Well Quasi Orders: GMT and VC

Exposition by William Gasarch

February 25, 2026

The Vertex Cover Problem

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G, k)\colon G \text{ has a vertex cover of size } k\ \}$.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G, k) \colon G \text{ has a vertex cover of size } k \}$.
VC is known to be NPC.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G, k) : G \text{ has a vertex cover of size } k\}$.
VC is known to be NPC.

Def $\mathrm{VC}_k = \{G : G \text{ has a vertex cover of size } k\}$.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G, k) : G \text{ has a vertex cover of size } k\}$.
VC is known to be NPC.

Def $\mathrm{VC}_k = \{G : G \text{ has a vertex cover of size } k\}$.

VC_k is clearly in $O(n^k)$ time. Can we do better? **VOTE**

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G, k)\colon G \text{ has a vertex cover of size } k\ \}$.
VC is known to be NPC.

Def $\mathrm{VC}_k = \{G\colon G \text{ has a vertex cover of size } k\ \}$.

VC_k is clearly in $O(n^k)$ time. Can we do better? **VOTE**

$(\exists \epsilon > 0)$ under hardness assumptions VC_k is in $\Omega(n^{\epsilon k})$.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G, k)\colon G \text{ has a vertex cover of size } k\ \}$.
VC is known to be NPC.

Def $\mathrm{VC}_k = \{G\colon G \text{ has a vertex cover of size } k\ \}$.

VC_k is clearly in $O(n^k)$ time. Can we do better? **VOTE**

$(\exists \epsilon > 0)$ under hardness assumptions VC_k is in $\Omega(n^{\epsilon k})$.

$(\exists a \in \mathbb{N})$ VC_k is in $O(f(k)n^a)$.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G, k)\colon G \text{ has a vertex cover of size } k\ \}$.
VC is known to be NPC.

Def $\mathrm{VC}_k = \{G\colon G \text{ has a vertex cover of size } k\ \}$.

VC_k is clearly in $O(n^k)$ time. Can we do better? **VOTE**

$(\exists \epsilon > 0)$ under hardness assumptions VC_k is in $\Omega(n^{\epsilon k})$.

$(\exists a \in \mathbb{N})$ VC_k is in $O(f(k)n^a)$.

$O(2^k n)$ and this is the best known.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G,k)\colon G \text{ has a vertex cover of size } k\ \}$.
VC is known to be NPC.

Def $\mathrm{VC}_k = \{G\colon G \text{ has a vertex cover of size } k\ \}$.

VC_k is clearly in $O(n^k)$ time. Can we do better? **VOTE**

$(\exists \epsilon > 0)$ under hardness assumptions VC_k is in $\Omega(n^{\epsilon k})$.

$(\exists a \in \mathbb{N})$ VC_k is in $O(f(k)n^a)$.

$O(2^k n)$ and this is the best known.

$O(kn + 1.274^k)$ and this is the best known.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Def $\mathrm{VC} = \{(G,k)\colon G \text{ has a vertex cover of size } k\ \}$.
VC is known to be NPC.

Def $\mathrm{VC}_k = \{G\colon G \text{ has a vertex cover of size } k\ \}$.

VC_k is clearly in $O(n^k)$ time. Can we do better? **VOTE**

$(\exists \epsilon > 0)$ under hardness assumptions VC_k is in $\Omega(n^{\epsilon k})$.

$(\exists a \in \mathbb{N})$ VC_k is in $O(f(k)n^a)$.

$O(2^k n)$ and this is the best known.

$O(kn + 1.274^k)$ and this is the best known.

Answer on next slide.

Vertex Cover

Vertex Cover

We will do the following:

Vertex Cover

We will do the following:

a) VC_k is in $O(f(k)n^3)$. This uses GMT.

Vertex Cover

We will do the following:

a) VC_k is in $O(f(k)n^3)$. This uses GMT.

b) VC_k is in $O(kn + 1.34^k)$. This is easy and practical.

Vertex Cover

We will do the following:

a) VC_k is in $O(f(k)n^3)$. This uses GMT.

b) VC_k is in $O(kn + 1.34^k)$. This is easy and practical.

Hence it has no place in this course!

Vertex Cover

We will do the following:

a) VC_k is in $O(f(k)n^3)$. This uses GMT.

b) VC_k is in $O(kn + 1.34^k)$. This is easy and practical.

Hence it has no place in this course!
We won't be covering it.

Obstruction Sets For Planarity

Obstruction Sets For Planar Graphs

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT,

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem?

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem? Yes.

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem? Yes.

Thm $\exists$ a finite set of graphs **OBS** such that

G is planar iff $(\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G]$.

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem? Yes.

Thm $\exists$ a finite set of graphs **OBS** such that

G is planar iff $(\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G]$.

Let $\mathbf{OBS} = \{H\colon H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem? Yes.

Thm $\exists$ a finite set of graphs **OBS** such that

G is planar iff $(\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G]$.

Let $\mathbf{OBS} = \{H : H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

On the next two slides we show:

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem? Yes.

