On the Hardness of Computing the Edit Distance of Shallow Trees

Panagiotis Charalampopoulos[1], Paweł Gawrychowski[2], Shay Mozes[3], Oren Weimann[4]

1. Birkbeck, University of London, UK
2. University of Wrocław, Poland
3. Reichman University, Herzliya, Israel
4. University of Haifa, Israel

SPIRE 2022

Concepción, Chile

Edit Distance of Trees

Edit Distance of Trees

Input: Two ordered vertex-labelled rooted trees F and G and a cost function.

Edit Distance of Trees

Input: Two ordered vertex-labelled rooted trees F and G and a cost function.

Output: The minimum cost of transforming F into G by a sequence of elementary **edit operations**:

Edit Distance of Trees

Input: Two ordered vertex-labelled rooted trees F and G and a cost function.

Output: The minimum cost of transforming F into G by a sequence of elementary **edit operations**:

- changing the label of a node v,

Edit Distance of Trees

Input: Two ordered vertex-labelled rooted trees F and G and a cost function.

Output: The minimum cost of transforming F into G by a sequence of elementary **edit operations**:

- changing the label of a node v,
- deleting a node v and setting the children of v as the children of v's parent (in the place of v in the left-to-right order),

Edit Distance of Trees

Input: Two ordered vertex-labelled rooted trees F and G and a cost function.

Output: The minimum cost of transforming F into G by a sequence of elementary **edit operations**:

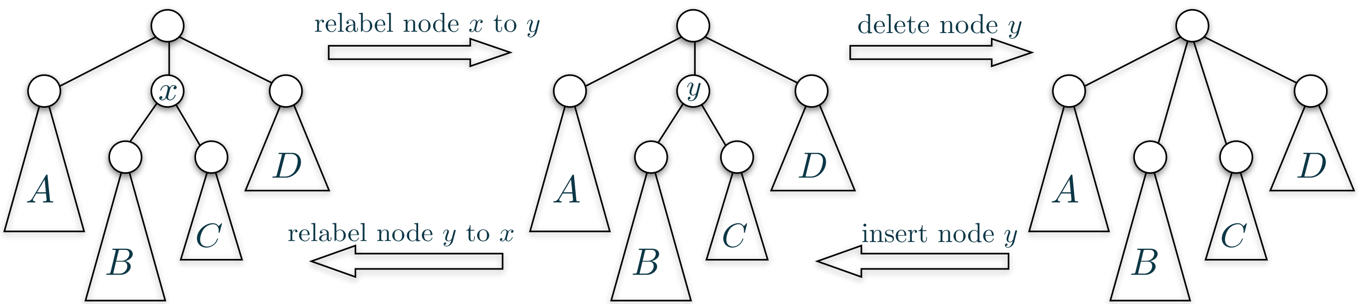
- changing the label of a node v,
- deleting a node v and setting the children of v as the children of v's parent (in the place of v in the left-to-right order),
- inserting a node v (defined as the inverse of a deletion).

Edit Distance of Trees

Input: Two ordered vertex-labelled rooted trees F and G and a cost function.

Output: The minimum cost of transforming F into G by a sequence of elementary **edit operations**:

- changing the label of a node v,
- deleting a node v and setting the children of v as the children of v's parent (in the place of v in the left-to-right order),
- inserting a node v (defined as the inverse of a deletion).

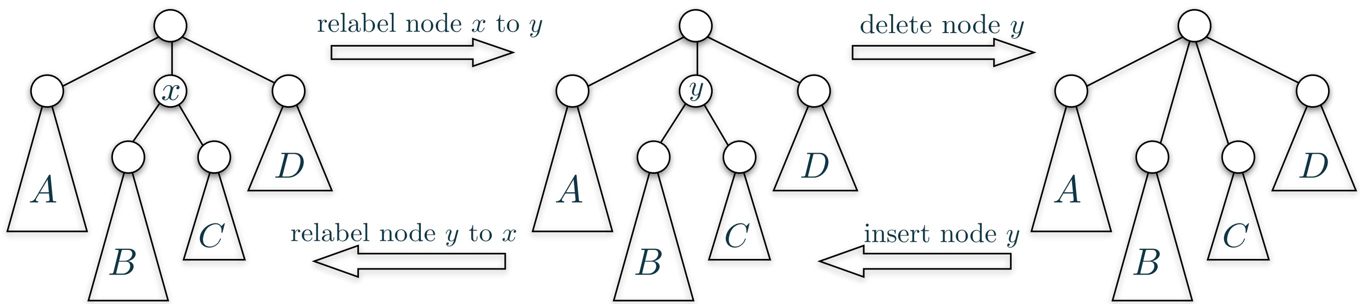


Edit Distance of Trees

Input: Two ordered vertex-labelled rooted trees F and G and a cost function.

Output: The minimum cost of transforming F into G by a sequence of elementary **edit operations**:

- changing the label of a node v,
- deleting a node v and setting the children of v as the children of v's parent (in the place of v in the left-to-right order),
- inserting a node v (defined as the inverse of a deletion).



Remark: The cost function may require space quadratic in the size of the trees.

History

History

$\mathcal{O}(n^6)$	[Tai; JACM 1979]
$\mathcal{O}(n^4)$	[Zhang, Shasha; SICOMP 1989]
$\mathcal{O}(n^3 \log n)$	[Klein; ESA 1998]
$\mathcal{O}(n^3)$	[Demaine, Mozes, Rossman, Weimann; TALG 2009]

History

$\mathcal{O}(n^6)$ [Tai; JACM 1979]

$\mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3 \log n)$ [Klein; ESA 1998]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

History

$\mathcal{O}(n^6)$ [Tai; JACM 1979]

$\mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3 \log n)$ [Klein; ESA 1998]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

Fact: Given two forests F and G, the rightmost (or leftmost) roots of F and G are either matched or (at least) one of them is deleted.

History

$\mathcal{O}(n^6)$ [Tai; JACM 1979]

$\mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3 \log n)$ [Klein; ESA 1998]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

Fact: Given two forests F and G, the rightmost (or leftmost) roots of F and G are either matched or (at least) one of them is deleted.

Dynamic Programming: consider all three such options and recurse.

History

$\mathcal{O}(n^6)$ [Tai; JACM 1979]

$\mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3 \log n)$ [Klein; ESA 1998]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

dec. algs $\begin{cases} \Omega(n^2 \log^2 n) & \text{[Dulucq, Touzet; JDA 2005]} \\ \Omega(n^3) & \text{[Demaine, Mozes, Rossman, Weimann; TALG 2009]} \end{cases}$

History

$\mathcal{O}(n^6)$	[Tai; JACM 1979]
$\mathcal{O}(n^4)$	[Zhang, Shasha; SICOMP 1989]
$\mathcal{O}(n^3 \log n)$	[Klein; ESA 1998]
$\mathcal{O}(n^3)$	[Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

dec. algs $\left\{ \begin{array}{ll} \Omega(n^2 \log^2 n) & \text{[Dulucq, Touzet; JDA 2005]} \\ \Omega(n^3) & \text{[Demaine, Mozes, Rossman, Weimann; TALG 2009]} \end{array} \right.$

No $\mathcal{O}(n^{3-\epsilon})$ [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]
under the APSP hypothesis or the stronger k-Clique hypothesis.

History

$\mathcal{O}(n^6)$ [Tai; JACM 1979]

$\mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3 \log n)$ [Klein; ESA 1998]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

$$\text{dec. algs} \begin{cases} \Omega(n^2 \log^2 n) & \text{[Dulucq, Touzet; JDA 2005]} \\ \Omega(n^3) & \text{[Demaine, Mozes, Rossman, Weimann; TALG 2009]} \end{cases}$$

No $\mathcal{O}(n^{3-\epsilon})$ [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]
under the APSP hypothesis or the stronger k-Clique hypothesis.

Question: What if the depths of the trees are bounded by some parameter d?

