# Computing treedepth faster than $2^n$

Fedor V. Fomin, Archontia C. Giannopoulou, Michał Pilipczuk

Department of Informatics, University of Bergen, Norway

IPEC 2013, Sophia Antipolis, France,
September $4^{\text{th}}$, 2013

# Treedepth

- Let $G$ be a graph, and let $F$ be a forest of rooted trees on the same set of vertices.

# Treedepth

- Let $G$ be a graph, and let $F$ be a forest of rooted trees on the same set of vertices.
- We say that $G$ is *embedded* into $F$ if for every edge $uv$ of $G$ either $u$ is an ancestor of $v$, or $v$ is an ancestor of $u$.

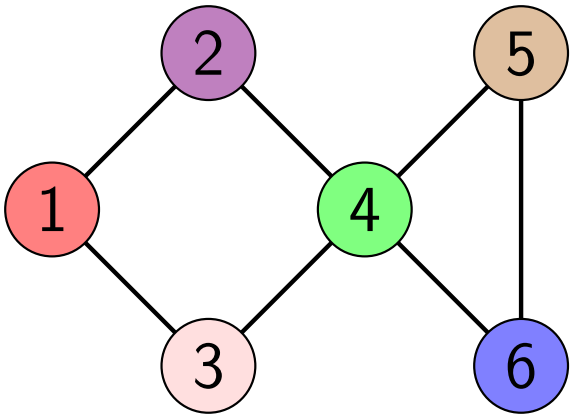
# Treedepth

- Let $G$ be a graph, and let $F$ be a forest of rooted trees on the same set of vertices.
- We say that $G$ is *embedded* into $F$ if for every edge $uv$ of $G$ either $u$ is an ancestor of $v$, or $v$ is an ancestor of $u$.
  - We also say that $F$ is a *treedepth decomposition* for $G$.

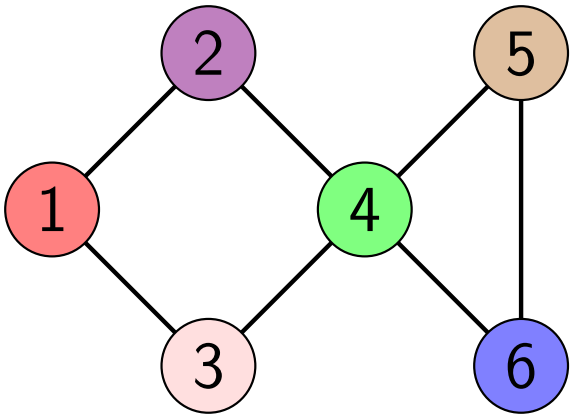
# Treedepth

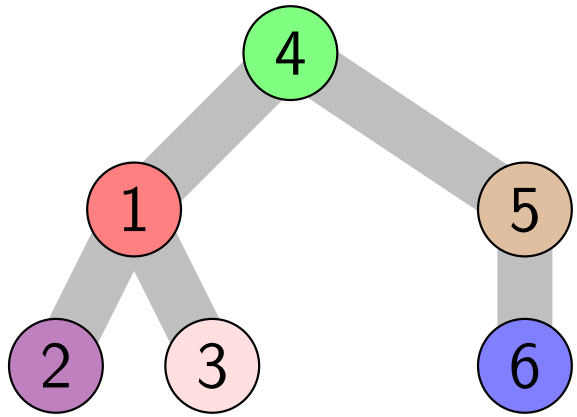
- Let $G$ be a graph, and let $F$ be a forest of rooted trees on the same set of vertices.
- We say that $G$ is *embedded* into $F$ if for every edge $uv$ of $G$ either $u$ is an ancestor of $v$, or $v$ is an ancestor of $u$.
  - We also say that $F$ is a *treedepth decomposition* for $G$.
- Treedepth of a graph, denoted $\mathbf{td}(G)$, is the least possible height of a forest into which $G$ can be embedded, where the height of a forest is maximum of heights of its trees.

# Example

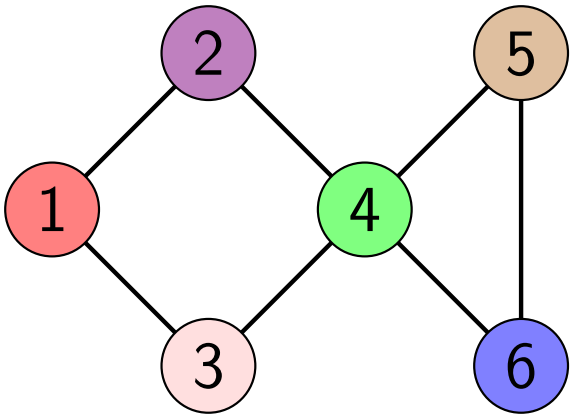


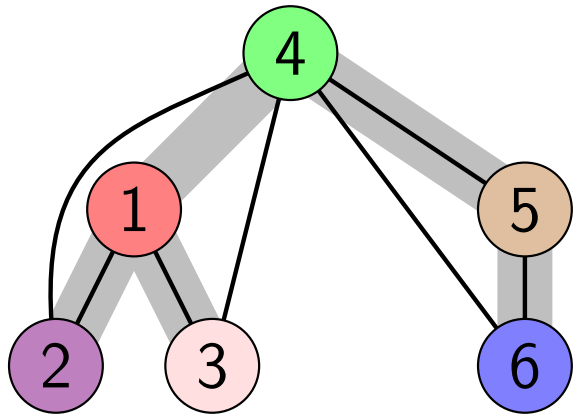
# Example



# Example



# Why treedepth?

- Rediscovered several times due to various applications.

# Why treedepth?

- Rediscovered several times due to various applications.
  - **Names**: treedepth, vertex ranking number, ordered coloring, minimum height of an elimination tree.

# Why treedepth?

- Rediscovered several times due to various applications.
  - **Names**: treedepth, vertex ranking number, ordered coloring, minimum height of an elimination tree.
  - **Applications**: graphs of bounded expansion, divide-and-conquer algorithms for hard problems, factorization of sparse matrices.