Thm $\exists$ a finite set of graphs **OBS** such that

$$G \text{ is planar iff } (\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G].$$

Let $\mathbf{OBS} = \{H : H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

On the next two slides we show:

1) **OBS** works. That is:

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem? Yes.

Thm $\exists$ a finite set of graphs **OBS** such that

G is planar iff $(\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G]$.

Let $\mathbf{OBS} = \{H : H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

On the next two slides we show:

1) **OBS** works. That is:

G is planar iff $(\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G]$.

Obstruction Sets For Planar Graphs

(Wagner's Thm) G is planar IFF $((K_{3,3} \not\preceq_{\mathrm{m}} G)$ and $(K_5 \not\preceq_{\mathrm{m}} G))$

If we didn't know Wagner's Theorem but did know the GMT, could we deduce something similar to Wagner's theorem? Yes.

Thm $\exists$ a finite set of graphs **OBS** such that

$$G \text{ is planar iff } (\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G].$$

Let $\mathbf{OBS} = \{H : H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

On the next two slides we show:

1) **OBS** works. That is:

$$G \text{ is planar iff } (\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G].$$

2) **OBS** is finite.

OBS Works

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}.$

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

Case 2.1 $G_1 \in \mathbf{OBS}$. Done.

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

Case 2.1 $G_1 \in \mathbf{OBS}$. Done.

Case 2.2 $G_1 \notin \mathbf{OBS}$. Then $\exists G_2$ nonplanar, $G_2 \prec_{\mathrm{m}} G$.

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

Case 2.1 $G_1 \in \mathbf{OBS}$. Done.

Case 2.2 $G_1 \notin \mathbf{OBS}$. Then $\exists G_2$ nonplanar, $G_2 \prec_{\mathrm{m}} G$.

If this goes on forever then we have

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

Case 2.1 $G_1 \in \mathbf{OBS}$. Done.

Case 2.2 $G_1 \notin \mathbf{OBS}$. Then $\exists G_2$ nonplanar, $G_2 \prec_{\mathrm{m}} G$.

If this goes on forever then we have

$\cdots G_i \prec_{\mathrm{m}} G_{i-1} \prec_{\mathrm{m}} \cdots \prec_{\mathrm{m}} G_1 \prec_{\mathrm{m}} G$.

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

Case 2.1 $G_1 \in \mathbf{OBS}$. Done.

Case 2.2 $G_1 \notin \mathbf{OBS}$. Then $\exists G_2$ nonplanar, $G_2 \prec_{\mathrm{m}} G$.

If this goes on forever then we have

$\cdots G_i \prec_{\mathrm{m}} G_{i-1} \prec_{\mathrm{m}} \cdots \prec_{\mathrm{m}} G_1 \prec_{\mathrm{m}} G$.

Two ways to show this is impossible:

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

Case 2.1 $G_1 \in \mathbf{OBS}$. Done.

Case 2.2 $G_1 \notin \mathbf{OBS}$. Then $\exists G_2$ nonplanar, $G_2 \prec_{\mathrm{m}} G$.

If this goes on forever then we have

$\cdots G_i \prec_{\mathrm{m}} G_{i-1} \prec_{\mathrm{m}} \cdots \prec_{\mathrm{m}} G_1 \prec_{\mathrm{m}} G$.

Two ways to show this is impossible:

1) The Graphs keep getting smaller so eventually planar.

OBS Works

$\mathbf{OBS} = \{H\colon H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Let G be nonplanar.

Case 1 $G \in \mathbf{OBS}$. Done.

Case 2 $G \notin \mathbf{OBS}$. Then $\exists G_1$ nonplanar, $G_1 \prec_{\mathrm{m}} G$.

Case 2.1 $G_1 \in \mathbf{OBS}$. Done.

Case 2.2 $G_1 \notin \mathbf{OBS}$. Then $\exists G_2$ nonplanar, $G_2 \prec_{\mathrm{m}} G$.

If this goes on forever then we have

$\cdots G_i \prec_{\mathrm{m}} G_{i-1} \prec_{\mathrm{m}} \cdots \prec_{\mathrm{m}} G_1 \prec_{\mathrm{m}} G$.

Two ways to show this is impossible:

1) The Graphs keep getting smaller so eventually planar.

2) This is a bad sequence, contradicts GMT.

Obstruction Sets Are Finite

We need that **OBS** is finite.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H : H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H : H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
$\mathbf{OBS} = \{H : H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Every element of **OBS** is minimal:
if $H \in \mathbf{OBS}$ then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
$\mathbf{OBS} = \{H : H \text{ not planar and } (\forall H' \prec_{\mathrm{m}} H)[H' \text{ is planar}]\}$.

Every element of **OBS** is minimal:
if $H \in \mathbf{OBS}$ then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in \mathbf{OBS}$ $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H : H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:
if $H \in$ **OBS** then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in$ **OBS** $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

So the elements of **OBS** are an antichain.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H\colon H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:
if $H \in$ **OBS** then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in$ **OBS** $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

So the elements of **OBS** are an antichain.
Hence, by GMT, **OBS** is finite.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H : H$ not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:
if $H \in$ **OBS** then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in$ **OBS** $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

So the elements of **OBS** are an antichain.
Hence, by GMT, **OBS** is finite.