History

$\mathcal{O}(n^6)$ [Tai; JACM 1979]

$\mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3 \log n)$ [Klein; ESA 1998]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

dec. algs $\left\{ \begin{array}{l} \Omega(n^2 \log^2 n) \\ \Omega(n^3) \end{array} \right.$ [Dulucq, Touzet; JDA 2005] / [Demaine, Mozes, Rossman, Weimann; TALG 2009]

No $\mathcal{O}(n^{3-\epsilon})$ [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020] under the APSP hypothesis or the stronger k-Clique hypothesis.

Question: What if the depths of the trees are bounded by some parameter d?

E.g., when the trees are stars the problem is essentially string edit distance.

History

$\mathcal{O}(n^2 d^4) = \mathcal{O}(n^6)$ [Tai; JACM 1979]

$\mathcal{O}(n^2 d^2) = \mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3 \log n)$ [Klein; ESA 1998]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

The last three results are based on decomposition algorithms.

dec. algs $\begin{cases} \Omega(n^2 \log^2 n) & \text{[Dulucq, Touzet; JDA 2005]} \\ \Omega(n^3) & \text{[Demaine, Mozes, Rossman, Weimann; TALG 2009]} \end{cases}$

No $\mathcal{O}(n^{3-\epsilon})$ [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]
under the APSP hypothesis or the stronger k-Clique hypothesis.

Question: What if the depths of the trees are bounded by some parameter d?

E.g., when the trees are stars the problem is essentially string edit distance.

History

$\mathcal{O}(n^2 d^2) = \mathcal{O}(n^4)$ [Zhang, Shasha; SICOMP 1989]

$\mathcal{O}(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

dec. algs $\Big\{$ $\Omega(n^3)$ [Demaine, Mozes, Rossman, Weimann; TALG 2009]

No $\mathcal{O}(n^{3-\epsilon})$ [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]
under the APSP hypothesis or the stronger k-Clique hypothesis.

Question: What if the depths of the trees are bounded by some parameter d?

E.g., when the trees are stars the problem is essentially string edit distance.

Our Results

Our Results

1. There is no $o(n^2 d^2)$-time decomposition algorithm.

Our Results

1. There is no $o(n^2 d^2)$-time decomposition algorithm.

2. There is no $\mathcal{O}(n^2 d^{1-\epsilon})$-time algorithm for any constant $\epsilon > 0$ when $d = \mathrm{poly}(n)$ under the APSP hypothesis.

Our Results

1. There is no $o(n^2 d^2)$-time decomposition algorithm.

2. There is no $\mathcal{O}(n^2 d^{1-\epsilon})$-time algorithm for any constant $\epsilon > 0$ when $d = \mathrm{poly}(n)$ under the APSP hypothesis.

APSP hypothesis: Computing all-pairs shortest paths in an n-vertex graph with polynomial edge-weights cannot be done in time $\mathcal{O}(n^{3-\epsilon})$.

Our Results

1. There is no $o(n^2 d^2)$-time decomposition algorithm.

2. There is no $\mathcal{O}(n^2 d^{1-\epsilon})$-time algorithm for any constant $\epsilon > 0$ when $d = \mathrm{poly}(n)$ under the APSP hypothesis.

APSP hypothesis: Computing all-pairs shortest paths in an n-vertex graph with polynomial edge-weights cannot be done in time $\mathcal{O}(n^{3-\epsilon})$.

Instead of reducing APSP to TED, we reduce from the equivalent NegativeTriangle problem [Vassilevska Williams, Williams; JACM 2018]:

Our Results

1. There is no $o(n^2d^2)$-time decomposition algorithm.

2. There is no $\mathcal{O}(n^2d^{1-\epsilon})$-time algorithm for any constant $\epsilon > 0$ when $d = \mathrm{poly}(n)$ under the APSP hypothesis.

APSP hypothesis: Computing all-pairs shortest paths in an n-vertex graph with polynomial edge-weights cannot be done in time $\mathcal{O}(n^{3-\epsilon})$.

Instead of reducing APSP to TED, we reduce from the equivalent NEGATIVETRIANGLE problem [Vassilevska Williams, Williams; JACM 2018]:

NegativeTriangle: Check whether a complete tripartite graph with parts of size at most n and polynomial edge-weights contains a negative triangle, that is, if there exist vertices u, v, z with $w(u, v) + w(v, z) + w(z, u) < 0$.

NegativeTriangle to TED

NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

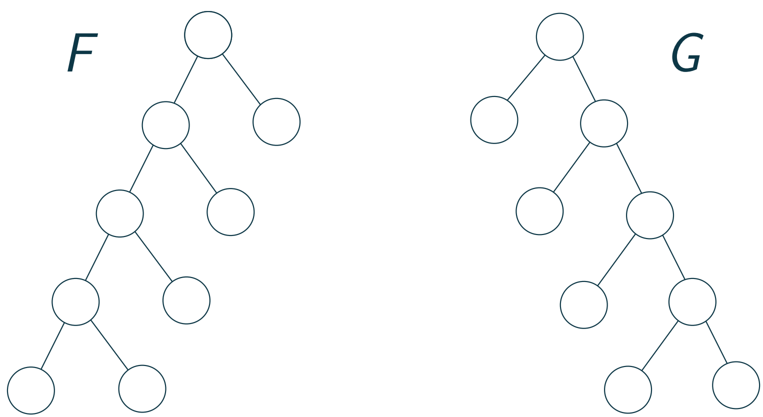
- deleting or inserting any node costs zero;

NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

- deleting or inserting any node costs zero;
- the trees are two opposing combs of depth $2n + 1$;

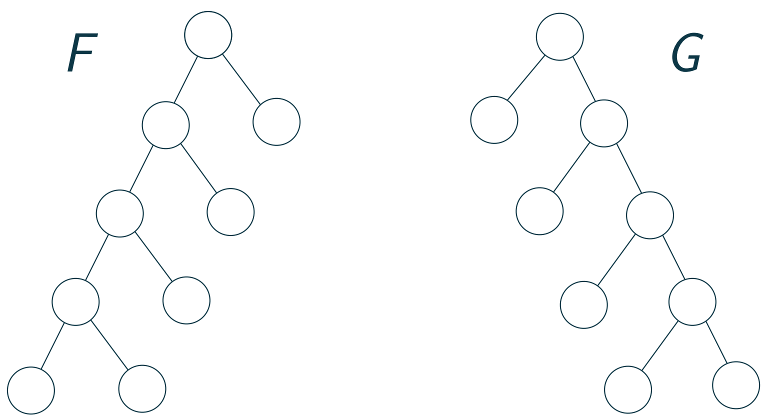


NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

- deleting or inserting any node costs zero;
- the trees are two opposing combs of depth $2n+1$;
- $\mathrm{TED}(F, G) = -3M^2 +$ minimum weight of a triangle, where $M \in \mathbb{N}$.



NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

- deleting or inserting any node costs zero;
- the trees are two opposing combs of depth $2n+1$;
- $\mathrm{TED}(F, G) = -3M^2 +$ minimum weight of a triangle, where $M \in \mathbb{N}$.

For this talk assume that $M = 0$.

NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

- deleting or inserting any node costs zero;
- the trees are two opposing combs of depth $2n+1$;
- $\mathrm{TED}(F, G) = -3M^2 +$ minimum weight of a triangle, where $M \in \mathbb{N}$.

For this talk assume that $M = 0$.

The constructed trees are deep.

NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

- deleting or inserting any node costs zero;
- the trees are two opposing combs of depth $2n+1$;
- $\mathrm{TED}(F, G) = -3M^2+$ minimum weight of a triangle, where $M \in \mathbb{N}$.

For this talk assume that $M = 0$.

The constructed trees are deep.

But, the shapes of the trees do not depend on the graph.