# Why treedepth?

- Rediscovered several times due to various applications.
  - **Names**: treedepth, vertex ranking number, ordered coloring, minimum height of an elimination tree.
  - **Applications**: graphs of bounded expansion, divide-and-conquer algorithms for hard problems, factorization of sparse matrices.
- $\mathbf{tw}(G) \leqslant \mathbf{td}(G) \leqslant \mathbf{tw}(G) \cdot \log n$

# Why treedepth?

- Rediscovered several times due to various applications.
  - **Names**: treedepth, vertex ranking number, ordered coloring, minimum height of an elimination tree.
  - **Applications**: graphs of bounded expansion, divide-and-conquer algorithms for hard problems, factorization of sparse matrices.
- $\mathbf{tw}(G) \leqslant \mathbf{td}(G) \leqslant \mathbf{tw}(G) \cdot \log n$
- One of the most overlooked graph parameters that we only now begin to understand.

# Our study

- We study treedepth from the point of view of **exact algorithms**.

# Our study

- We study treedepth from the point of view of **exact algorithms**.
- Natural DP on subsets: $\mathcal{O}^\star(2^n)$.

# Our study

- We study treedepth from the point of view of **exact algorithms**.
- Natural DP on subsets: $\mathcal{O}^\star(2^n)$.
  - Other parameters, like treewidth or pathwidth, admit algorithms with running time $\mathcal{O}^\star(c^n)$ for $c < 2$.

# Our study

- We study treedepth from the point of view of **exact algorithms**.
- Natural DP on subsets: $\mathcal{O}^\star(2^n)$.
  - Other parameters, like treewidth or pathwidth, admit algorithms with running time $\mathcal{O}^\star(c^n)$ for $c < 2$.
- **Our result**: An exact algorithm computing treedepth in time $\mathcal{O}^\star\left(\binom{n}{0.4 \cdot n}\right) \leqslant \mathcal{O}^\star(1.9602^n)$.

# Our study

- We study treedepth from the point of view of **exact algorithms**.
- Natural DP on subsets: $\mathcal{O}^\star(2^n)$.
  - Other parameters, like treewidth or pathwidth, admit algorithms with running time $\mathcal{O}^\star(c^n)$ for $c < 2$.
- **Our result**: An exact algorithm computing treedepth in time $\mathcal{O}^\star\left(\binom{n}{0.4 \cdot n}\right) \leqslant \mathcal{O}^\star(1.9602^n)$.
- **This talk**: Sketching breaking $2^n$.

# Trees and forests

- If a graph is connected, then it must be embedded into a rooted tree. Treedepth of a graph is maximum of treedepths of its connected components.

# Trees and forests

- If a graph is connected, then it must be embedded into a rooted tree. Treedepth of a graph is maximum of treedepths of its connected components.
- From now on we concentrate on the case when $G$ is connected, so the decomposition must be a tree.

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.
- For a rooted tree $T$, let $\ell_i(T)$ be the number of vertices on the $i$-th level of $T$.

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.
- For a rooted tree $T$, let $\ell_i(T)$ be the number of vertices on the $i$-th level of $T$.
- Consider the sequence $\ell_1(T), \ell_2(T), \ell_3(T), \ldots$

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.
- For a rooted tree $T$, let $\ell_i(T)$ be the number of vertices on the $i$-th level of $T$.
- Consider the sequence $\ell_1(T), \ell_2(T), \ell_3(T), \ldots$
- We say that $T_1 \prec T_2$ if there is an index $i$ such that $\ell_i(T_1) < \ell_i(T_2)$ and $\ell_j(T_1) = \ell_j(T_2)$ for all $j > i$.

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.
- For a rooted tree $T$, let $\ell_i(T)$ be the number of vertices on the $i$-th level of $T$.
- Consider the sequence $\ell_1(T), \ell_2(T), \ell_3(T), \ldots$
- We say that $T_1 \prec T_2$ if there is an index $i$ such that $\ell_i(T_1) < \ell_i(T_2)$ and $\ell_j(T_1) = \ell_j(T_2)$ for all $j > i$.
- Tree $T$ is *minimal* for $G$ if

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.
- For a rooted tree $T$, let $\ell_i(T)$ be the number of vertices on the $i$-th level of $T$.
- Consider the sequence $\ell_1(T), \ell_2(T), \ell_3(T), \ldots$
- We say that $T_1 \prec T_2$ if there is an index $i$ such that $\ell_i(T_1) < \ell_i(T_2)$ and $\ell_j(T_1) = \ell_j(T_2)$ for all $j > i$.
- Tree $T$ is *minimal* for $G$ if
  (i) $G$ is embedded in $T$, and

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.
- For a rooted tree $T$, let $\ell_i(T)$ be the number of vertices on the $i$-th level of $T$.
- Consider the sequence $\ell_1(T), \ell_2(T), \ell_3(T), \ldots$
- We say that $T_1 \prec T_2$ if there is an index $i$ such that $\ell_i(T_1) < \ell_i(T_2)$ and $\ell_j(T_1) = \ell_j(T_2)$ for all $j > i$.
- Tree $T$ is *minimal* for $G$ if
  - (i) $G$ is embedded in $T$, and
  - (ii) $T$ is $\prec$-minimum among trees satisfying (i).

# Minimal trees

- To have more control over the decomposition, we define a class of *minimal trees*.
- For a rooted tree $T$, let $\ell_i(T)$ be the number of vertices on the $i$-th level of $T$.
- Consider the sequence $\ell_1(T), \ell_2(T), \ell_3(T), \ldots$
- We say that $T_1 \prec T_2$ if there is an index $i$ such that $\ell_i(T_1) < \ell_i(T_2)$ and $\ell_j(T_1) = \ell_j(T_2)$ for all $j > i$.
- Tree $T$ is *minimal* for $G$ if
  - (i) $G$ is embedded in $T$, and
  - (ii) $T$ is $\prec$-minimum among trees satisfying (i).
- **Note**: a minimal tree has minimum possible height.

# Minimal trees: properties

- **Notation**: $T_u$ is a subtree rooted in $u$.

# Minimal trees: properties

- **Notation**: $T_u$ is a subtree rooted in $u$.

**Lemma 1**

Let $T$ be a minimal tree for $G$ and let $v$ be any vertex. Then $G[V(T_v)]$ is connected.

# Minimal trees: properties

- **Notation**: $T_u$ is a subtree rooted in $u$.