1) We have shown there exists a finite obstruction set.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H$: H not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:
if $H \in$ **OBS** then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in$ **OBS** $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

So the elements of **OBS** are an antichain.
Hence, by GMT, **OBS** is finite.

1) We have shown there exists a finite obstruction set.

2) Proof was nonconst. The proof does not help us find **OBS**.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H$: H not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:
if $H \in$ **OBS** then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in$ **OBS** $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

So the elements of **OBS** are an antichain.
Hence, by GMT, **OBS** is finite.

1) We have shown there exists a finite obstruction set.

2) Proof was nonconst. The proof does not help us find **OBS**.

3) The proof cannot be made constructive.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H$: H not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:
if $H \in$ **OBS** then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in$ **OBS** $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

So the elements of **OBS** are an antichain.
Hence, by GMT, **OBS** is finite.

1) We have shown there exists a finite obstruction set.

2) Proof was nonconst. The proof does not help us find **OBS**.

3) The proof cannot be made constructive. Proven by Gasarch.

Obstruction Sets Are Finite

We need that **OBS** is finite. Recall
OBS $= \{H$: H not planar and $(\forall H' \prec_{\mathrm{m}} H)[H'$ is planar$]\}$.

Every element of **OBS** is minimal:
if $H \in$ **OBS** then there is no $H' \prec_{\mathrm{m}} H$ that is nonplanar.

Hence for all $H, H' \in$ **OBS** $H \not\preceq_{\mathrm{m}} H'$ and $H' \not\preceq_{\mathrm{m}} H$.

So the elements of **OBS** are an antichain.
Hence, by GMT, **OBS** is finite.

1) We have shown there exists a finite obstruction set.

2) Proof was nonconst. The proof does not help us find **OBS**.

3) The proof cannot be made constructive. Proven by Gasarch.

4) There may be other finite obstruction sets.

Contrast Wagner and GMT

Contrast Wagner and GMT

Wagner: The obstruction set for planar is $\{K_{3,3}, K_5\}$.

Contrast Wagner and GMT

Wagner: The obstruction set for planar is $\{K_{3,3}, K_5\}$.

GMT: The set of planar graphs has a finite obstruction set.

Contrast Wagner and GMT

Wagner: The obstruction set for planar is $\{K_{3,3}, K_5\}$.

GMT: The set of planar graphs has a finite obstruction set.

The result from GMT is weaker; however it generalizes.

Contrast Wagner and GMT

Wagner: The obstruction set for planar is $\{K_{3,3}, K_5\}$.

GMT: The set of planar graphs has a finite obstruction set.

The result from GMT is weaker; however it generalizes.

We will see this later and "apply" it.

Using the Obstruction Set

Using the Obstruction Set

Robertson & Seymour proved the following:

Using the Obstruction Set

Robertson & Seymour proved the following:

Lemma Fix H.

Using the Obstruction Set

Robertson & Seymour proved the following:

Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Using the Obstruction Set

Robertson & Seymour proved the following:
Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Thm Planarity is in $O(n^3)$.

Using the Obstruction Set

Robertson & Seymour proved the following:
Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Thm Planarity is in $O(n^3)$.
Let **OBS** be an obstructions set for Planarity.

Using the Obstruction Set

Robertson & Seymour proved the following:
Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Thm Planarity is in $O(n^3)$.
Let **OBS** be an obstructions set for Planarity.
Here is an $O(n^3)$ algorithm for planarity

Using the Obstruction Set

Robertson & Seymour proved the following:
Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Thm Planarity is in $O(n^3)$.
Let **OBS** be an obstructions set for Planarity.
Here is an $O(n^3)$ algorithm for planarity
1) Input G

Using the Obstruction Set

Robertson & Seymour proved the following:
Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Thm Planarity is in $O(n^3)$.
Let **OBS** be an obstructions set for Planarity.
Here is an $O(n^3)$ algorithm for planarity
1) Input G
2) For all $H \in$ **OBS** test $H \preceq_{\mathrm{m}} G$.

Using the Obstruction Set

Robertson & Seymour proved the following:
Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Thm Planarity is in $O(n^3)$.
Let **OBS** be an obstructions set for Planarity.
Here is an $O(n^3)$ algorithm for planarity
1) Input G
2) For all $H \in$ **OBS** test $H \preceq_{\mathrm{m}} G$.
3) If $\exists H \in$ **OBS** with $H \preceq_{\mathrm{m}} G$ then output NO.

Using the Obstruction Set

Robertson & Seymour proved the following:
Lemma Fix H. The set $\{G : H \preceq_{\mathrm{m}} G\}$ is in $O(n^3)$ times.