NegativeTriangle to TED

Theorem [Bringmann, Gawrychowski, Mozes, Weimann; TALG 2020]

Computing the minimum weight of a triangle in a complete undirected n-vertex graph reduces in $\mathcal{O}(n^2)$ time to solving an instance of TED with trees of size $\mathcal{O}(n)$ such that:

- deleting or inserting any node costs zero;
- the trees are two opposing combs of depth $2n+1$;
- $\mathrm{TED}(F, G) = -3M^2+$ minimum weight of a triangle, where $M \in \mathbb{N}$.

For this talk assume that $M = 0$.

The constructed trees are deep.

But, the shapes of the trees do not depend on the graph.

The whole game is labelling the nodes and defining the substitution costs.

Strategy

Strategy

NegativeTriangle, with $3n$ vertices

Strategy

$$\textsc{NegativeTriangle}\text{, with } 3n \text{ vertices}$$

$$\downarrow$$

$$\mathcal{O}(n^3/d^3) \text{ instances of } \textsc{NegativeTriangle}\text{, each with } 3d \text{ vertices}$$

Strategy

$$\textsc{NegativeTriangle}\text{, with } 3n \text{ vertices}$$
$$\downarrow$$
$$\mathcal{O}(n^3/d^3) \text{ instances of } \textsc{NegativeTriangle}\text{, each with } 3d \text{ vertices}$$
$$\downarrow$$
$$\mathcal{O}(n^3/d^3) \text{ instances of } \mathrm{TED}\text{, each on a pair of combs of depth } 6d+1$$

Strategy

$\textsc{NegativeTriangle}$, with $3n$ vertices

$$\downarrow$$

$\mathcal{O}(n^3/d^3)$ instances of $\textsc{NegativeTriangle}$, each with $3d$ vertices

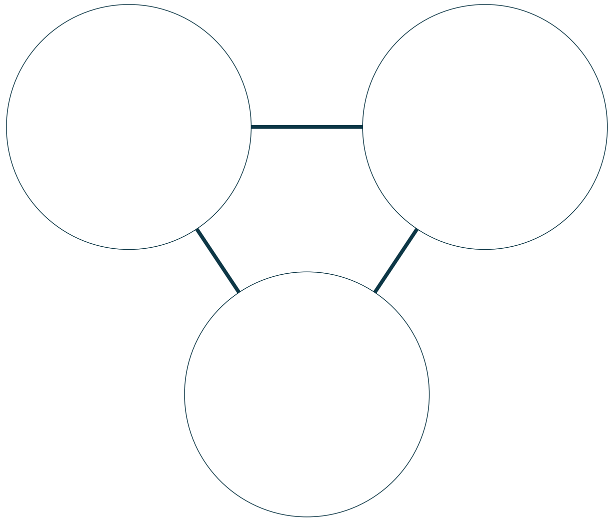
$$\downarrow$$

$\mathcal{O}(n^3/d^3)$ instances of TED, each on a pair of combs of depth $6d + 1$

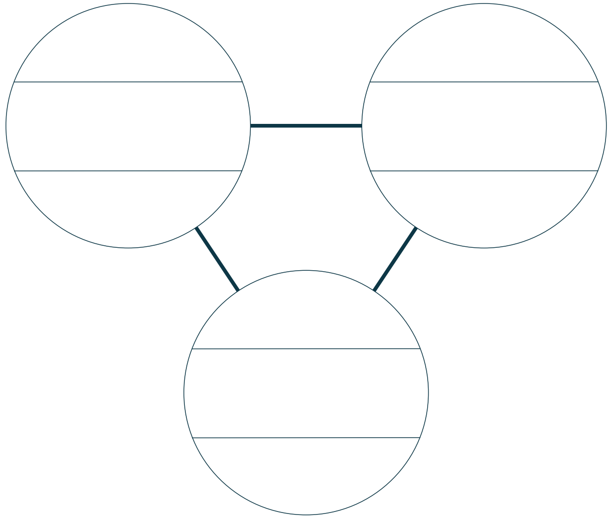
$$\downarrow$$

TED, over $\mathcal{O}(n^{1.5}/\sqrt{d})$-size, $\mathcal{O}(d)$-depth trees

Many Smaller NegativeTriangle Instances

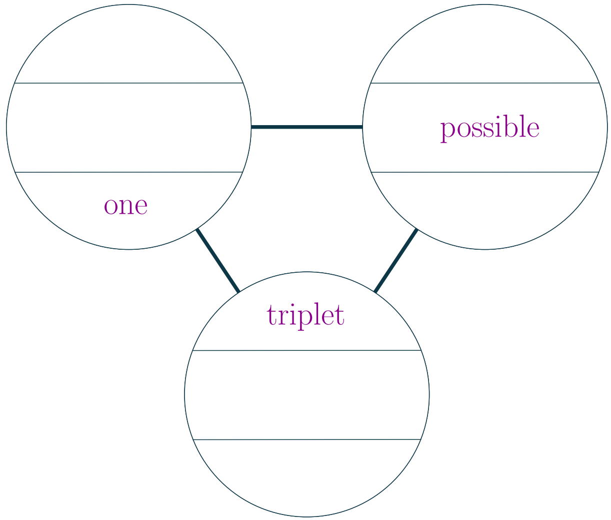


Many Smaller NegativeTriangle Instances



Split each part into $\lceil n/d \rceil$ chunks of size at most d.

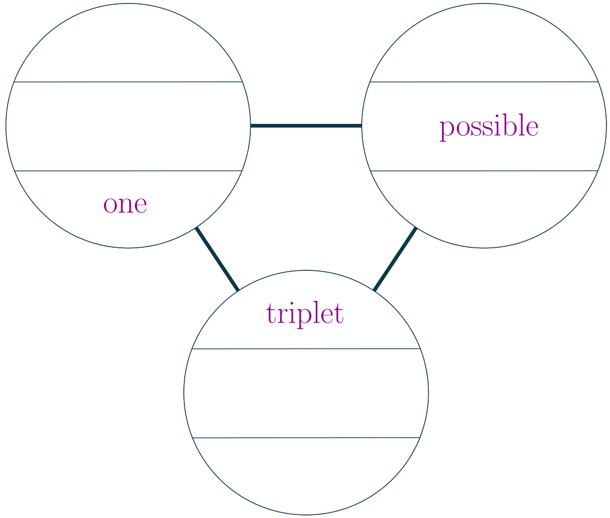
Many Smaller NegativeTriangle Instances



Split each part into $\lceil n/d \rceil$ chunks of size at most d.

$\mathcal{O}(n^3/d^3)$ choices of triplets.

Many Smaller NegativeTriangle Instances

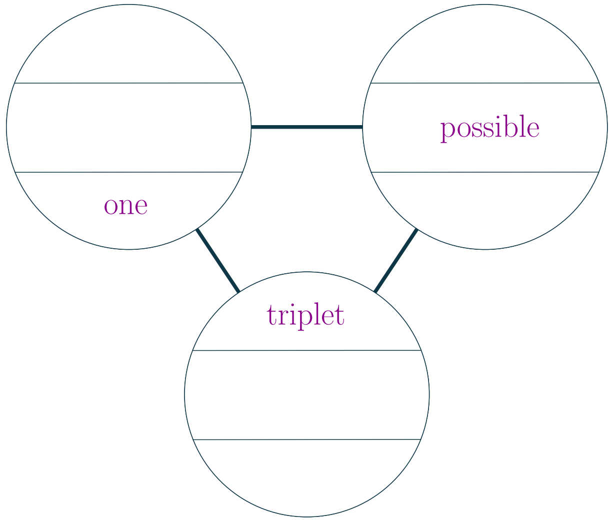


Split each part into $\lceil n/d \rceil$ chunks of size at most d.

$\mathcal{O}(n^3/d^3)$ choices of triplets.

Output size: $\mathcal{O}(d^2 \cdot n^3/d^3) = \mathcal{O}(n^3/d)$.