**Lemma 1**

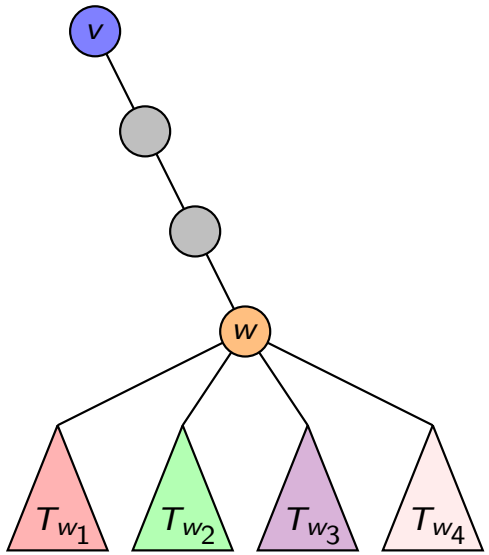
Let $T$ be a minimal tree for $G$ and let $v$ be any vertex. Then $G[V(T_v)]$ is connected.

- **Proof sketch**: Otherwise one can partition $T_v$ into two smaller subtrees, and replacing $T_v$ with these subtrees in $T$ gives a decomposition smaller wrt. $\prec$.

# Minimal trees: properties

## Lemma 2

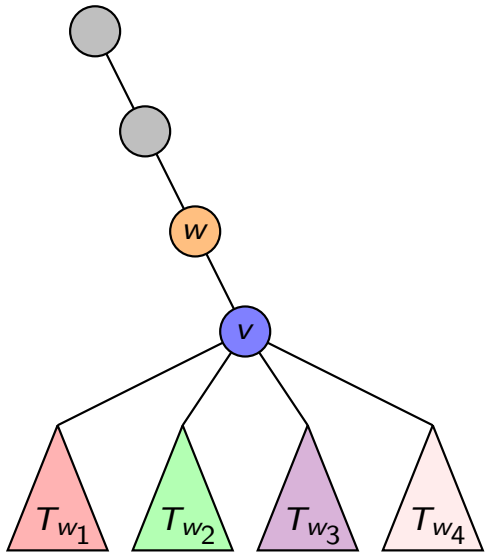
Let $T$ be a minimal tree for $G$ and let $v$ be any vertex. Let $w$ be the nearest branching point of $T$ below $v$ (possibly $v = w$), and let $w_1, w_2, \ldots, w_p$ be children of $w$. Then vertex $v$ has a neighbour in $G$ in each of sets $V(T_{w_1}), V(T_{w_2}), \ldots, V(T_{w_p})$.



# Minimal trees: properties

## Lemma 2

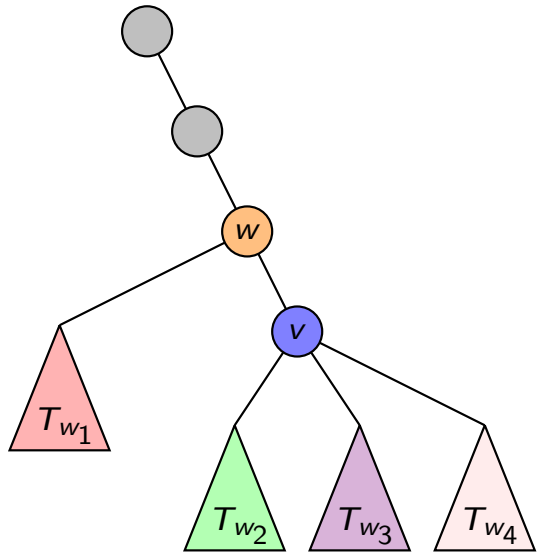
Let $T$ be a minimal tree for $G$ and let $v$ be any vertex. Let $w$ be the nearest branching point of $T$ below $v$ (possibly $v = w$), and let $w_1, w_2, \ldots, w_p$ be children of $w$. Then vertex $v$ has a neighbour in $G$ in each of sets $V(T_{w_1}), V(T_{w_2}), \ldots, V(T_{w_p})$.



# Minimal trees: properties

## Lemma 2

Let $T$ be a minimal tree for $G$ and let $v$ be any vertex. Let $w$ be the nearest branching point of $T$ below $v$ (possibly $v = w$), and let $w_1, w_2, \ldots, w_p$ be children of $w$. Then vertex $v$ has a neighbour in $G$ in each of sets $V(T_{w_1}), V(T_{w_2}), \ldots, V(T_{w_p})$.



# A simple DP

- **Recurrence**: For $X \subseteq V(G)$ inducing a connected subgraph, the following holds:

$$\mathbf{td}(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v)} \mathbf{td}(H)\right).$$

# A simple DP

- **Recurrence**: For $X \subseteq V(G)$ inducing a connected subgraph, the following holds:

$$\mathbf{td}(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v)} \mathbf{td}(H)\right).$$

- Compute $\mathbf{td}(G[X])$ for $X$ inducing connected subgraphs in a bottom-up manner. Running time: $\mathcal{O}^\star(2^n)$.

# A simple DP

- **Recurrence**: For $X \subseteq V(G)$ inducing a connected subgraph, the following holds:

$$\mathbf{td}(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v)} \mathbf{td}(H)\right).$$

- Compute $\mathbf{td}(G[X])$ for $X$ inducing connected subgraphs in a bottom-up manner. Running time: $\mathcal{O}^\star(2^n)$.
- **Strategy**:

# A simple DP

- **Recurrence**: For $X \subseteq V(G)$ inducing a connected subgraph, the following holds:

$$\mathbf{td}(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v)} \mathbf{td}(H)\right).$$

- Compute $\mathbf{td}(G[X])$ for $X$ inducing connected subgraphs in a bottom-up manner. Running time: $\mathcal{O}^\star(2^n)$.
- **Strategy**:
  - Restrict the family of sets used in the above DP.

# A simple DP

- **Recurrence**: For $X \subseteq V(G)$ inducing a connected subgraph, the following holds:

$$\mathbf{td}(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v)} \mathbf{td}(H)\right).$$

- Compute $\mathbf{td}(G[X])$ for $X$ inducing connected subgraphs in a bottom-up manner. Running time: $\mathcal{O}^\star(2^n)$.
- **Strategy**:
  - Restrict the family of sets used in the above DP.
  - Show that if the pruned DP omitted the optimum decomposition, then this decomposition must have a special structure for which there is significantly less than $2^n$ possibilities.