Thm Planarity is in $O(n^3)$.
Let **OBS** be an obstructions set for Planarity.
Here is an $O(n^3)$ algorithm for planarity
1) Input G
2) For all $H \in$ **OBS** test $H \preceq_{\mathrm{m}} G$.
3) If $\exists H \in$ **OBS** with $H \preceq_{\mathrm{m}} G$ then output NO.
4) If $\forall H \in$ **OBS** $H \not\preceq_{\mathrm{m}} G$ then output YES.

Obstruction Sets For Classes Closed Downward Under Minor

What Did We Use About Planarity?

What Did We Use About Planarity?

We used that **Planarity** is closed downward under $\preceq_{\mathrm{m}}$.

What Did We Use About Planarity?

We used that **Planarity** is closed downward under $\preceq_{\mathrm{m}}$.

Def $(X, \preceq)$ be a wqo, $Y \subseteq X$. Y is **closed downward under** $\preceq$ if,

What Did We Use About Planarity?

We used that **Planarity** is closed downward under $\preceq_{\mathrm{m}}$.

Def $(X, \preceq)$ be a wqo, $Y \subseteq X$. Y is **closed downward under** $\preceq$ if,

$$(\forall x \in X)(\forall y \in Y)[x \preceq y \implies x \in Y].$$

What Did We Use About Planarity?

We used that **Planarity** is closed downward under $\preceq_{\mathrm{m}}$.

Def $(X, \preceq)$ be a wqo, $Y \subseteq X$. Y is **closed downward under** $\preceq$ if,

$$(\forall x \in X)(\forall y \in Y)[x \preceq y \implies x \in Y].$$

We used that for a fixed H testing if $H \preceq_{\mathrm{m}} G$ takes $O(n^3)$ time.

What Did We Use About Planarity?

We used that **Planarity** is closed downward under $\preceq_{\mathrm{m}}$.

Def $(X, \preceq)$ be a wqo, $Y \subseteq X$. Y is **closed downward under** $\preceq$ if,

$$(\forall x \in X)(\forall y \in Y)[x \preceq y \implies x \in Y].$$

We used that for a fixed H testing if $H \preceq_{\mathrm{m}} G$ takes $O(n^3)$ time.

See next slide for what we can show in general.

General Theorem

General Theorem

Thm Let X be the set of all graphs. Let Y be a set of graphs that is closed downward closed under minor. Then Y is in $O(n^3)$

Sketch of Proof

General Theorem

Thm Let X be the set of all graphs. Let Y be a set of graphs that is closed downward closed under minor. Then Y is in $O(n^3)$

Sketch of Proof

1) $\exists$ **OBS**, a finite obs set for Y.

General Theorem

Thm Let X be the set of all graphs. Let Y be a set of graphs that is closed downward closed under minor. Then Y is in $O(n^3)$

Sketch of Proof

1) $\exists$ **OBS**, a finite obs set for Y.

2) Algorithm: Given G, for every $H \in$ **OBS**, test $H \preceq_{\mathrm{m}} G$.
If $(\exists H \in \mathbf{OBS})[H \preceq_{\mathrm{m}} G]$ then $G \notin Y$.
If $(\forall H \in \mathbf{OBS})[H \not\preceq_{\mathrm{m}} G]$ then $G \in Y$.

What the General Theorem Can Be Used For

What the General Theorem Can Be Used For

1) Allows us to show problems are in P, even just cubic time!

What the General Theorem Can Be Used For

1) Allows us to show problems are in P, even just cubic time!

2) (Whiggish) allows us to show problems are in FPT.

What the General Theorem Can Be Used For

1) Allows us to show problems are in P, even just cubic time!

2) (Whiggish) allows us to show problems are in FPT.

We look at Vertex Cover for Fixed k as our main example

Vertex Cover For Fixed k

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Recall

The Vertex Cover Problem

Def If G is a graph then a **vertex cover** is a set of vertices such that every edge has at least one endpoint in that set.

Recall

$$\mathrm{VC}_k = \{G : G \text{ has a vertex cover of size } k \}.$$

Vertex Cover

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).
KeyVC_k is closed under minor. AH-HA!

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).
KeyVC_k is closed under minor. AH-HA!

So there exists a finite obstruction set $O_1, \ldots, O_{f(k)}$.

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).
KeyVC_k is closed under minor. AH-HA!

So there exists a finite obstruction set $O_1, \ldots, O_{f(k)}$.

Algorithm: Given G test if any of $O_1, \ldots, O_{f(k)}$ are a minor of G. (Each test takes $O(n^3)$ time.)

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).
Key VC_k is closed under minor. AH-HA!

So there exists a finite obstruction set $O_1, \ldots, O_{f(k)}$.

Algorithm: Given G test if any of $O_1, \ldots, O_{f(k)}$ are a minor of G. (Each test takes $O(n^3)$ time.)

If $(\forall 1 \leq i \leq f(k))[O_i \not\preceq_{\mathrm{m}} G]$ then output YES, $G \in \mathrm{VC}_k$

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).
Key VC_k is closed under minor. AH-HA!

So there exists a finite obstruction set $O_1, \ldots, O_{f(k)}$.