Many Smaller NegativeTriangle Instances



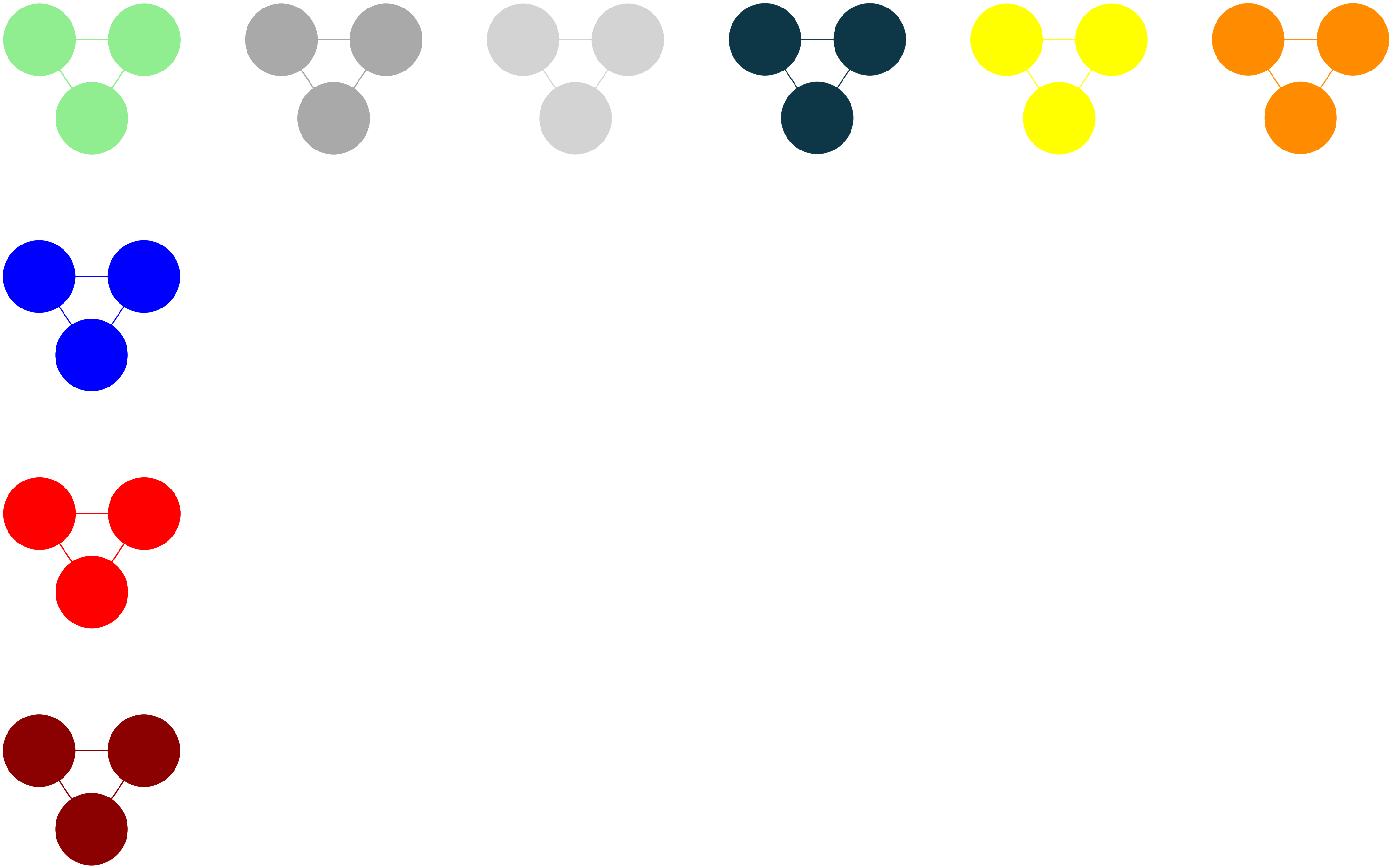
Split each part into $\lceil n/d \rceil$ chunks of size at most d.

$\mathcal{O}(n^3/d^3)$ choices of triplets.

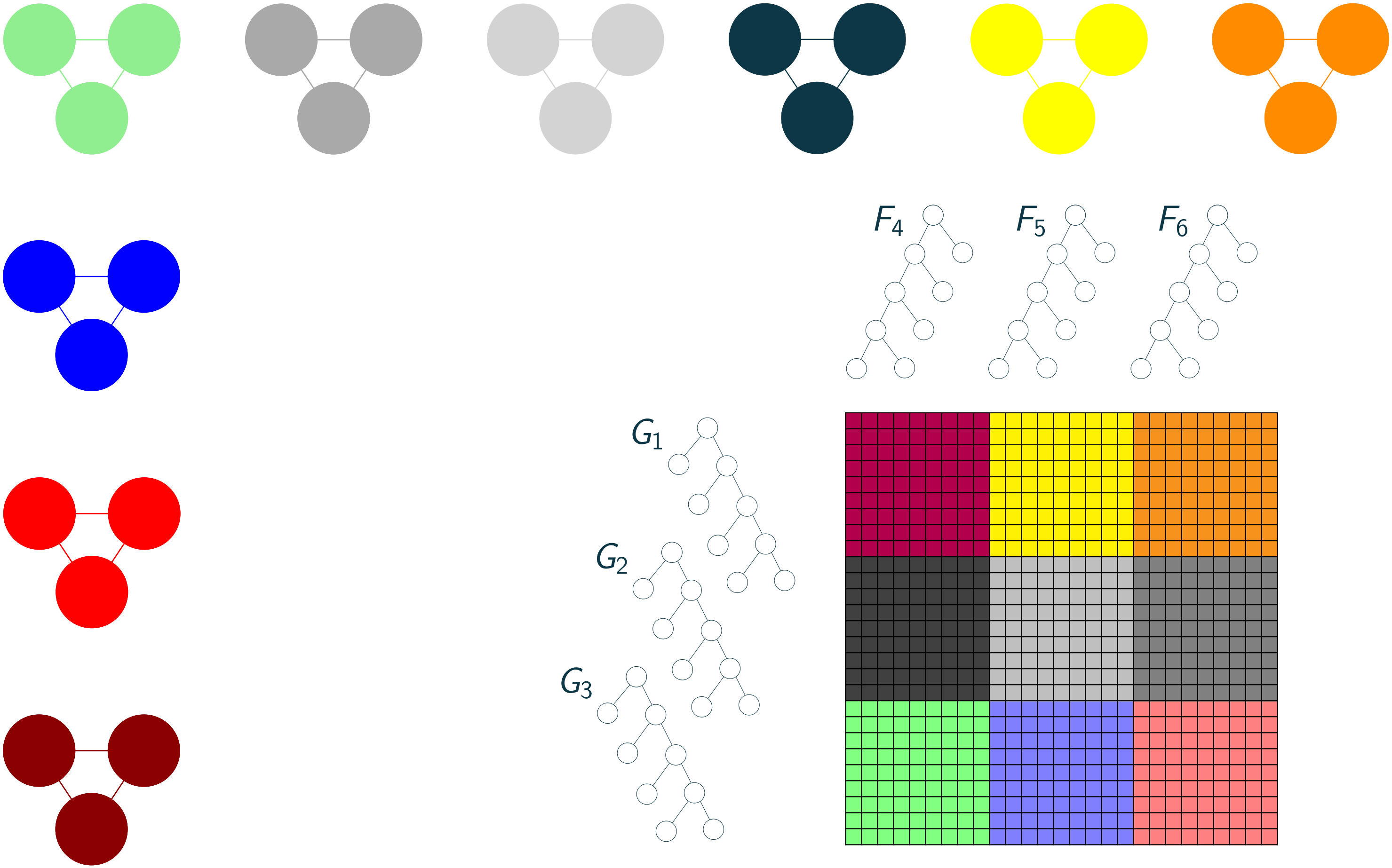
Output size: $\mathcal{O}(d^2 \cdot n^3/d^3) = \mathcal{O}(n^3/d)$.

Time: $\mathcal{O}(n^2 + n^3/d)$. Recall that d is polynomial!