# Pruned family of states

- Fix some $\varepsilon > 0$. Let $\mathcal{S}$ consist of:

# Pruned family of states

- Fix some $\varepsilon > 0$. Let $\mathcal{S}$ consist of:
  (i) all the connected subsets of $V(G)$ of size at most $(\frac{1}{2} - \varepsilon) \cdot n$,

# Pruned family of states

- Fix some $\varepsilon > 0$. Let $\mathcal{S}$ consist of:
  - (i) all the connected subsets of $V(G)$ of size at most $(\frac{1}{2} - \varepsilon) \cdot n$,
  - (ii) and all the subsets of $V(G)$ of the following form: *the vertex set of a connected component of $G \setminus Y$ for some $Y \subseteq V(G)$ with $|Y| \leqslant (\frac{1}{2} - \varepsilon) \cdot n$.*

# Pruned family of states

- Fix some $\varepsilon > 0$. Let $\mathcal{S}$ consist of:
  - (i) all the connected subsets of $V(G)$ of size at most $(\frac{1}{2} - \varepsilon) \cdot n$,
  - (ii) and all the subsets of $V(G)$ of the following form: *the vertex set of a connected component of* $G \setminus Y$ *for some* $Y \subseteq V(G)$ *with* $|Y| \leqslant (\frac{1}{2} - \varepsilon) \cdot n$.
- Cardinality of $\mathcal{S}$ is at most $\mathcal{O}^\star\left(\binom{n}{(\frac{1}{2}-\varepsilon)\cdot n}\right)$.

# The pruned DP

- **Recurrence**: For $X \in \mathcal{S}$, let:

$$\mathbf{td}_*(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v), V(H) \in \mathcal{S}} \mathbf{td}_*(H)\right),$$

where $\mathbf{td}_*(G[X]) = +\infty$ if $X \notin \mathcal{S}$.

# The pruned DP

- **Recurrence**: For $X \in \mathcal{S}$, let:

$$\mathbf{td}_*(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v), V(H) \in \mathcal{S}} \mathbf{td}_*(H)\right),$$

  where $\mathbf{td}_*(G[X]) = +\infty$ if $X \notin \mathcal{S}$.
- Values of $\mathbf{td}_*(\cdot)$ can be computed in $\mathcal{O}^*(|\mathcal{S}|)$ time, and $\mathbf{td}_*(G[X]) \geqslant \mathbf{td}(G[X])$.

# The pruned DP

- **Recurrence**: For $X \in \mathcal{S}$, let:

$$\mathbf{td}_*(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v), V(H) \in \mathcal{S}} \mathbf{td}_*(H)\right),$$

  where $\mathbf{td}_*(G[X]) = +\infty$ if $X \notin \mathcal{S}$.
- Values of $\mathbf{td}_*(\cdot)$ can be computed in $\mathcal{O}^\star(|\mathcal{S}|)$ time, and $\mathbf{td}_*(G[X]) \geqslant \mathbf{td}(G[X])$.
- **Observation**: if for some optimum tree $T$ for $G[X]$ it holds that $V(T_u) \in \mathcal{S}$ for all $u \in X$, then $\mathbf{td}_*(G[X]) = \mathbf{td}(G[X])$.

# The pruned DP

- **Recurrence**: For $X \in \mathcal{S}$, let:

$$\mathbf{td}_*(G[X]) = \min_{v \in X} \left(1 + \max_{H \in \mathcal{C}(G[X] \setminus v), V(H) \in \mathcal{S}} \mathbf{td}_*(H)\right),$$

  where $\mathbf{td}_*(G[X]) = +\infty$ if $X \notin \mathcal{S}$.
- Values of $\mathbf{td}_*(\cdot)$ can be computed in $\mathcal{O}^\star(|\mathcal{S}|)$ time, and $\mathbf{td}_*(G[X]) \geqslant \mathbf{td}(G[X])$.
- **Observation**: if for some optimum tree $T$ for $G[X]$ it holds that $V(T_u) \in \mathcal{S}$ for all $u \in X$, then $\mathbf{td}_*(G[X]) = \mathbf{td}(G[X])$.
- **Corollary**:
  $\mathbf{td}_*(G[X]) = \mathbf{td}(G[X])$ for all $X$ with $|X| \leqslant (\frac{1}{2} - \varepsilon) \cdot n$.

# Looking for omitted trees

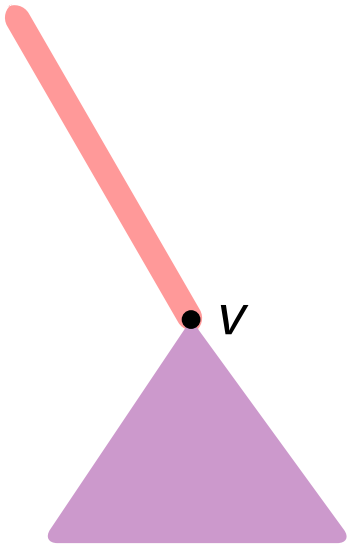
- $\mathbf{td}_*(G) = \mathbf{td}(G)$ holds unless for every optimum tree there is a subtree whose vertex set is omitted in $\mathcal{S}$.

# Looking for omitted trees

- $\mathbf{td}_*(G) = \mathbf{td}(G)$ holds unless for every optimum tree there is a subtree whose vertex set is omitted in $\mathcal{S}$.
- Let $T$ be a minimal tree for $G$.