Algorithm: Given G test if any of $O_1, \ldots, O_{f(k)}$ are a minor of G. (Each test takes $O(n^3)$ time.)

If $(\forall 1 \leq i \leq f(k))[O_i \not\preceq_{\mathrm{m}} G]$ then output YES, $G \in \mathrm{VC}_k$

If $(\exists 1 \leq i \leq f(k))[O_i \preceq_{\mathrm{m}} G]$ then output NO, $G \notin \mathrm{VC}_k$

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).
Key VC_k is closed under minor. AH-HA!

So there exists a finite obstruction set $O_1, \ldots, O_{f(k)}$.

Algorithm: Given G test if any of $O_1, \ldots, O_{f(k)}$ are a minor of G. (Each test takes $O(n^3)$ time.)

If $(\forall 1 \leq i \leq f(k))[O_i \not\preceq_{\mathrm{m}} G]$ then output YES, $G \in \mathrm{VC}_k$

If $(\exists 1 \leq i \leq f(k))[O_i \preceq_{\mathrm{m}} G]$ then output NO, $G \notin \mathrm{VC}_k$

This algorithm takes time $O(f(k)n^3)$.

Vertex Cover

We first show VC_k is in $O(f(k)n^3)$ time (we find f later).
KeyVC_k is closed under minor. AH-HA!

So there exists a finite obstruction set $O_1, \ldots, O_{f(k)}$.

Algorithm: Given G test if any of $O_1, \ldots, O_{f(k)}$ are a minor of G. (Each test takes $O(n^3)$ time.)

If $(\forall 1 \leq i \leq f(k))[O_i \not\preceq_{\mathrm{m}} G]$ then output YES, $G \in \mathrm{VC}_k$

If $(\exists 1 \leq i \leq f(k))[O_i \preceq_{\mathrm{m}} G]$ then output NO, $G \notin \mathrm{VC}_k$

This algorithm takes time $O(f(k)n^3)$.

Done. Is this a good algorithm?

What is Good?

What is Good?

Def An algorithm for VC_k is **Fixed Parameter Tractable (FPT)** if it runs in time $O(f(k)n^{O(1)})$ for some function $f(k)$.

What is Good?

Def An algorithm for VC_k is **Fixed Parameter Tractable (FPT)** if it runs in time $O(f(k)n^{O(1)})$ for some function $f(k)$.

Even if $f(k) = 2^k$ this is an improvement over $O(n^k)$ for med values of k since k is in the mult constant, not in the exponent.

What is Good?

Def An algorithm for VC_k is **Fixed Parameter Tractable (FPT)** if it runs in time $O(f(k)n^{O(1)})$ for some function $f(k)$.

Even if $f(k) = 2^k$ this is an improvement over $O(n^k)$ for med values of k since k is in the mult constant, not in the exponent.

We will revisit the practicality later.

What is Good?

Def An algorithm for VC_k is **Fixed Parameter Tractable (FPT)** if it runs in time $O(f(k)n^{O(1)})$ for some function $f(k)$.

Even if $f(k) = 2^k$ this is an improvement over $O(n^k)$ for med values of k since k is in the mult constant, not in the exponent.

We will revisit the practicality later.

We will use the term FPT for any problem that has a parameter k such that, if k is fixed, there is an $O(f(k)n^{O(1)})$ algorithm.

What is Good?

Def An algorithm for VC_k is **Fixed Parameter Tractable (FPT)** if it runs in time $O(f(k)n^{O(1)})$ for some function $f(k)$.

Even if $f(k) = 2^k$ this is an improvement over $O(n^k)$ for med values of k since k is in the mult constant, not in the exponent.

We will revisit the practicality later.

We will use the term FPT for any problem that has a parameter k such that, if k is fixed, there is an $O(f(k)n^{O(1)})$ algorithm.

This is one of the tools to cope with NP-completeness.

Can We Get The Code?

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

1) I added some more actual math to the skit.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

1) I added some more actual math to the skit.

2) We will switch roles! I will sit next to Ryan and you come up here.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

1) I added some more actual math to the skit.

2) We will switch roles! I will sit next to Ryan and you come up here.

Ryan's Friend So you'll be Ryan's friend and I'll be you?

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

1) I added some more actual math to the skit.

2) We will switch roles! I will sit next to Ryan and you come up here.

Ryan's Friend So you'll be Ryan's friend and I'll be you?

Bill Yes.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

1) I added some more actual math to the skit.

2) We will switch roles! I will sit next to Ryan and you come up here.

Ryan's Friend So you'll be Ryan's friend and I'll be you?

Bill Yes. Though I would like to think I am also Ryan's friend.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

1) I added some more actual math to the skit.

2) We will switch roles! I will sit next to Ryan and you come up here.

Ryan's Friend So you'll be Ryan's friend and I'll be you?

Bill Yes. Though I would like to think I am also Ryan's friend. So, in what follows you read where it says **Bill**, and I'll read where it says **Ryan's Friend.**

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Are we going to do the same skit we did Tuesday. That would be boring.