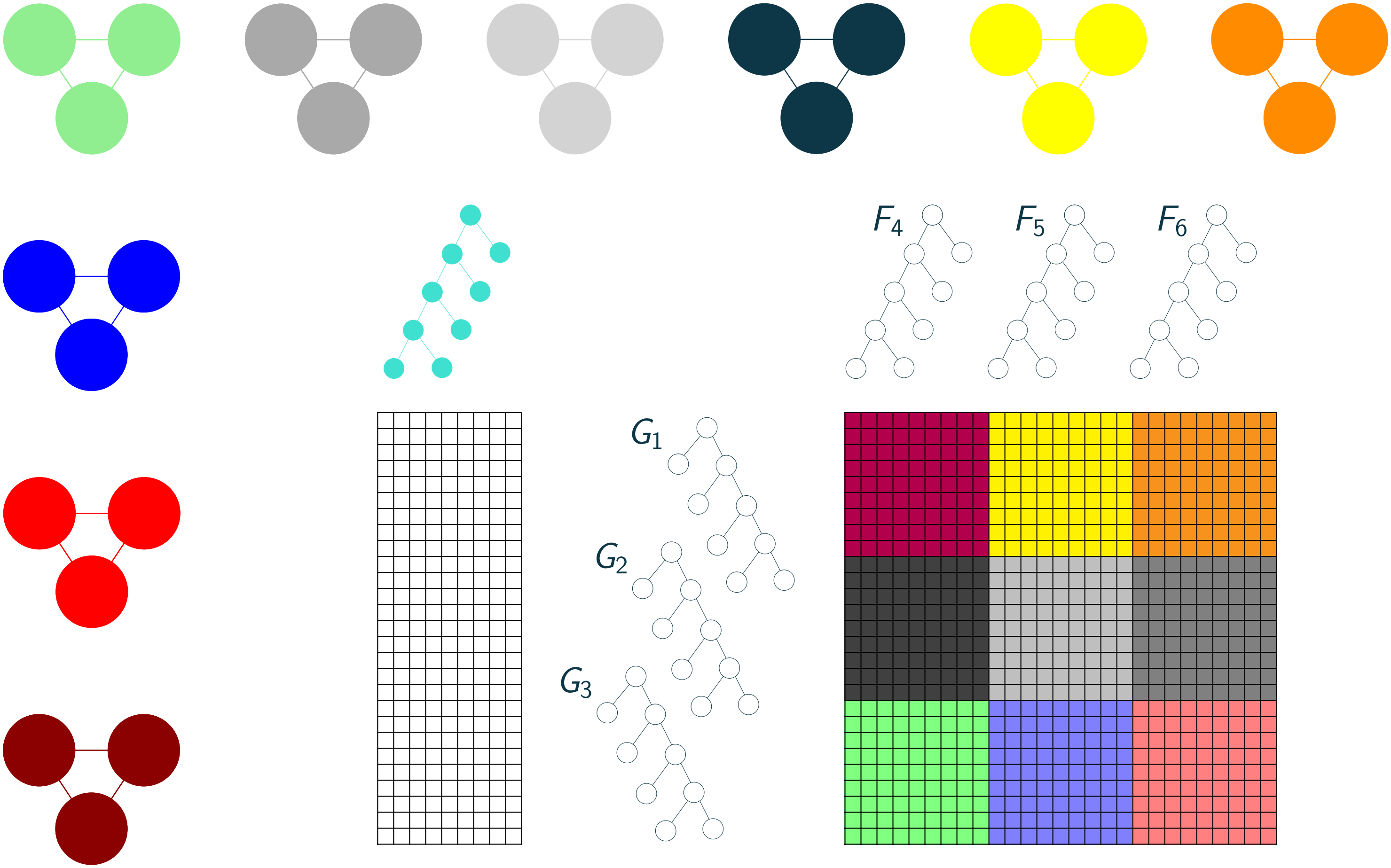
Many TED instances



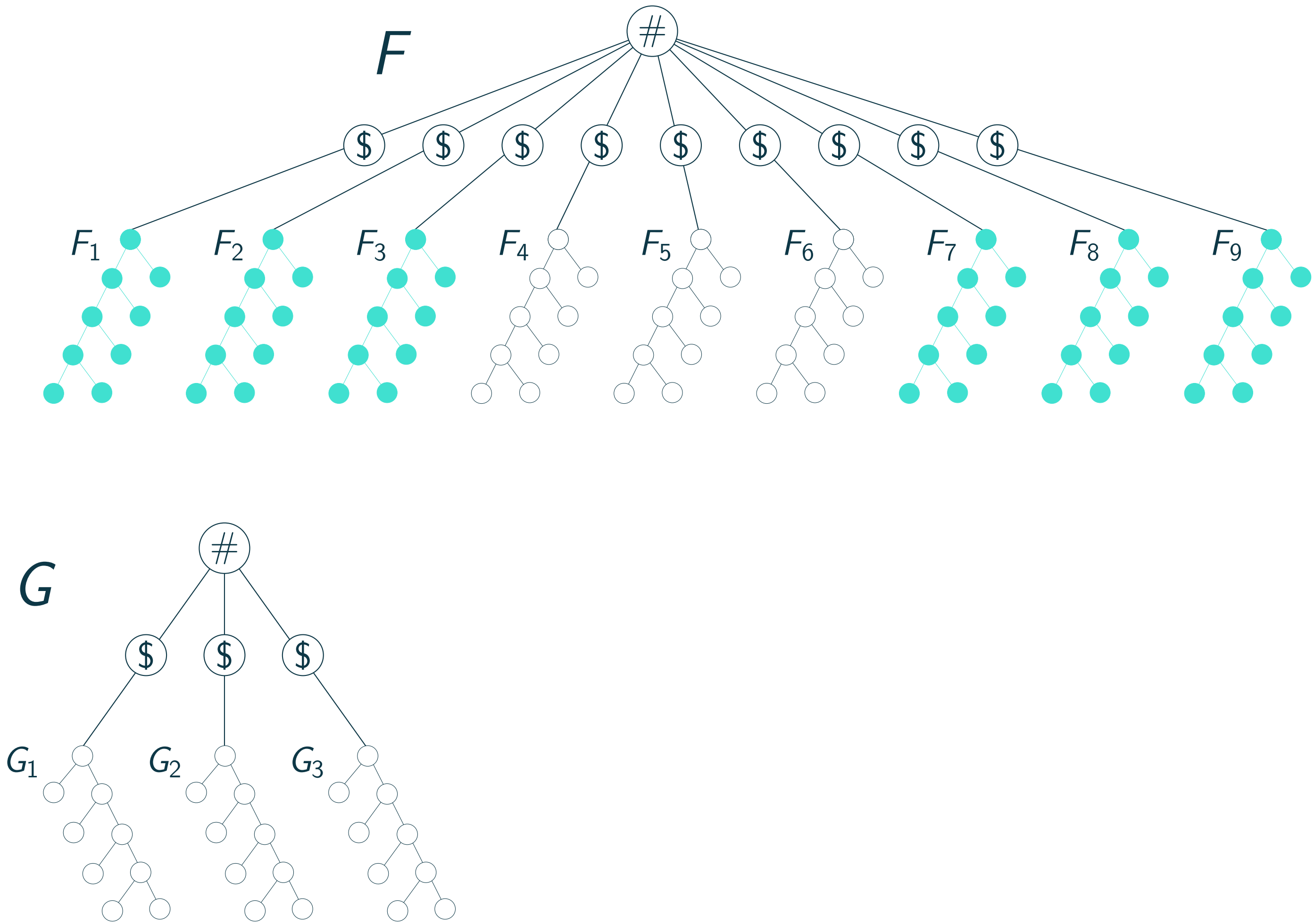
Many TED instances



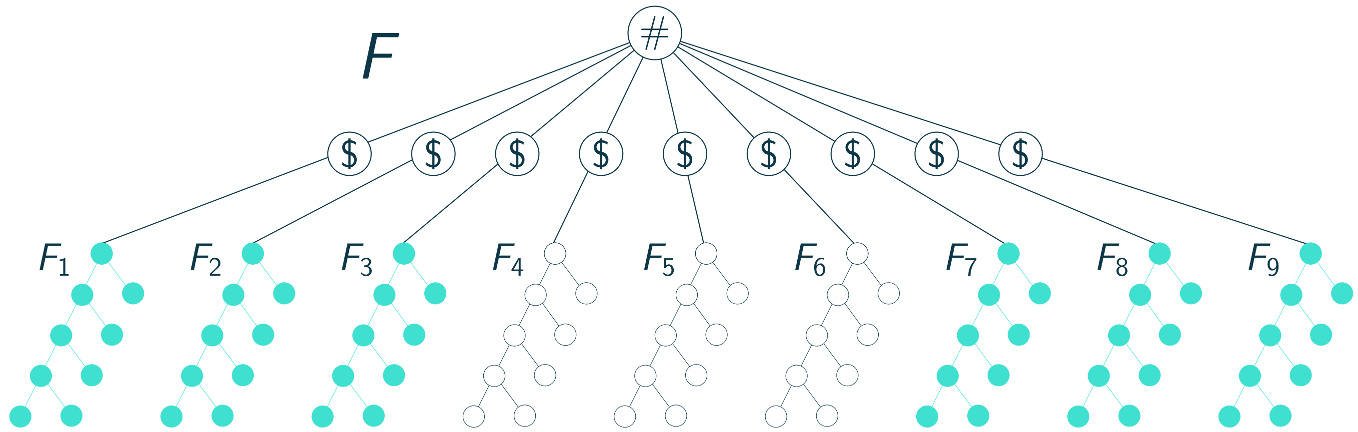
Many TED instances

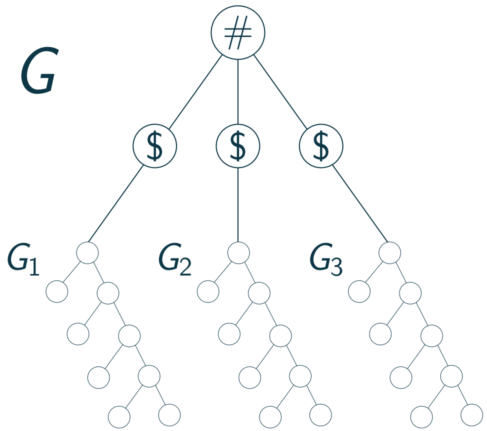


The Final Step



The Final Step

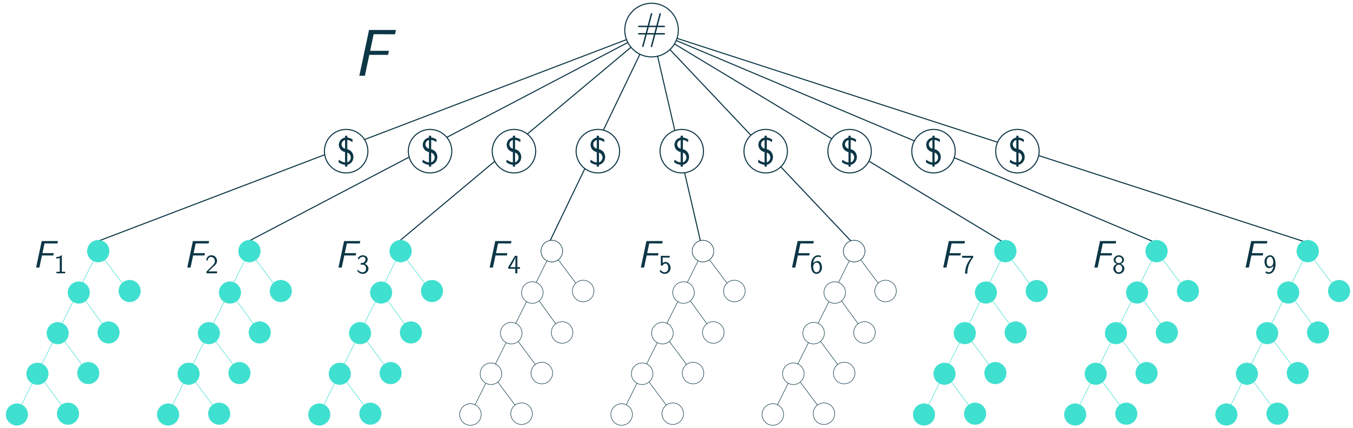


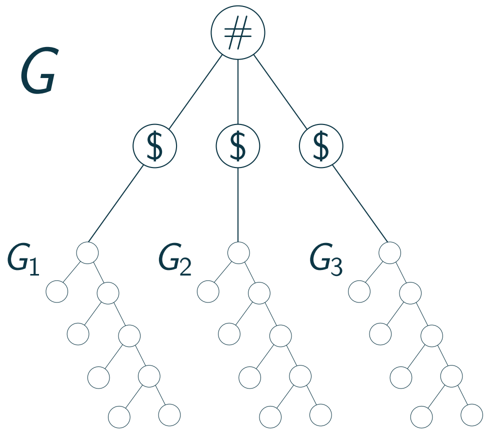


$$c_{match}(\#,\#) = c_{match}(\$,\$) = -\psi, \text{ for huge } \psi$$