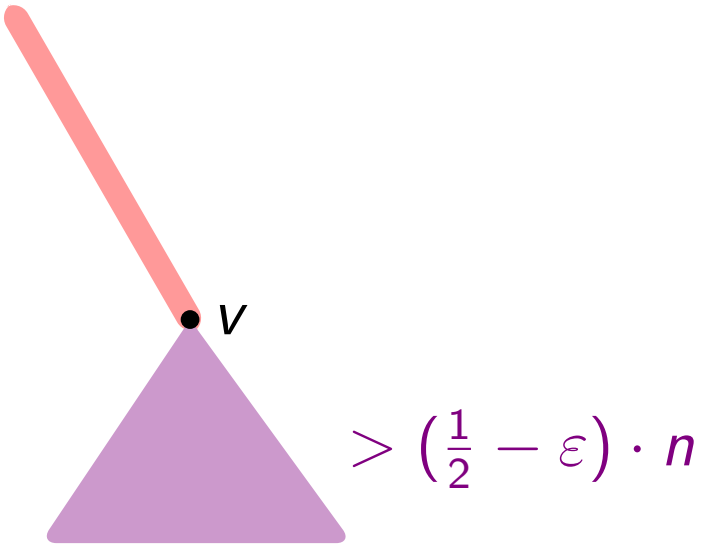
# Looking for omitted trees

- $\mathbf{td}_*(G) = \mathbf{td}(G)$ holds unless for every optimum tree there is a subtree whose vertex set is omitted in $\mathcal{S}$.
- Let $T$ be a minimal tree for $G$.
- Set $V(T_v)$ can be omitted only if



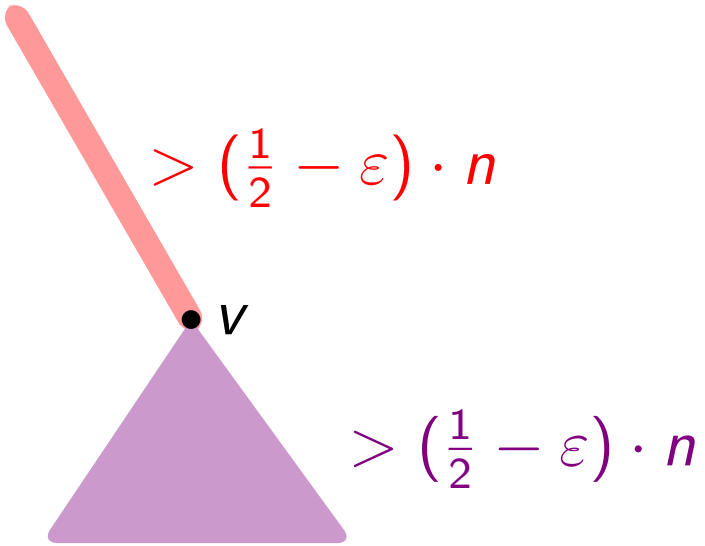
# Looking for omitted trees

- $\mathbf{td}_*(G) = \mathbf{td}(G)$ holds unless for every optimum tree there is a subtree whose vertex set is omitted in $\mathcal{S}$.
- Let $T$ be a minimal tree for $G$.
- Set $V(T_v)$ can be omitted only if
  - $V(T_v)$ is of size more than $(\frac{1}{2} - \varepsilon) \cdot n$, and

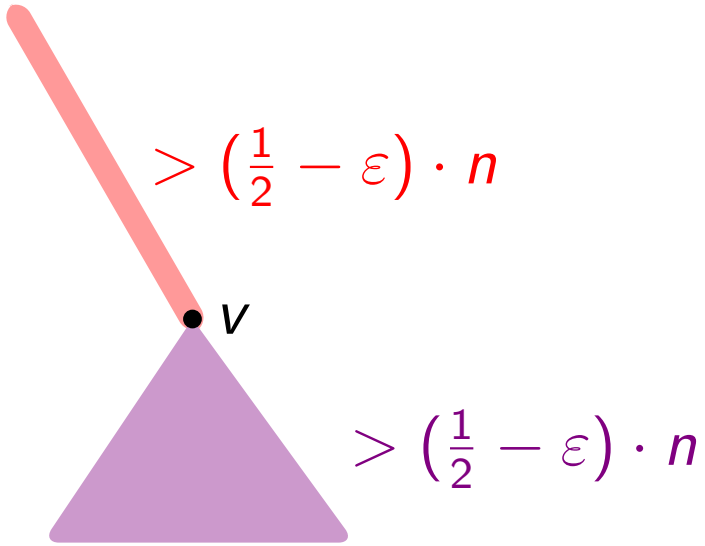


# Looking for omitted trees

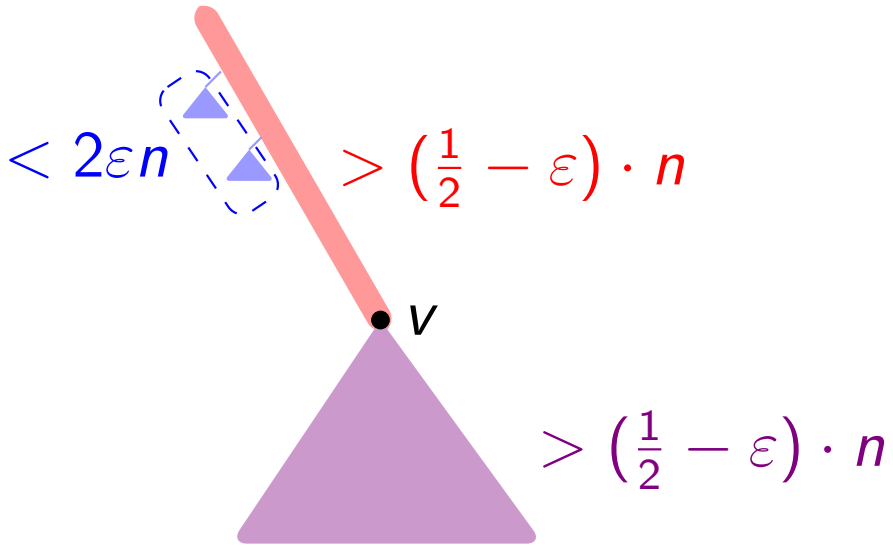
- $\mathbf{td}_*(G) = \mathbf{td}(G)$ holds unless for every optimum tree there is a subtree whose vertex set is omitted in $\mathcal{S}$.
- Let $T$ be a minimal tree for $G$.
- Set $V(T_v)$ can be omitted only if
  - $V(T_v)$ is of size more than $(\frac{1}{2} - \varepsilon) \cdot n$, and
  - $v$ is at depth more than $(\frac{1}{2} - \varepsilon) \cdot n$ in $T$.