Bill There are two reasons why it won't be boring.

1) I added some more actual math to the skit.

2) We will switch roles! I will sit next to Ryan and you come up here.

Ryan's Friend So you'll be Ryan's friend and I'll be you?

Bill Yes. Though I would like to think I am also Ryan's friend. So, in what follows you read where it says **Bill**, and I'll read where it says **Ryan's Friend.**

Ryan's Friend Great!

Can We Get The Code?

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Ryan's Friend Are there some small value of k where we **do** know the obstruction set and hence we can run the algorithm.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Ryan's Friend Are there some small value of k where we **do** know the obstruction set and hence we can run the algorithm.

Bill Good question! Indeed, for $k = 1$ through 7 we actually know the obstruction set.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Ryan's Friend Are there some small value of k where we **do** know the obstruction set and hence we can run the algorithm.

Bill Good question! Indeed, for $k = 1$ through 7 we actually know the obstruction set.

Ryan's Friend Great! How big are the obstruction sets for $k = 1$ through 7. If they are small enough then the algorithm using them is useable.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Ryan's Friend Are there some small value of k where we **do** know the obstruction set and hence we can run the algorithm.

Bill Good question! Indeed, for $k = 1$ through 7 we actually know the obstruction set.

Ryan's Friend Great! How big are the obstruction sets for $k = 1$ through 7. If they are small enough then the algorithm using them is useable.

Bill Useable?

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Ryan's Friend Are there some small value of k where we **do** know the obstruction set and hence we can run the algorithm.

Bill Good question! Indeed, for $k = 1$ through 7 we actually know the obstruction set.

Ryan's Friend Great! How big are the obstruction sets for $k = 1$ through 7. If they are small enough then the algorithm using them is useable.

Bill Useable? In this course?

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Ryan's Friend Are there some small value of k where we **do** know the obstruction set and hence we can run the algorithm.

Bill Good question! Indeed, for $k = 1$ through 7 we actually know the obstruction set.

Ryan's Friend Great! How big are the obstruction sets for $k = 1$ through 7. If they are small enough then the algorithm using them is useable.

Bill Useable? In this course? Uh,..., we'll see.

Can We Get The Code?

Bill I have shown there exists an $O(n^3)$ algorithm for VC_{1000}.

Ryan's Friend Great! Show it to me so I can code it up.

Bill The proof was nonconstructive. I don't **have** an algorithm.

Ryan's Friend Are there some small value of k where we **do** know the obstruction set and hence we can run the algorithm.

Bill Good question! Indeed, for $k = 1$ through 7 we actually know the obstruction set.

Ryan's Friend Great! How big are the obstruction sets for $k = 1$ through 7. If they are small enough then the algorithm using them is useable.

Bill Useable? In this course? Uh,..., we'll see.
The next slide has a table of sizes of the obstruction sets.

Sizes of the Obstruction Sets

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$
I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$
I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$.

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$
I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$. Really!?

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$ I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$. Really!?

Bill Its worse. For $k \geq 8$ we don't know the obstruction set.

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$ I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$. Really!?

Bill Its worse. For $k \geq 8$ we don't know the obstruction set.

Ryan's Friend Surely there is some way to get a const alg?

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$ I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$. Really!?

Bill Its worse. For $k \geq 8$ we don't know the obstruction set.

Ryan's Friend Surely there is some way to get a const alg?

Bill Yes.

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$ I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$. Really!?

Bill Its worse. For $k \geq 8$ we don't know the obstruction set.

Ryan's Friend Surely there is some way to get a const alg?

Bill Yes. And stop calling me Shirley.

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$ I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$. Really!?

Bill Its worse. For $k \geq 8$ we don't know the obstruction set.

Ryan's Friend Surely there is some way to get a const alg?

Bill Yes. And stop calling me Shirley.

Ryan's Friend I'll stop calling you Shirley if you stop using that joke.

Sizes of the Obstruction Sets

Bill Let $f(k)$ be the size of the smallest obstruction set for VC_k. Then we have this table (Bill does not read the table.)

k	1	2	3	4	5	6	7
$f(k)$	2	4	8	18	56	260	2250

Bill Known: $f(k) = 2^{O(k)}$. I think it is known that $f(k) = \Omega(k)$ I believe that ChatGPT believes that people believe $f(k) = 2^{\Theta(k)}$.

Ryan's Friend $f(k) = 2250$. Really!?

Bill Its worse. For $k \geq 8$ we don't know the obstruction set.

Ryan's Friend Surely there is some way to get a const alg?

Bill Yes. And stop calling me Shirley.

Ryan's Friend I'll stop calling you Shirley if you stop using that joke.

Bill Deal. Plus, on the next slide, I have the constructive algorithm.

Terminology for Constructive Proofs

Terminology for Constructive Proofs

If I write

Terminology for Constructive Proofs

If I write

There is an $O(n^3)$ Algorithm for Blah.

