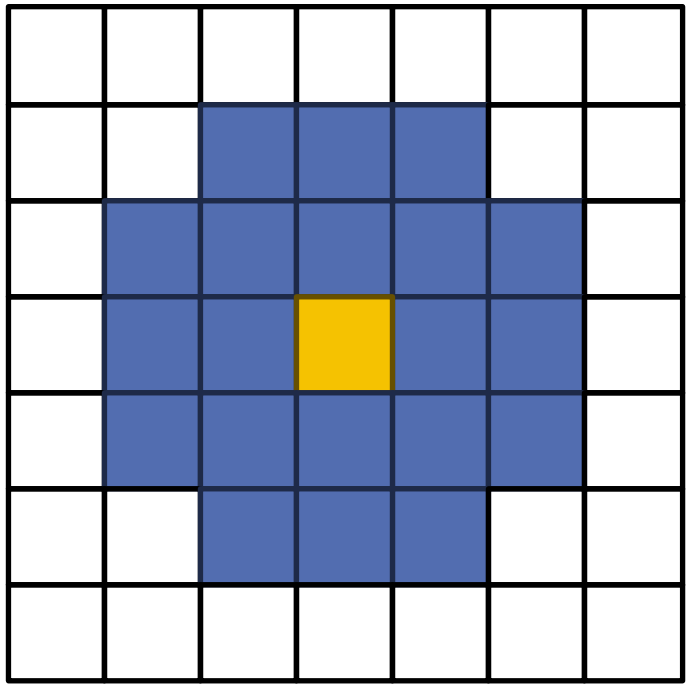


- We obtain a partition of the vertices into cliques
- Define the weight of a clique with n_i vertices to be $\mathrm{w}(n_i) = \log_2(n_i + 1)$

Clique-partition

- Consider a grid with side length $\sqrt{2}$
  - Vertices in the same grid cell form a clique
  - Vertices in each clique can only be adjacent to vertices in 20 other cliques

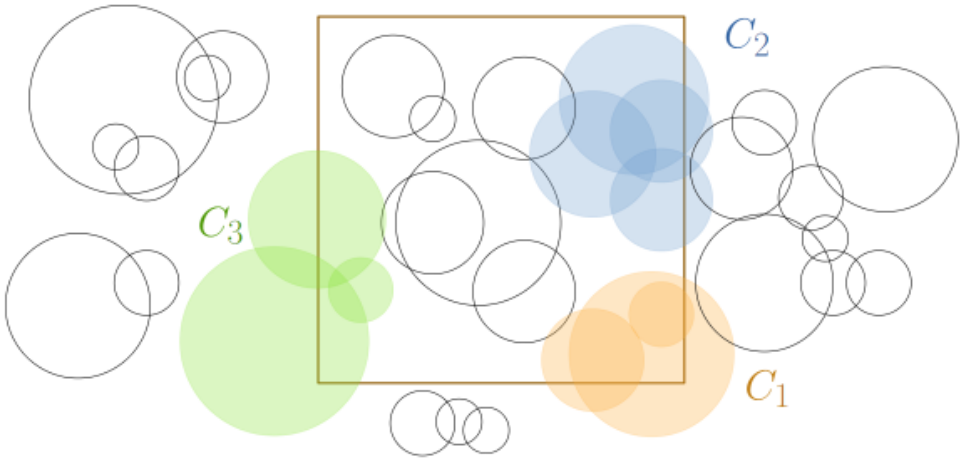
- We obtain a partition of the vertices into cliques
- Define the weight of a clique with n_i vertices to be $\mathrm{w}(n_i) = \log_2(n_i + 1)$



Clique-partitioned separator

- A separator of G is clique-partitioned if it consists of a union of cliques (as defined before)

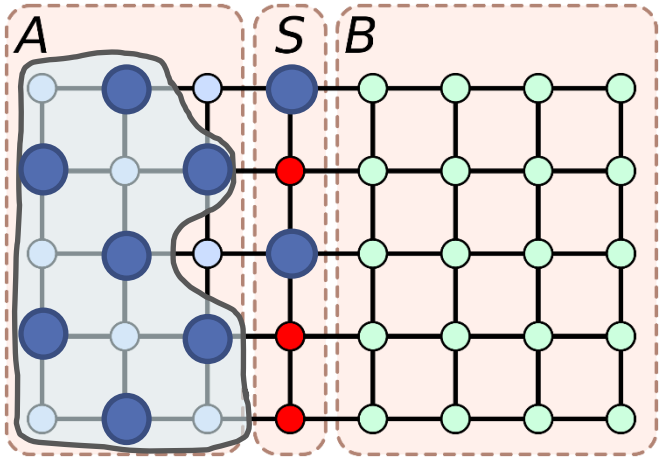
- Theorem: any unit disk graph has a (balanced) clique-partitioned separator of total weight at most $O(\sqrt{n})$