$$c_{match}(x,y) = \infty \text{ for } (x,y) \notin \Sigma^2 \cup \{\#\}^2 \cup \{\$\}^2$$

The Final Step



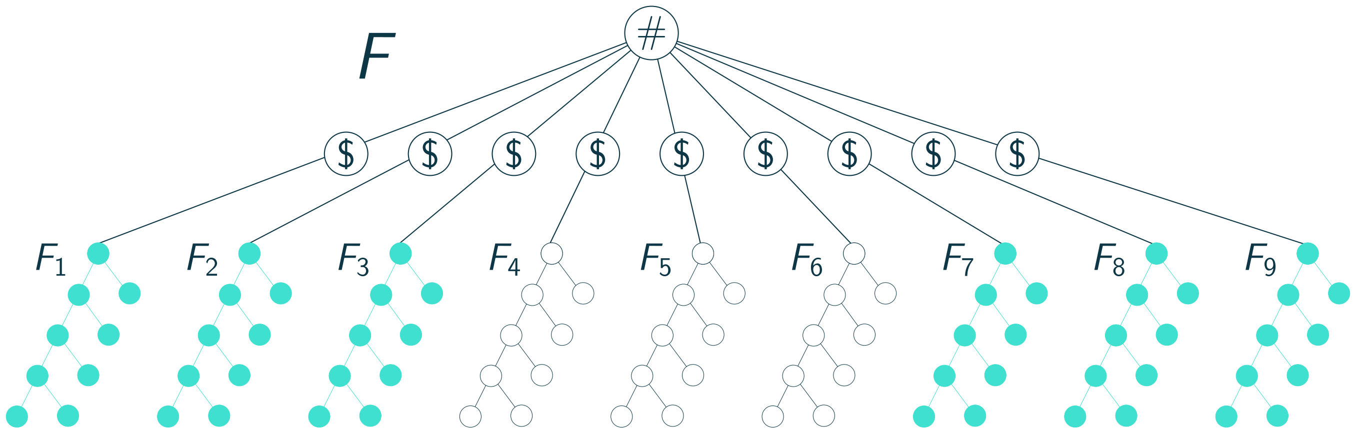


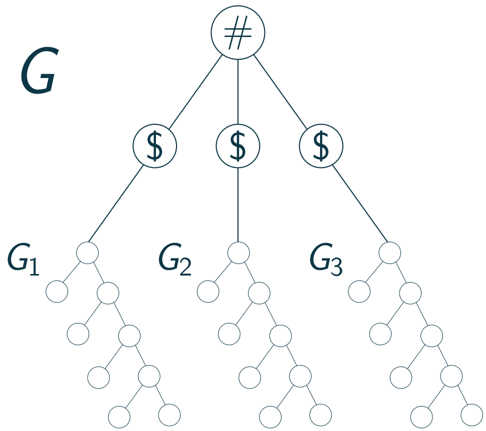
$$c_{match}(\#, \#) = c_{match}(\$, \$) = -\psi, \text{ for huge } \psi$$

$$c_{match}(x, y) = \infty \text{ for } (x, y) \notin \Sigma^2 \cup \{\#\}^2 \cup \{\$\}^2$$

Roots matched, each \$ in G matched with a \$ in F.

