Advanced Algorithms (II)

Yijia Chen
Fudan University
Review
Basics

1. Big-$O$ notation
2. Divide and conquer
3. DFS and BFS
4. Independent set in trees
Treewidth
Tree decompositions of graphs

Definition

Let $G$ be a graph. A tree decomposition of $G$ is a tuple $(T, (B_t)_{t \in V(T)})$, where $T$ is a tree and $B_t$ the bag at $t$ such that the following conditions are satisfied:

(T1) For every $v \in V(G)$ the set

$$T_v := \{ t \in V(T) \mid v \in B_t \}$$

is nonempty and connected in $T$, i.e., $T[T_v]$ is a subtree of $T$.

(T2) For every $e \in E(G)$ there exists a $t \in V(T)$ such that $e \subseteq B_t$. 
1. The complete graphs $K_n$ for $n \in \mathbb{N}$.

2. The trees.

3. The grids $G_{n \times n}$ for $n \in \mathbb{N}$.
The **width** of a tree decomposition \((\mathcal{T}, (B_t)_{t \in V(\mathcal{T})})\) is

\[
\text{width}(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}) := \max \{ |B_t| - 1 \mid t \in V(\mathcal{T}) \}.
\]

The **treewidth** of \(G\) is

\[
\text{tw}(G) := \min \left\{ \text{width}(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}) \mid (\mathcal{T}, (B_t)_{t \in V(\mathcal{T})}) \text{ is a tree decomposition of } G \right\}.
\]
1. $\text{tw}(K_n) = n - 1$ for the complete graphs $K_n$.

2. $\text{tw}(T) = 1$ for every tree $T$ of size at least 2.

3. $\text{tw}(G_{n \times n}) = n$ for every grid $G_{n \times n}$. 
Smooth tree decomposition

Definition
A tree decomposition \((T, (B_t)_{t \in V(T)})\) is smooth if for every \(\{t, t'\} \in E(T)\) we have
\[
\left|B_t \setminus B_{t'}\right| = \left|B_{t'} \setminus B_t\right| = 1.
\]

Theorem
Every tree decomposition can be efficiently transferred to a smooth one of the same width.

Theorem
Every graph \(G\) has a smooth tree decomposition of width \(tw(G)\).
Make tree decomposition smooth

Let \((T, (B_t)_{t \in V(T)})\) be a tree decomposition of width \(w\).

1. **Make bags equal size:** We choose a node \(r \in V(T)\) with \(|B_r| = w + 1\) as the root. Let \(t\) be a child of \(r\) with \(|B_t| \leq w\). Clearly

\[
|B_r \setminus B_t| + |B_t| \geq w + 1.
\]

We add \(w + 1 - |B_t|\) vertices in \(B_r \setminus B_t\) to \(B_t\). After repeating this procedure recursively from the root to leaves, every bag has size \(w + 1\).

2. **Remove repetition:** If there is an edge \(\{t, t'\} \in E(T)\) with \(B_t = B_{t'}\), then we merge \(t'\) with \(t\).

3. **Interpolation:** Let \(\{t, t'\} \in E(T)\) with \(|B_t \cap B_{t'}| < w\), i.e.,

\[
B_t \setminus B_{t'} = \{u_1, \ldots, u_\ell\} \quad \text{and} \quad B_{t'} \setminus B_t = \{v_1, \ldots, v_\ell\}
\]

for some \(\ell > 2\) and pairwise distinct \(u_1, \ldots, u_\ell, v_1, \ldots, v_\ell\). We insert new nodes \(t_1, \ldots, t_{\ell - 1}\) between \(t\) and \(t'\) with

\[
B_{t_i} := (B_t \cap B_{t'}) \cup \{v_1, \ldots, v_i, u_{i+1}, \ldots, u_\ell\}
\]

for every \(i \in [\ell - 1]\).
The size of smooth tree decompositions

**Theorem**

For every smooth tree decomposition \((T, (B_t)_{t \in V(T)})\) of \(G\) we have

\[ |V(T)| \leq |V(G)|. \]

**Theorem**

\[ |E(G)| \leq tw(G) \cdot |V(G)|. \]
\( \text{tw}(\mathcal{K}_n) = n - 1 \)

**tw(\mathcal{K}_n) \leq n - 1:** Take a tree decomposition with a singleton tree.

**tw(\mathcal{K}_n) \geq n - 1:** Let \((T, (B_t)_{t \in V(T)})\) be a smooth tree decomposition of \(\mathcal{K}_n\) of width \(\text{tw}(\mathcal{K}_n)\). We show that there exists a \(B_t\) with \(|B_t| = n\).