Clique-partitioned treewidth

- A separator of G is clique-partitioned if it consists of a union of cliques (as defined before)
- Theorem: any unit disk graph has a (balanced) clique-partitioned separator of total weight at most $O(\sqrt{n})$
- This notion extends to clique-partitioned treewidth: can find a tree decomposition in which every bag has cliques of weight $O(\sqrt{n})$

Independent Set using Clique Partitions

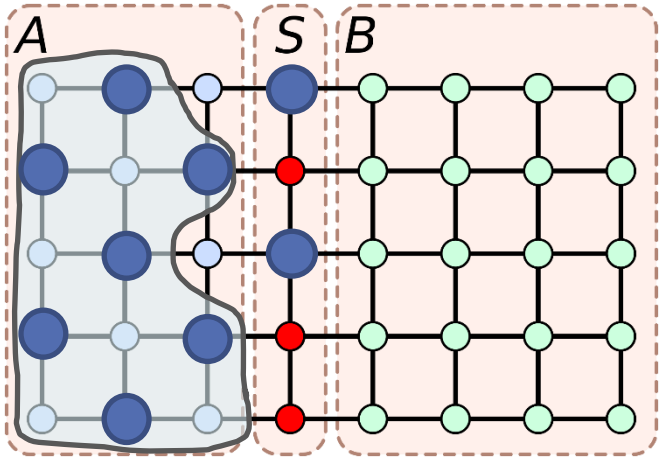


- To solve Independent Set, we considered all $2^{|S|}$ possible subsets of S
- Now consider the case that S is clique partitioned:

$$S = C_1 \cup C_2 \cup C_3 \cup \cdots \cup C_k$$

- Total weight of cliques is $\log(|C_1| + 1) + \log(|C_2| + 1) + \cdots \log(|C_k| + 1) = O(\sqrt{n})$

Independent Set using Clique Partitions



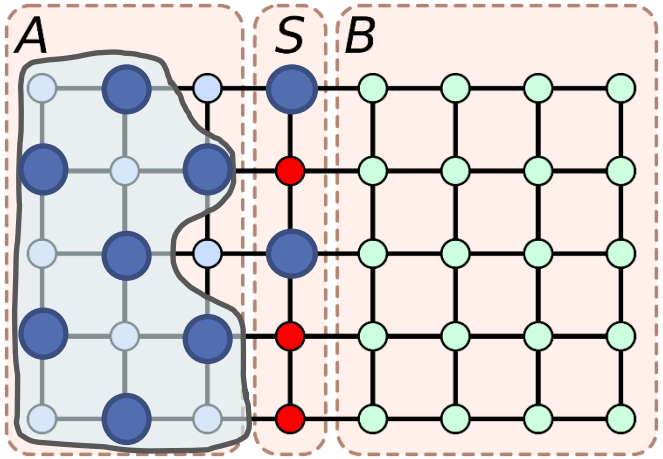
- To solve Independent Set, we considered all $2^{|S|}$ possible subsets of S
- Now consider the case that S is clique partitioned:

$$S = C_1 \cup C_2 \cup C_3 \cup \cdots \cup C_k$$

- Total weight of cliques is $\log(|C_1| + 1) + \log(|C_2| + 1) + \cdots \log(|C_k| + 1) = O(\sqrt{n})$

- Any independent set of S contains ≤ 1 vertex of each clique

Independent Set using Clique Partitions



- To solve Independent Set, we considered all $2^{|S|}$ possible subsets of S
- Now consider the case that S is clique partitioned:

$$S = C_1 \cup C_2 \cup C_3 \cup \cdots \cup C_k$$

- Total weight of cliques is $w(S) = \log(|C_1| + 1) + \log(|C_2| + 1) + \cdots \log(|C_k| + 1) = O(\sqrt{n})$

- Any independent set of S contains ≤ 1 vertex of each clique

$$|IS(S)| = (|C_1| + 1) \cdot (|C_2| + 1) \cdot \ldots \cdot (|C_k| + 1)$$

Independent Set using Clique Partitions

- Total weight of cliques is $\mathrm{w}(\mathrm{S}) = \log(|C_1| + 1) + \log(|C_2| + 1) + \cdots \log(|C_k| + 1) = O(\sqrt{n})$

- Any independent set of S contains ≤ 1 vertex of each clique

$$|IS(S)| = (|C_1| + 1) \cdot (|C_2| + 1) \cdot \ldots \cdot (|C_k| + 1) =$$
$$2^{\log(|C_1|+1)} \cdot 2^{\log(|C_2|+1)} \cdot \ldots \cdot 2^{\log(|C_k|+1)} =$$
$$2^{\log(|C_1|+1)+\log(|C_2|+1)+\cdots\log(|C_k|+1)} =$$
$$2^{O(\sqrt{n})}$$

Results using treewidth

- Many of the problems we saw can be solved in $2^{O(tw)}$
- →$2^{O(\sqrt{n})}$ in planar graphs

- But also: $2^{O(\sqrt{n})}$ in unit disk graphs

- We can consider d-dimensional ball graphs: $2^{O(n^{1-1/d})}$