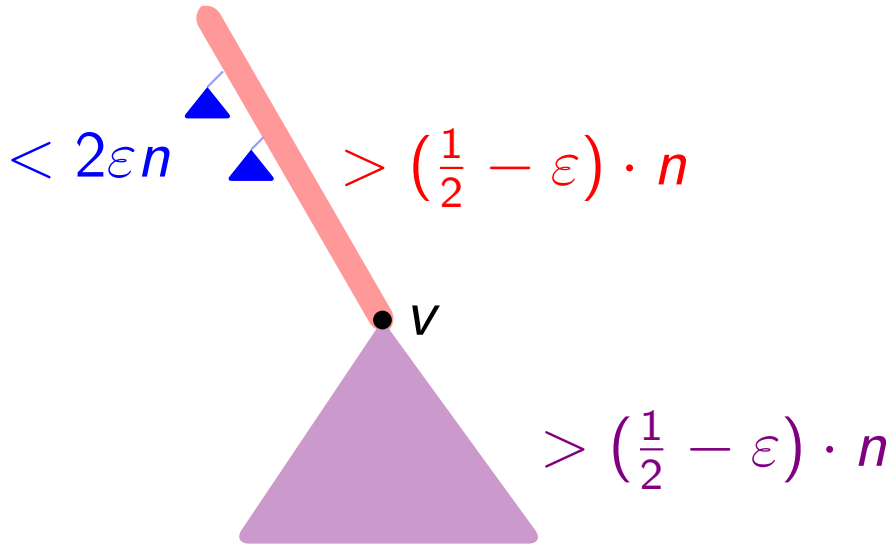
# Guessing an omitted minimal tree (sketch)



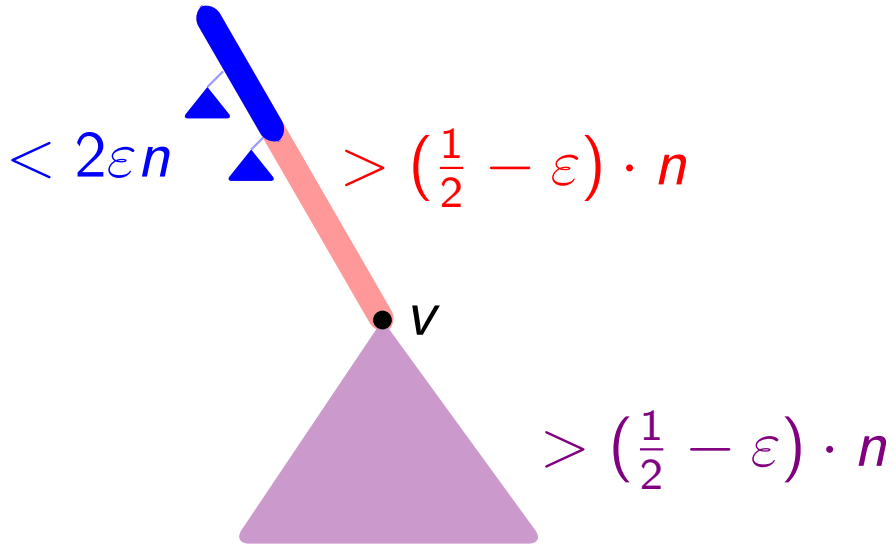
# Guessing an omitted minimal tree (sketch)



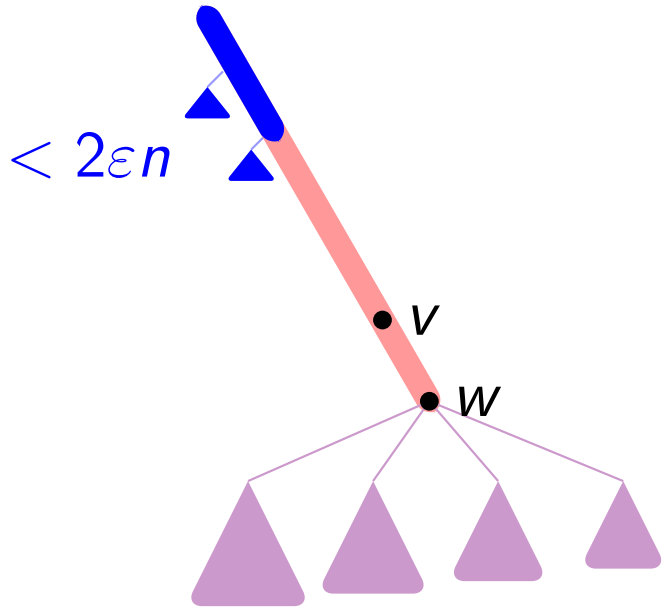
# Guessing an omitted minimal tree (sketch)



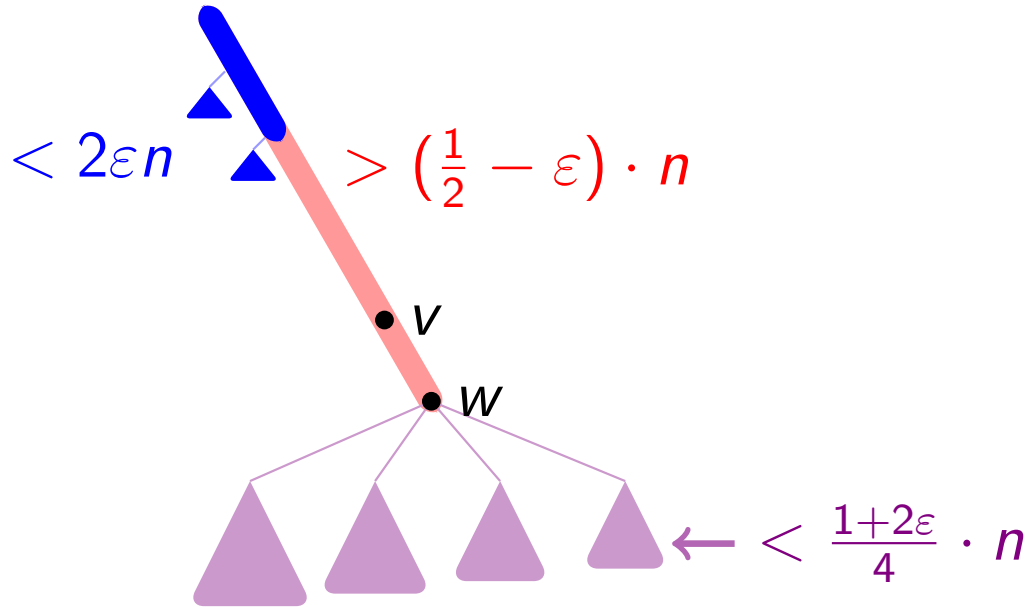
# Guessing an omitted minimal tree (sketch)