The Final Step





$$c_{match}(\#, \#) = c_{match}(\$, \$) = -\psi, \text{ for huge } \psi$$

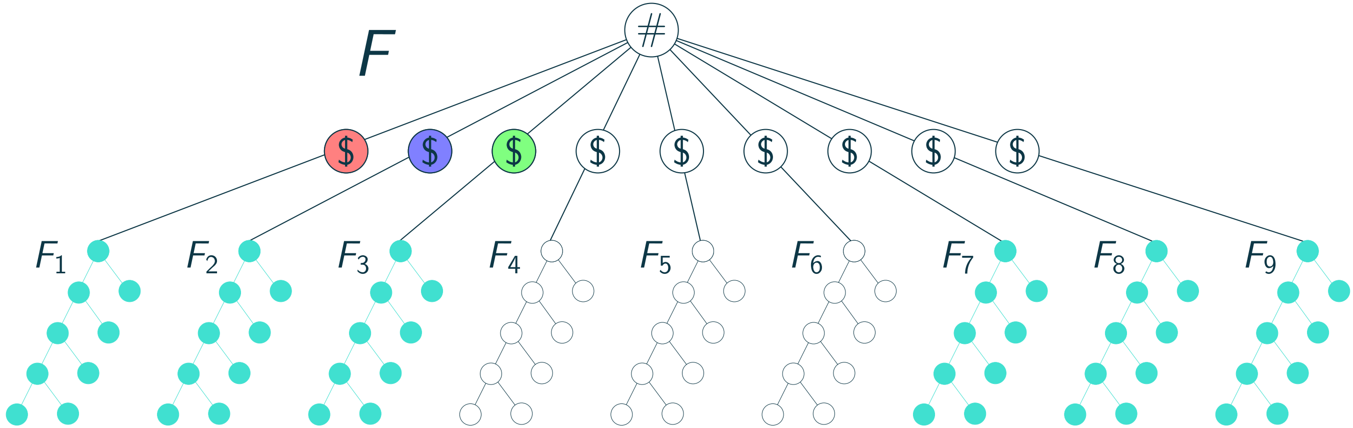
$$c_{match}(x, y) = \infty \text{ for } (x, y) \notin \Sigma^2 \cup \{\#\}^2 \cup \{\$\}^2$$

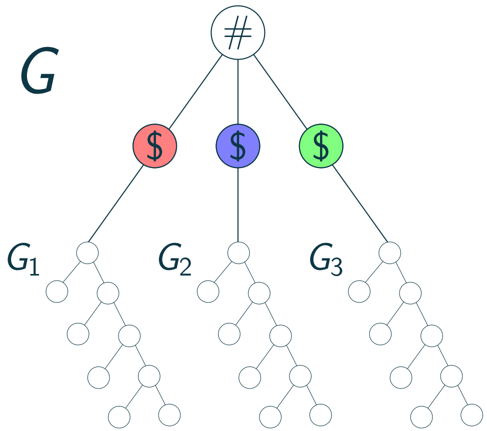
Roots matched, each \$ in G matched with a \$ in F.

$$\mathrm{TED}(F, G) = -4\psi + \min_p \sum_{j=1}^{s} \mathrm{TED}(F_{p(j)}, G_j)$$

over incr. functions $p : \{1, 2, 3\} \to \{1, 2, \ldots, 9\}$.

The Final Step





$c_{match}(\#,\#) = c_{match}(\$,\$) = -\psi$, for huge ψ

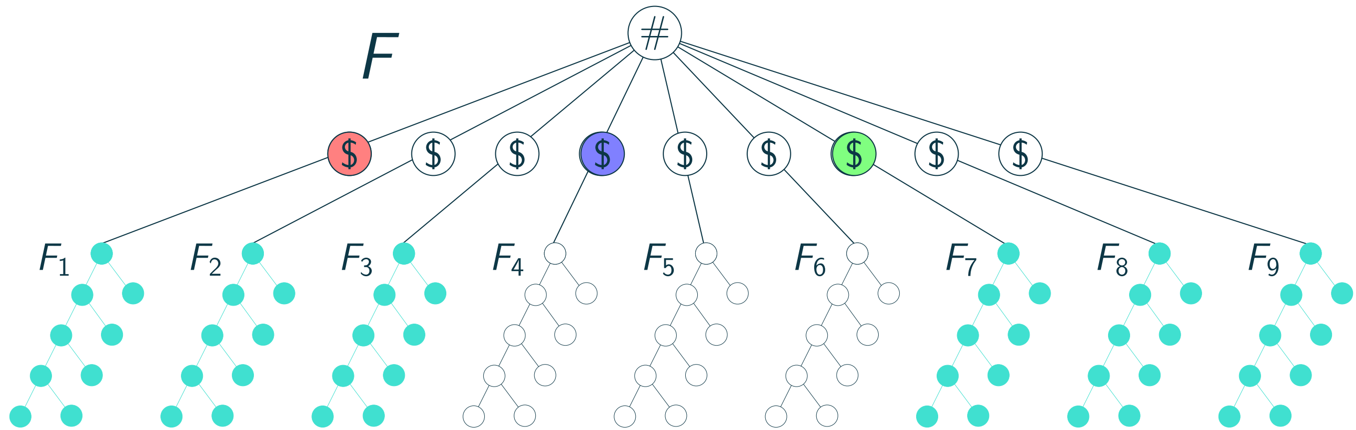
$c_{match}(x,y) = \infty$ for $(x,y) \notin \Sigma^2 \cup \{\#\}^2 \cup \{\$\}^2$

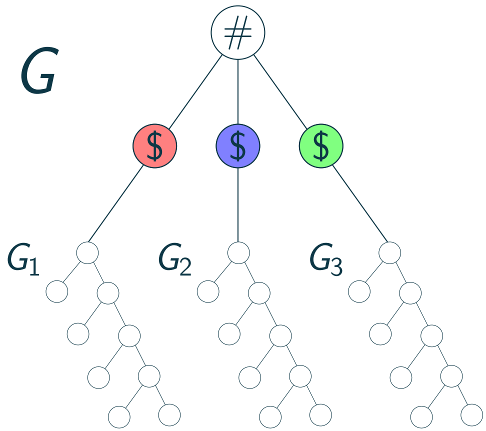
Roots matched, each \$ in G matched with a \$ in F.

$\mathrm{TED}(F,G) = -4\psi + \min_p \sum_{j=1}^{s} \mathrm{TED}(F_{p(j)}, G_j)$

over incr. functions $p : \{1,2,3\} \to \{1,2,\ldots,9\}$.

The Final Step





$c_{match}(\#, \#) = c_{match}(\$, \$) = -\psi$, for huge ψ

$c_{match}(x, y) = \infty$ for $(x, y) \notin \Sigma^2 \cup \{\#\}^2 \cup \{\$\}^2$

Roots matched, each \$ in G matched with a \$ in F.

$\mathrm{TED}(F, G) = -4\psi + \min_p \sum_{j=1}^{s} \mathrm{TED}(F_{p(j)}, G_j)$

over incr. functions $p : \{1, 2, 3\} \to \{1, 2, \ldots, 9\}$.