Trivial if \(|V(T)| = 1\). Otherwise choose a leaf \(t\) and let \(t'\) be its parent in \(T\). By the smoothness

\[ B_t \setminus B_{t'} = \{v\} \text{ for some } v \in V(\mathcal{K}_n). \]

Since \(v\) is adjacent to every other vertex in \(\mathcal{K}_n\), we see that

\[ B_t = V(\mathcal{K}_n). \]
Helly property for trees

Theorem
Let $T$ be a tree and $T_1, \ldots, T_n$ subtrees of $T$ such that

$$V(T_i) \cap V(T_j) \neq \emptyset$$

for every $i, j \in [n]$. Then

$$\bigcap_{i \in [n]} V(T_i) \neq \emptyset.$$
We prove by induction on the size of $\mathcal{T}$.

Trivial for $|V(\mathcal{T})| = 1$.

Otherwise, let $t$ be a leaf of $\mathcal{T}$. If $t \in V(\mathcal{T}_i)$ for every $i \in [n]$, then we are done.

Now assume $t \notin V(\mathcal{T}_i)$ for some $i \in [n]$. Consider

$$\mathcal{T} \setminus \{t\}, \mathcal{T}_1 \setminus \{t\}, \ldots, \mathcal{T}_n \setminus \{t\}.$$  

Then

- every $\mathcal{T}_i \setminus \{t\}$ is a (nonempty) subtree of $\mathcal{T} \setminus \{t\}$;
- $V(\mathcal{T}_i \setminus \{t\}) \cap V(\mathcal{T}_j \setminus \{t\}) \neq \emptyset$ for every $i, j \in [n]$.

The result follows from the induction hypothesis. \qed
tw(\mathcal{K}_n) = n - 1, again

**Theorem**
Let $\mathcal{G} = (V, E)$ be a graph and $S \subseteq V$ a clique. Then for every tree decomposition $(T, (B_t)_{t \in V(T)})$ there is a node $t \in V(T)$ with $S \subseteq B_t$.

**Proof.**
For every $v \in V$ recall

$$T_v := \{ t \in V(T) \mid v \in B_t \}$$

induces a subtree $T_v := T[T_v]$ of $T$.

Clearly for every $u, v \in S$ we have

$$V(T_u) \cap V(T_v) = T_u \cap T_v \neq \emptyset,$$

since there is an edge $\{u, v\}$ in $\mathcal{G}$.

The result follows from Helly property. \qed
Computing the treewidth

Theorem (Bodlaender, 1996)

The problem

<table>
<thead>
<tr>
<th>TREEWIDTH</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A graph $G$ and a number $k \in \mathbb{N}$.</td>
</tr>
<tr>
<td><strong>Problem:</strong> Decide whether $\text{tw}(G) \leq k$ and if so output a tree decomposition of $G$ with width $\leq k$.</td>
</tr>
</tbody>
</table>

can be computed in time

$$2^{O(1)} \cdot \|G\|.$$ 

Corollary

For every $k \in \mathbb{N}$ there is a linear time algorithm which on every graph $G$ either outputs a tree decomposition of $G$ of width $\leq k$ or reports that $\text{tw}(G) > k$. 
Independent sets via tree decompositions
Let $G$ be a graph and $(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})})$ a smooth tree decomposition of $G$. And let $w := \text{width}(\mathcal{T}, (B_t)_{t \in V(\mathcal{T})})$.

We fix an arbitrary node $r \in V(\mathcal{T})$ as the root of $\mathcal{T}$.