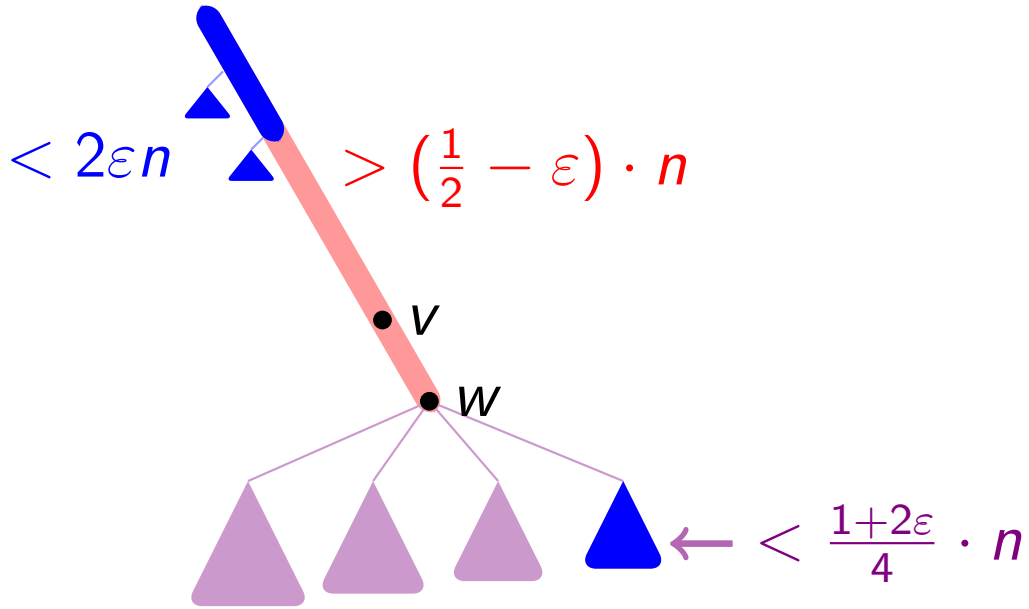
# Guessing an omitted minimal tree (sketch)



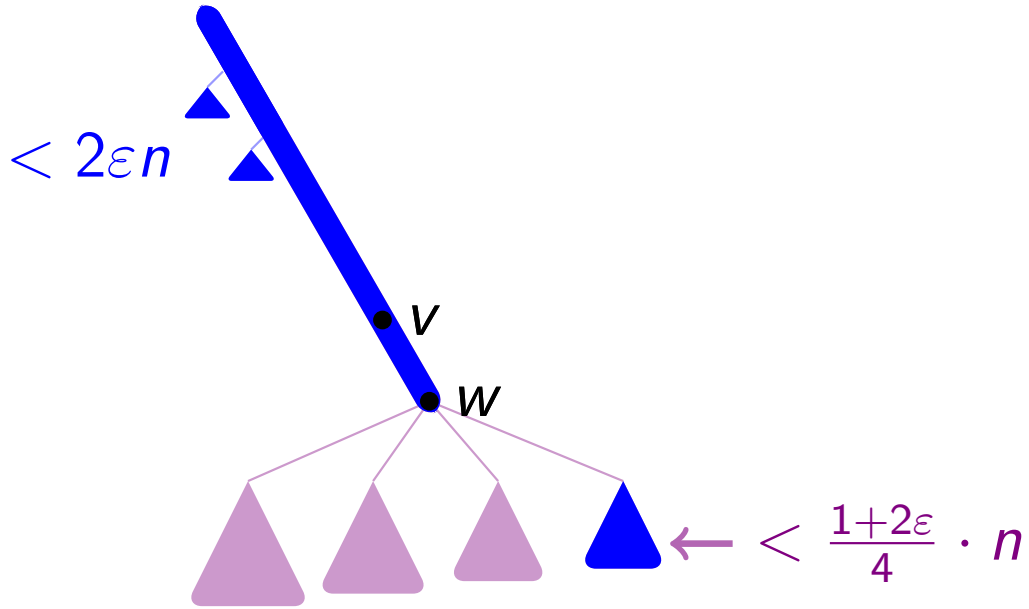
# Guessing an omitted minimal tree (sketch)



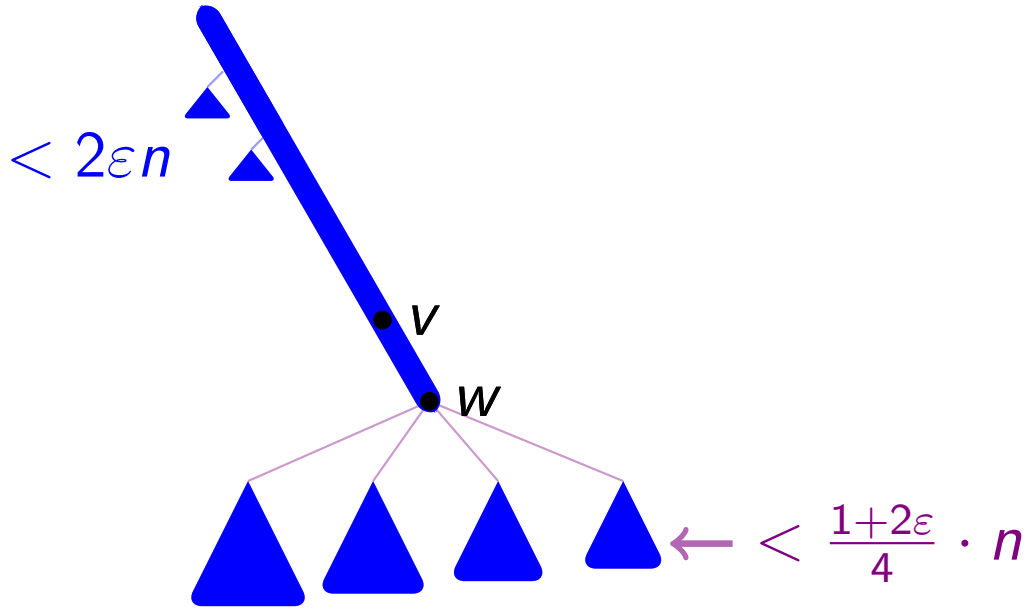
# Guessing an omitted minimal tree (sketch)