Terminology for Constructive Proofs

If I write

There is an $O(n^3)$ Algorithm for Blah.

then that just means the algorithm exists, and not that I have the code or even could have the code.

Terminology for Constructive Proofs

If I write

There is an $O(n^3)$ Algorithm for Blah.

then that just means the algorithm exists, and not that I have the code or even could have the code.

I proved that, for $k \in \mathbb{N}$,

There is an $O(f(k)n^3)$ Algorithm for VC_k.

Terminology for Constructive Proofs

If I write

There is an $O(n^3)$ Algorithm for Blah.

then that just means the algorithm exists, and not that I have the code or even could have the code.

I proved that, for $k \in \mathbb{N}$,

There is an $O(f(k)n^3)$ Algorithm for VC_k.

The proof **did not** give me enough information to write the code.

Terminology for Constructive Proofs

If I write

There is an $O(n^3)$ Algorithm for Blah.

then that just means the algorithm exists, and not that I have the code or even could have the code.

I proved that, for $k \in \mathbb{N}$,

There is an $O(f(k)n^3)$ Algorithm for VC_k.

The proof **did not** give me enough information to write the code.

The proof **did not** even tell me what $f(k)$ is!.

Terminology for Constructive Proofs

If I write

There is an $O(n^3)$ Algorithm for Blah.

then that just means the algorithm exists, and not that I have the code or even could have the code.

I proved that, for $k \in \mathbb{N}$,

There is an $O(f(k)n^3)$ Algorithm for VC_k.

The proof **did not** give me enough information to write the code.

The proof **did not** even tell me what $f(k)$ is!.

The statement

Terminology for Constructive Proofs

If I write

There is an $O(n^3)$ Algorithm for Blah.

then that just means the algorithm exists, and not that I have the code or even could have the code.

I proved that, for $k \in \mathbb{N}$,

There is an $O(f(k)n^3)$ Algorithm for VC_k.

The proof **did not** give me enough information to write the code.

The proof **did not** even tell me what $f(k)$ is!.

The statement

We have an $O(n^2)$ alg for BLAH

means that we have the code.

Algorithms We Have and Will Need

Algorithms We Have and Will Need

Algorithm 1 We have an $O(|E|) = O(n^2)$ algorithm for:

Algorithms We Have and Will Need

Algorithm 1 We have an $O(|E|) = O(n^2)$ algorithm for:

Given a graph $G = (V, E)$ and a set $U \subseteq V$, determine if U is a vertex cover.

Algorithms We Have and Will Need

Algorithm 1 We have an $O(|E|) = O(n^2)$ algorithm for:

Given a graph $G = (V, E)$ and a set $U \subseteq V$, determine if U is a vertex cover.

Visit every edge and see if one of the endpoints is in U. If always yes, YES, if ever no, NO.

Algorithms We Have and Will Need

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin \mathbf{VC}_k$

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin\notin \mathbf{VC_k}$

$G \in \mathbf{VC_k} - \mathbf{VC_{k-1}}$

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin\notin \mathbf{VC_k}$

$G \in \mathbf{VC_k} - \mathbf{VC_{k-1}}$

$G \in \mathbf{VC_{k-1}} - \mathbf{VC_{k-2}}$

$\vdots$

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin\notin \mathbf{VC}_k$

$G \in \mathbf{VC}_k - \mathbf{VC}_{k-1}$

$G \in \mathbf{VC}_{k-1} - \mathbf{VC}_{k-2}$

$\vdots$

$G \in \mathbf{VC}_2 - \mathbf{VC}_1$.

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin \mathbf{VC_k}$

$G \in \mathbf{VC_k} - \mathbf{VC_{k-1}}$

$G \in \mathbf{VC_{k-1}} - \mathbf{VC_{k-2}}$

$\vdots$

$G \in \mathbf{VC_2} - \mathbf{VC_1}$.

Check all $\binom{V}{1}$, $\binom{V}{2}$, ..., $\binom{V}{k}$ and use Algorithm 1.

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin\notin \mathbf{VC_k}$

$G \in \mathbf{VC_k} - \mathbf{VC_{k-1}}$

$G \in \mathbf{VC_{k-1}} - \mathbf{VC_{k-2}}$

$\vdots$

$G \in \mathbf{VC_2} - \mathbf{VC_1}$.

Check all $\binom{V}{1}$, $\binom{V}{2}$, ..., $\binom{V}{k}$ and use Algorithm 1.
Why state this Algorithm:

(1) to emphasize that **we have** such an algorithm, and

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin\notin \mathbf{VC_k}$

$G \in \mathbf{VC_k} - \mathbf{VC_{k-1}}$

$G \in \mathbf{VC_{k-1}} - \mathbf{VC_{k-2}}$

$\vdots$

$G \in \mathbf{VC_2} - \mathbf{VC_1}$.

Check all $\binom{V}{1}$, $\binom{V}{2}$, ..., $\binom{V}{k}$ and use Algorithm 1.
Why state this Algorithm:

(1) to emphasize that **we have** such an algorithm, and
(2) we will use it on graphs with small n ($n \leq 10^{10}$ for example).