Let $t \in V(\mathcal{T})$. We define $G_t$ as the induced subgraph of $G$ on vertices in $B_t$. Furthermore, $G_{\leq t}$ is the induced subgraph of $G$ on vertices in

$$B_t \cup \bigcup \text{descendants } t' \text{ of } t.$$
By dynamic programming we compute for every $t \in V(T)$ and every $X \subseteq B_t$ independent in $G_t$:

$$I(t, X) := \text{size of a largest independent set } I \text{ of } G_{\leq t} \text{ with } I \cap B_t = X$$

$$= |X| + \sum_{\text{children } t' \text{ of } t} \max \{ I(t', X') - |X' \cap X| \mid X' \cap B_t = X \cap B_{t'} \}.$$ 

Note there are at most

$$|V(T)| \cdot 2^{w+1} \leq |V(G)| \cdot 2^{w+1}$$

many $I(t, X)$. 
Theorem

For every $k \in \mathbb{N}$ there is a linear time algorithm which on every graph $G$ with $tw(G) \leq k$ outputs a largest independent set in $G$. 
Partial $k$-Trees
\textbf{Definition}

Let $k \in \mathbb{N}$. Then the set of $k$-trees is defined as follows.

\begin{itemize}
  \item[(K1)] A complete graph $\mathcal{K}_{k+1}$ is a $k$-tree.
  \item[(K2)] Let $\mathcal{G}$ be a graph and $v \in V$ such that \(\mathcal{N}^\mathcal{G}[v]\) is isomorphic to $\mathcal{K}_{k+1}$, where $\mathcal{N}^\mathcal{G}[v]$ is the induced subgraph of $\mathcal{G}$ on
  \[\mathcal{N}^\mathcal{G}[v] := \{u \in V(\mathcal{G}) \mid \{u, v\} \in E(\mathcal{G})\} \cup \{v\}\]
  
  - $\mathcal{N}^\mathcal{G}[v]$ is isomorphic to $\mathcal{K}_{k+1}$, where $\mathcal{N}^\mathcal{G}[v]$ is the induced subgraph of $\mathcal{G}$ on
  
  - $\mathcal{G}[V(\mathcal{G}) \setminus \{v\}]$ is a $k$-tree.

  \end{itemize}

Then $\mathcal{G}$ is a $k$-tree.

\textbf{Definition}

A graph is a \underline{partial $k$-tree} if it is a subgraph of a $k$-tree.
Partial \( k \)-trees and bounded treewidth

**Theorem**
A graph \( G \) is a partial \( k \)-tree if and only if \( \text{tw}(G) \leq k \).

**Lemma**
Let \( G \) be a subgraph of \( H \), i.e., \( V(G) \subseteq V(H) \) and \( E(G) \subseteq E(H) \). Then \( \text{tw}(G) \leq \text{tw}(H) \).

**Theorem**
1. Every graph of treewidth \( \leq k \) is a partial \( k \)-tree.
2. Every \( k \)-tree has a tree decomposition of width \( \leq k \).
Proof

Let $G$ be a graph with $\text{tw}(G) \leq k$. Moreover, let $\mathcal{T} = (\mathcal{T}, (B_t)_{t \in V(\mathcal{T})})$ be a smooth tree decomposition of $G$ of width $k$.

From $\mathcal{T}$ we define a graph $H_{\mathcal{T}}$ by induction on $V(\mathcal{T})$ such that

(H1) $H_{\mathcal{T}}$ is a $k$-tree;

(H2) $H_{\mathcal{T}}[B_t]$ is isomorphic to $K_{k+1}$ for every $t \in V(\mathcal{T})$.

(H3) $G \subseteq H_{\mathcal{T}}$.

If $|V(\mathcal{T})| = 1$, then $H_{\mathcal{T}}$ is $K_{k+1}$, and we are done.

Otherwise, choose a leaf $t$ and let $t'$ be its parent in $\mathcal{T}$. Therefore, $B_t \setminus B_{t'} = \{v\}$ for some $v \in V(K_n)$. Then $\mathcal{T}' := (\mathcal{T} \setminus \{t\}, (B_t)_{t \in V(\mathcal{T}\setminus\{t\})})$ is a smooth tree decomposition of the graph $G \setminus \{v\}$.

By (H2) of the induction hypothesis, $H_{\mathcal{T}'}[B_t \cap B_{t'}]$ is isomorphic to $K_k$. Then from $H_{\mathcal{T}'}$, we obtain $H_{\mathcal{T}}$ by adding the vertex $v$ and the edges $\{v, u\}$ for every $u \in B_t \cap B_{t'}$. 
Let $\mathcal{H}$ be a $k$-tree. We show that $\text{tw}(\mathcal{H}) \leq k$ by induction on the construction of $\mathcal{H}$.

If $\mathcal{H}$ is isomorphic to $K_{k+1}$, i.e., (K1), then we are done.