Wrap-up

Wrap-up

$\textsc{NegativeTriangle}$, with $3n$ vertices

$$\downarrow$$

$\mathcal{O}(n^3/d^3)$ instances of $\textsc{NegativeTriangle}$, each with $3d$ vertices

$$\downarrow$$

$\mathcal{O}(n^3/d^3)$ instances of TED, each on a pair of combs of depth $6d+1$

$$\downarrow$$

TED, over $\mathcal{O}(n^{1.5}/\sqrt{d})$-size, $\mathcal{O}(d)$-depth trees

Wrap-up

$\textsc{NegativeTriangle}$, with $3n$ vertices

$$\Big\downarrow \quad \mathcal{O}(n^2 + n^3/d) \text{ time}$$

$\mathcal{O}(n^3/d^3)$ instances of $\textsc{NegativeTriangle}$, each with $3d$ vertices

$$\Big\downarrow$$

$\mathcal{O}(n^3/d^3)$ instances of TED, each on a pair of combs of depth $6d + 1$

$$\Big\downarrow$$

TED, over $\mathcal{O}(n^{1.5}/\sqrt{d})$-size, $\mathcal{O}(d)$-depth trees

Wrap-up

$\textsc{NegativeTriangle}$, with $3n$ vertices

$$\Big\downarrow \quad \mathcal{O}(n^2 + n^3/d) \text{ time}$$

$\mathcal{O}(n^3/d^3)$ instances of $\textsc{NegativeTriangle}$, each with $3d$ vertices

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

$\mathcal{O}(n^3/d^3)$ instances of TED, each on a pair of combs of depth $6d + 1$

$$\Big\downarrow$$

TED, over $\mathcal{O}(n^{1.5}/\sqrt{d})$-size, $\mathcal{O}(d)$-depth trees

Wrap-up

$$\textsc{NegativeTriangle}\text{, with } 3n \text{ vertices}$$

$$\Big\downarrow \quad \mathcal{O}(n^2 + n^3/d) \text{ time}$$

$$\mathcal{O}(n^3/d^3) \text{ instances of } \textsc{NegativeTriangle}\text{, each with } 3d \text{ vertices}$$

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

$$\mathcal{O}(n^3/d^3) \text{ instances of } \mathrm{TED}\text{, each on a pair of combs of depth } 6d+1$$

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

$$\mathrm{TED}\text{, over } \mathcal{O}(n^{1.5}/\sqrt{d})\text{-size, } \mathcal{O}(d)\text{-depth trees}$$

Wrap-up

$\textsc{NegativeTriangle}$, with $3n$ vertices

$$\Big\downarrow \quad \mathcal{O}(n^2 + n^3/d) \text{ time}$$

$\mathcal{O}(n^3/d^3)$ instances of $\textsc{NegativeTriangle}$, each with $3d$ vertices

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

$\mathcal{O}(n^3/d^3)$ instances of TED, each on a pair of combs of depth $6d + 1$

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

TED, over $\mathcal{O}(n^{1.5}/\sqrt{d})$-size, $\mathcal{O}(d)$-depth trees

An algorithm for TED that takes time $\mathcal{O}(\text{size}^2 \cdot \text{depth}^{1-\epsilon})$ gives:

Wrap-up

$$\textsc{NegativeTriangle}\text{, with } 3n \text{ vertices}$$

$$\Big\downarrow \quad \mathcal{O}(n^2 + n^3/d) \text{ time}$$

$$\mathcal{O}(n^3/d^3) \text{ instances of } \textsc{NegativeTriangle}\text{, each with } 3d \text{ vertices}$$

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

$$\mathcal{O}(n^3/d^3) \text{ instances of } \mathrm{TED}\text{, each on a pair of combs of depth } 6d+1$$

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

$$\mathrm{TED}\text{, over } \mathcal{O}(n^{1.5}/\sqrt{d})\text{-size, } \mathcal{O}(d)\text{-depth trees}$$

An algorithm for TED that takes time $\mathcal{O}(\text{size}^2 \cdot \text{depth}^{1-\epsilon})$ gives:

$$\mathcal{O}((n^{1.5}/\sqrt{d})^2 \cdot d^{1-\epsilon}) = \mathcal{O}(n^3/d^{\epsilon})$$

Wrap-up

$\textsc{NegativeTriangle}$, with $3n$ vertices

$$\Big\downarrow \quad \mathcal{O}(n^2 + n^3/d) \text{ time}$$

$\mathcal{O}(n^3/d^3)$ instances of $\textsc{NegativeTriangle}$, each with $3d$ vertices

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

$\mathcal{O}(n^3/d^3)$ instances of TED, each on a pair of combs of depth $6d + 1$

$$\Big\downarrow \quad \mathcal{O}(n^3/d) \text{ time}$$

TED, over $\mathcal{O}(n^{1.5}/\sqrt{d})$-size, $\mathcal{O}(d)$-depth trees

An algorithm for TED that takes time $\mathcal{O}(\text{size}^2 \cdot \text{depth}^{1-\epsilon})$ gives:

$$\mathcal{O}((n^{1.5}/\sqrt{d})^2 \cdot d^{1-\epsilon}) = \mathcal{O}(n^3/d^{\epsilon})$$

and hence a strongly subcubic algorithm for $\textsc{NegativeTriangle}$.

Final Remarks and Open Problems

Final Remarks and Open Problems

Recent breakthrough $\mathcal{O}(n^{2.9546})$-time algorithm when all operations cost 1 [Mao; FOCS 2021]; announced improvement to $\mathcal{O}(n^{2.9149})$ [Dürr; arXiv 2022].

Final Remarks and Open Problems

Recent breakthrough $\mathcal{O}(n^{2.9546})$-time algorithm when all operations cost 1 [Mao; FOCS 2021]; announced improvement to $\mathcal{O}(n^{2.9149})$ [Dürr; arXiv 2022].

Can it be adapted for shallow trees?

Final Remarks and Open Problems

Recent breakthrough $\mathcal{O}(n^{2.9546})$-time algorithm when all operations cost 1 [Mao; FOCS 2021]; announced improvement to $\mathcal{O}(n^{2.9149})$ [Dürr; arXiv 2022].

Can it be adapted for shallow trees?

Can we improve the conditional lower bound or get one for smaller alphabets?

Final Remarks and Open Problems

Recent breakthrough $\mathcal{O}(n^{2.9546})$-time algorithm when all operations cost 1 [Mao; FOCS 2021]; announced improvement to $\mathcal{O}(n^{2.9149})$ [Dürr; arXiv 2022].

Can it be adapted for shallow trees?

Can we improve the conditional lower bound or get one for smaller alphabets?

Perhaps using the instance in the tight lower bound for decomposition algorithms.

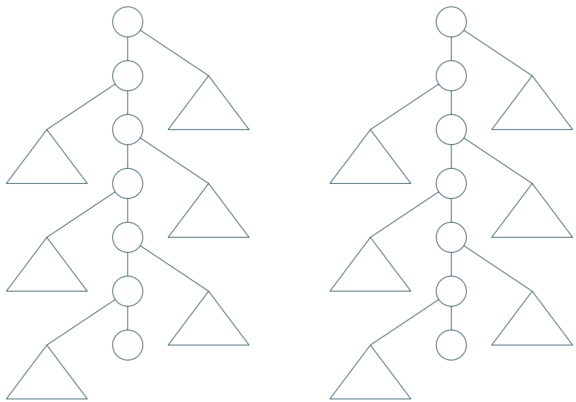
Final Remarks and Open Problems

Recent breakthrough $\mathcal{O}(n^{2.9546})$-time algorithm when all operations cost 1 [Mao; FOCS 2021]; announced improvement to $\mathcal{O}(n^{2.9149})$ [Dürr; arXiv 2022].

Can it be adapted for shallow trees?

Can we improve the conditional lower bound or get one for smaller alphabets?

Perhaps using the instance in the tight lower bound for decomposition algorithms.



The End

There is no $o(n^2 d^2)$-time decomposition algorithm.

There is no $\mathcal{O}(n^2 d^{1-\epsilon})$-time algorithm for any constant $\epsilon > 0$ when $d = \mathrm{poly}(n)$ and $\Sigma = \Omega(n)$ under the APSP hypothesis.

Thank you for your attention! Questions?