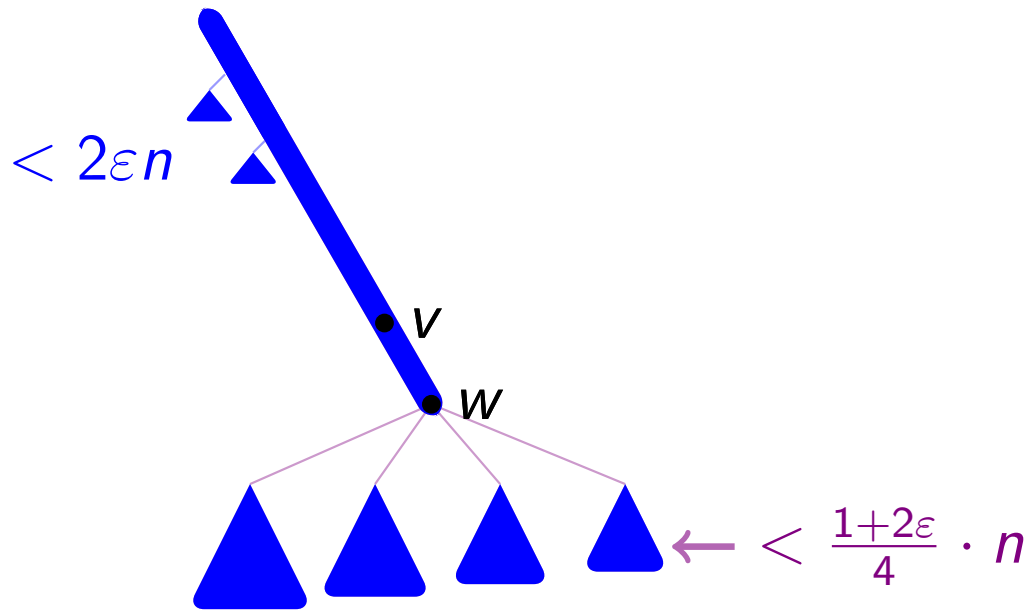
# Guessing an omitted minimal tree (sketch)



# Guessing an omitted minimal tree (sketch)

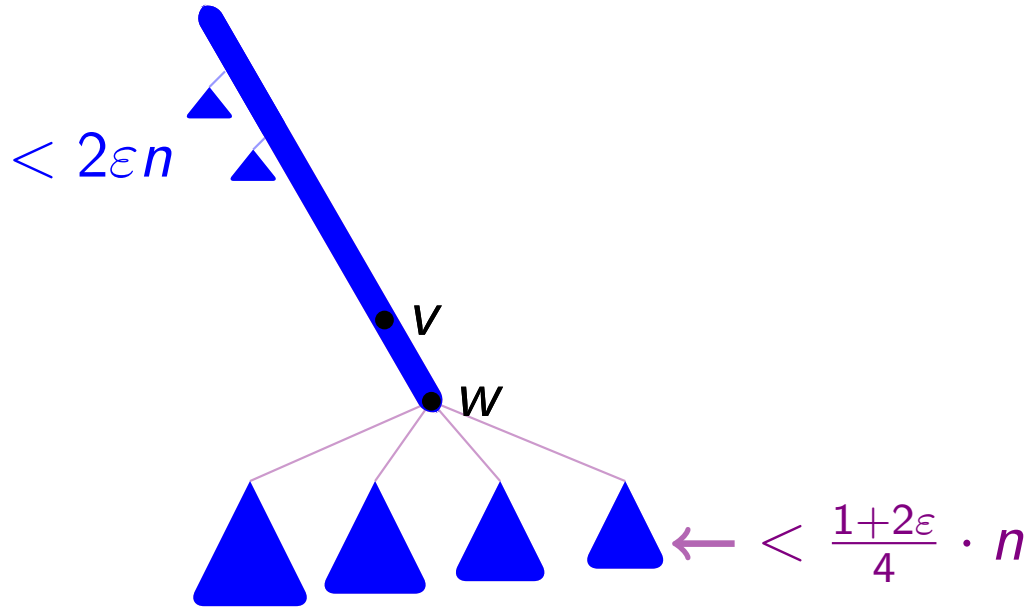


# Guessing an omitted minimal tree (sketch)




- **Number of choices**: at most $\mathcal{O}^\star\left(\binom{n}{2\varepsilon n} \cdot \binom{n}{\frac{1+2\varepsilon}{4} \cdot n}\right)$.

# Guessing an omitted minimal tree (sketch)




- **Number of choices**: at most $\mathcal{O}^\star\left(\binom{n}{2\varepsilon n} \cdot \binom{n}{\frac{1+2\varepsilon}{4} \cdot n}\right)$.
- For each choice, use $\mathcal{O}^\star\left(\binom{n}{2\varepsilon n}\right)$ time to compute the treedepth of each subtree.

# Conclusions

- We have presented an exact algorithm computing treedepth with running time $\mathcal{O}^\star(1.9602^n)$.

# Conclusions

- We have presented an exact algorithm computing treedepth with running time $\mathcal{O}^\star(1.9602^n)$.
- The algorithm is based on properties of *minimal trees*, which can be useful also in other contexts.

# Conclusions

- We have presented an exact algorithm computing treedepth with running time $\mathcal{O}^\star(1.9602^n)$.
- The algorithm is based on properties of *minimal trees*, which can be useful also in other contexts.
- Can we get significantly below $\mathcal{O}^\star(1.9602^n)$?

# Conclusions

- We have presented an exact algorithm computing treedepth with running time $\mathcal{O}^\star(1.9602^n)$.
- The algorithm is based on properties of *minimal trees*, which can be useful also in other contexts.
- Can we get significantly below $\mathcal{O}^\star(1.9602^n)$?
- What can we do in polyspace?

# Conclusions

- We have presented an exact algorithm computing treedepth with running time $\mathcal{O}^\star(1.9602^n)$.
- The algorithm is based on properties of *minimal trees*, which can be useful also in other contexts.
- Can we get significantly below $\mathcal{O}^\star(1.9602^n)$?
- What can we do in polyspace?
- **Thank you!**