Otherwise by (K2) let $v \in V(\mathcal{H})$ satisfy that $\mathcal{H} \setminus \{v\}$ is a $k$-tree and $\mathcal{N}^{G}[v]$ is isomorphic to $K_{k+1}$.

By induction hypothesis, there is a tree decomposition $(T, (B_t)_{t \in V(T)})$ of $\mathcal{H} \setminus \{v\}$ of width $k$. As

$$N^\mathcal{H}(v) := \{ u \in V(\mathcal{H}) \mid \{u, v\} \in E(\mathcal{H}) \}$$

is a clique in $\mathcal{H} \setminus \{v\}$, by Helly property, there is a $B_t$ with $N^\mathcal{H}(v) \subseteq B_t$.

We add a new leaf $t_0$ adjacent to $t$ and set $B_{t_0} := N^\mathcal{H}[v]$. 
Graph Isomorphism Problems
Definition

Let $G$ and $H$ be two graphs. A function $f : V(G) \rightarrow V(H)$ is an isomorphism if:

1. $f$ is a bijection;
2. For every $u, v \in V(G)$ we have $\{u, v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(H)$.

If such an $f$ exists, then $G$ and $H$ are isomorphic.
Graph Isomorphism (GI) problem

Input: Two graphs $G$ and $H$.
Problem: Decides whether $G$ and $H$ are isomorphic.

Remark
1. GI is in NP.
2. GI is not NP-complete, unless Polynomial Hierarchy collapses (which most people do not believe).
3. We don’t know whether GI is P-hard.
4. Some people believe GI is in P, but we don’t even have a quantum polynomial time algorithm.
Graph isomorphism problems and treewidth
Theorem (Bodlaender, 1990)

Let $k \in \mathbb{N}$. Then there is a polynomial time algorithm which decides GI on graphs $G$ with $\text{tw}(G) \leq k$. 
I will present an algorithm deciding the problem

<table>
<thead>
<tr>
<th>Input:</th>
<th>Two graphs $G$ and $H$ and a smooth tree decomposition of $G$ of width $k$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem:</td>
<td>Decides whether $G$ and $H$ are isomorphic.</td>
</tr>
</tbody>
</table>

in time

$$(|V(G)| + |V(H)|)^{O(k)}.$$
Isomorphism via connected components

Let $\mathcal{C}_G$ be the set of connected components of $\mathcal{G}$ and $\mathcal{C}_H$ the set of connected components of $\mathcal{H}$.

Then $\mathcal{G}$ and $\mathcal{H}$ are isomorphic if and only if there is a bijection $h : \mathcal{C}_G \rightarrow \mathcal{C}_H$ such that $\mathcal{G}[C]$ and $\mathcal{H}[h(C)]$ are isomorphic for every $C \in \mathcal{C}_G$.

This is equivalent to that there is a perfect matching in the following bipartite graph.

1. The left part is $\mathcal{C}_G$ and the right part $\mathcal{C}_H$.

2. There is an edge between a $C \in \mathcal{C}_G$ and a $C' \in \mathcal{C}_H$ if $\mathcal{G}[C]$ and $\mathcal{H}[C']$ are isomorphic.
Isomorphism via separators

Let $S \subseteq V(G)$ and

$$C_{G \setminus S} := \{ C \mid C \text{ a connected component of } G \setminus S \}.$$ 

Then $G$ and $H$ are isomorphic if and only if there is a set $S' \subseteq V(H)$, a function $h : C_{G \setminus S} \to C_{H \setminus S'}$ and functions $f_C : S \cup C \to S' \cup h(C)$ for all $C \in C_{G \setminus S}$ such that

1. $|S| = |S'|$;
2. $h$ is a bijection;
3. $f_C$ is an isomorphism between $G[S \cup C]$ and $H[S' \cup h(C)]$ for every $C \in C_{G \setminus S}$, and $f_C(S) = S'$;
4. $f_{C_1} \upharpoonright S = f_{C_2} \upharpoonright S$ for every $C_1, C_2 \in C_{G \setminus S}$. 

Let \((T, (B_t)_{t \in V(T)})\) be a smooth tree decomposition of width \(k\) for the graph \(G\). Again we choose an arbitrary root \(r\) in \(T\).

For every \(t \in V(T)\) we define

\[\mathcal{C}_t := \{ C \mid C = \emptyset \text{ or } C \text{ a connected component of } G_{\leq t} \setminus B_t \}.\]
Lemma

Every nonempty $C \in \mathcal{C}_t$, i.e., a connect component in $G_{\leq t} \setminus B_t$, is a connected component of $G \setminus B_t$.

Proof.

Clearly there is a connected component $C'$ in $G \setminus B_t$ with $C \subseteq C'$.

Assume that $C' \setminus C \neq \emptyset$. Then there is an edge $\{u, v\} \in E(G)$ with $u \in V(G_{\leq t}) \setminus B_t$ and $v \in V(G) \setminus V(G_{\leq t})$.

But then, $\{u, v\}$ is not contained in any bag of the tree decomposition. \qed
Lemma

Let \( t_1 \) be a child of \( t \). Then for every nonempty \( C_1 \in \mathcal{C}_{t_1} \) there is a unique \( C \in \mathcal{C}_t \) with \( C_1 \subseteq C \), and \( C_1 \cap C' = \emptyset \) for all other \( C' \in \mathcal{C}_t \).

Proof.

Let \( C_1 \) be a connected component of \( G_{\leq t_1} \setminus B_{t_1} \).

Observe that

\[
G_{\leq t_1} \setminus B_{t_1} \subseteq G_{\leq t} \setminus B_t,
\]

so \( C_1 \) is connected in \( G_{\leq t} \setminus B_t \), and the result follows.
Lemma

Let $t$ be a node in $\mathcal{T}$ with children $t_1, \ldots, t_n$. And let $C \in \mathcal{C}_t$ be nonempty. Then, there is a unique $i \in [m]$ such that

$$C \subseteq \bigcup_{i} \mathcal{C}_{t_i} \cup \{v\} \quad \text{where} \quad \{v\} = B_{t_i} \setminus B_t.$$

Intuitively, $C$ is shattered, i.e., broken into several smaller connected components, by the bag of exactly one child of $t$. 
Connected components via tree decompositions (4)

Lemma
Let $t_1, t_2$ be two distinct children of $t$. For every $i \in [2]$, let $v_i$ be the vertex in $G$ with $\{v_i\} = B_{t_i} \setminus B_t$; and $C_i \in \mathcal{C}_{t_i}$. Then for every $C \in \mathcal{C}_t$

$$(C_1 \cup \{v_1\}) \cap C = \emptyset \quad \text{or} \quad (C_2 \cup \{v_2\}) \cap C = \emptyset.$$
Proof.
It is easy to see
\[(C_1 \cup \{v_1\}) \cap (C_2 \cup \{v_2\}) = \emptyset.\]
Assume \((C_1 \cup \{v_1\}) \cap C \neq \emptyset \neq (C_2 \cup \{v_2\}) \cap C\). Then there is a path \(P\) from \(C_1 \cup \{v_1\}\) to \(C_2 \cup \{v_2\}\) in \(C\). Without loss of generality, we can assume that all vertices on \(P\) are in \((C_1 \cup \{v_1\}) \cup (C_2 \cup \{v_2\})\).

Then there is an edge between \(C_1 \cup \{v_1\}\) and \(C_2 \cup \{v_2\}\), which cannot be contained in any bag of the tree decomposition. \(\square\)
Let $\mathcal{H}$ be a second graph for which we want to decide whether $\mathcal{G}$ and $\mathcal{H}$ are isomorphic.

We define (the set of pairs of separators and connected components)

\[ \mathcal{SC}(\mathcal{H}) := \{(S, C) \mid S \subseteq V(\mathcal{H}) \text{ with } |S| = k + 1 \]
\[ \text{and } (C = \emptyset \text{ or } C \text{ a connected component of } \mathcal{H} \setminus S)\} \]
Partial isomorphisms

Definition
Let \( t \in V(T) \), \( S_1 := B_t \), and \( C_1 \in C_t \). Moreover, let \( (S_2, C_2) \in SC(H) \). We say \( (S_1, C_1) \) and \( (S_2, C_2) \) are \( f \)-isomorphic for a function \( f : S_1 \to S_2 \), denoted by \( (S_1, C_1) \equiv^f (S_2, C_2) \), if there is a function \( F : S_1 \cup C_1 \to S_2 \cup C_2 \) such that

(F1) \( F \upharpoonright S_1 = f \);

(F2) for every \( u, v \in S_1 \cup C_1 \) we have \( \{u, v\} \in E(G) \) if and only if \( \{F(u), F(v)\} \in E(H) \).

That is, \( F \) is an isomorphism between \( G[S_1 \cup C_1] \) and \( H[S_2 \cup C_2] \) extending \( f \).
Extending partial isomorphisms

Our goal is to compute for each $t \in V(T)$ the set

$$\mathcal{F}_t := \left\{ (f, B_t, C_1, S_2, C_2) \mid (B_t, C_1) \equiv^f (S_2, C_2) \right\}.$$

where $C_1 \in C_t$ and $(S_2, C_2) \in SC(H)$

using dynamic programming.
Leaves

Let $t$ be a leaf of $\mathcal{T}$. Then $\mathcal{C}_t = \{\emptyset\}$. Hence,

$$\mathcal{F}_t := \{(f, B_t, \emptyset, S_2, \emptyset) \mid (B_t, \emptyset) \equiv^f (S, \emptyset) \}
\text{where } S_2 \subseteq V(\mathcal{H}) \text{ with } |S_2| = k + 1.$$

This can be computed in time

$$(k + 1)! \cdot |V(\mathcal{H})|^{O(k)}.$$
Let $t$ be a node in $\mathcal{T}$ with children $t_1, \ldots, t_m$ for some $m \geq 1$.

Now let $C_1 \in \mathcal{C}_t$ be nonempty. By Lemma 3, there is a unique $i \in [m]$ such that

$$C_1 \subseteq \bigcup \mathcal{C}_{t_i} \cup \{v\} \quad \text{where } \{v\} = B_{t_i} \setminus B_t.$$

For every $(S_2, C_2) \in \mathcal{LCH}(\mathcal{H})$ and every $f : B_t \to S_2$ with $(B_t, \emptyset) \equiv^f (S_2, \emptyset)$ we want to check whether $(B_t, C_1) \equiv^f (S_2, C_2)$. 
Non-leaves (2)

\((B_t, C_1) \equiv^f (S_2, C_2)\) if and only if some for \(v' \in V(H) \setminus S_2, u \in S_2\), and

- \(S'_2 := S_2 \cup \{v'\} \setminus \{u\}\),
- \(C^*_1 := \{C^* \mid C^* \text{ a connected component of } G \setminus B_{t_i} \text{ with } C^* \subseteq C_1\}\)
  \(\) and
- \(C^*_2 := \{C^* \mid C^* \text{ a connected component of } H \setminus S'_2 \text{ with } C^* \subseteq C_2\}\),
- \(f' : B_{t_i} \to S'_2\) defined by
  
  \[f'(w) = \begin{cases} 
  v' & \text{if } w = v \\
  f(w) & \text{otherwise,} 
  \end{cases}\]

we have

\((\text{N1})\) every connected component of \(H \setminus S'_2\) is either contained in or disjoint with \(C_2\);
\((\text{N2})\) \(C_2 \subseteq \bigcup C^*_2 \cup \{v'\}\);
\((\text{N3})\) there is a bijection \(h : C^*_1 \to C^*_2\) such that for every \(C^* \in C^*_1\)

\[(B_{t_i}, C^*) \equiv^{f'} (S'_2, h(C^*)).\]
(N1) and (N2) can be checked in polynomial time.

To verify (N3) we create a bipartite graph \( B \):

1. the left part is \( C_1^* \) and the right part \( C_2^* \);
2. there is an edge between \( C_1^* \in C_1^* \) and \( C_2^* \in C_2^* \) if \((B_{t_i}, C_1^*) \equiv f'(S'_2, C_2^*)\).

Then (N3) holds if and only if there is a perfect matching in \( B \), which can be decided in polynomial time.
The final step

\( \mathcal{G} \) and \( \mathcal{H} \) are isomorphic if and only if for some \( S_2 \subseteq V(\mathcal{H}) \) with \( |S_2| = k + 1 \) and \( f : B_r \to S_2 \) there is a perfect matching in the following bipartite graph.

1. The left part is \( \mathcal{C}_r \) and the right part \( \mathcal{C}^* := \{ C_2 \mid (S_2, C_2) \in SC(\mathcal{H}) \} \).
2. There is an edge between a \( C_1 \in \mathcal{C}_r \) and a \( C_2 \in \mathcal{C}^* \) if \( (B_r, C_1) \equiv^f (S_2, C^*) \).