Algorithms We Have and Will Need

Algorithm 2 We have an $O(n^k)$ algorithm for:

Given G determine which of the following holds:

$G \notin \mathbf{VC_k}$

$G \in \mathbf{VC_k} - \mathbf{VC_{k-1}}$

$G \in \mathbf{VC_{k-1}} - \mathbf{VC_{k-2}}$

$\vdots$

$G \in \mathbf{VC_2} - \mathbf{VC_1}$.

Check all $\binom{V}{1}$, $\binom{V}{2}$, ..., $\binom{V}{k}$ and use Algorithm 1.
Why state this Algorithm:

(1) to emphasize that **we have** such an algorithm, and
(2) we will use it on graphs with small n ($n \leq 10^{10}$ for example).

Note that Alg 2 only gives us the **size** of the min vertex cov.

More Algorithms We Will Need

More Algorithms We Will Need

Algorithm 3 H is a graph. **We have** an $O(g(H)n^3)$ algorithm for:

More Algorithms We Will Need

Algorithm 3 H is a graph. **We have** an $O(g(H)n^3)$ algorithm for:

Given G, determine if $H \preceq_{\mathrm{m}} G$.

More Algorithms We Will Need

Algorithm 3 H is a graph. **We have** an $O(g(H)n^3)$ algorithm for:

Given G, determine if $H \preceq_{\mathrm{m}} G$.

This was proven by Robertson and Seymour and is difficult.

More Algorithms We Will Need

Algorithm 3 H is a graph. **We have** an $O(g(H)n^3)$ algorithm for:

Given G, determine if $H \preceq_{\mathrm{m}} G$.

This was proven by Robertson and Seymour and is difficult.

Given H we can obtain the code very quickly.

More Algorithms We Will Need

Algorithm 3 H is a graph. **We have** an $O(g(H)n^3)$ algorithm for:

Given G, determine if $H \preceq_{\mathrm{m}} G$.

This was proven by Robertson and Seymour and is difficult.

Given H we can obtain the code very quickly.

Hence we can use this in an algorithm that itself generates H.

More Algorithms We Will Need

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given *G* determine if *G* has a vertex covers of size $\leq k$ (we've called this VC_k).

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given G determine if G has a vertex covers of size $\leq k$ (we've called this VC_k).

($\forall H \in \mathrm{OBS}_k$) determine if $H \preceq_{\mathrm{m}} G$ (Algorithm 3).

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given G determine if G has a vertex covers of size $\leq k$ (we've called this VC_k).

$(\forall H \in \mathrm{OBS}_k)$ determine if $H \preceq_{\mathrm{m}} G$ (Algorithm 3).

If $(\exists H \in \mathrm{OBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO.

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given G determine if G has a vertex covers of size $\leq k$ (we've called this VC_k).

$(\forall H \in \mathrm{OBS}_k)$ determine if $H \preceq_{\mathrm{m}} G$ (Algorithm 3).

If $(\exists H \in \mathrm{OBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO.
If $(\forall H \in \mathrm{OBS}_k)[H \not\preceq_{\mathrm{m}} G]$ then output YES.

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given G determine if G has a vertex covers of size $\leq k$ (we've called this VC_k).

$(\forall H \in \mathrm{OBS}_k)$ determine if $H \preceq_{\mathrm{m}} G$ (Algorithm 3).

If $(\exists H \in \mathrm{OBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO.
If $(\forall H \in \mathrm{OBS}_k)[H \not\preceq_{\mathrm{m}} G]$ then output YES.

Algorithm 5 If **we have** $\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ then **we have** an FPT algorithm for:

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given G determine if G has a vertex covers of size $\leq k$ (we've called this VC_k).

$(\forall H \in \mathrm{OBS}_k)$ determine if $H \preceq_{\mathrm{m}} G$ (Algorithm 3).

If $(\exists H \in \mathrm{OBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO.
If $(\forall H \in \mathrm{OBS}_k)[H \not\preceq_{\mathrm{m}} G]$ then output YES.

Algorithm 5 If **we have** $\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ then **we have** an FPT algorithm for:

Given a graph G find if it has a vertex cov of size k and if it does then output the actual vertex cov.

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given G determine if G has a vertex covers of size $\leq k$ (we've called this VC_k).

$(\forall H \in \mathrm{OBS}_k)$ determine if $H \preceq_{\mathrm{m}} G$ (Algorithm 3).

If $(\exists H \in \mathrm{OBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO.
If $(\forall H \in \mathrm{OBS}_k)[H \not\preceq_{\mathrm{m}} G]$ then output YES.

Algorithm 5 If **we have** $\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ then **we have** an FPT algorithm for:

Given a graph G find if it has a vertex cov of size k and if it does then output the actual vertex cov.

Use: have FPT algorithms for $\mathrm{VC}_1, \ldots, \mathrm{VC}_k$.

More Algorithms We Will Need

Algorithm 4 If **we have** OBS_k, then **we have** an FPT alg for:

Given G determine if G has a vertex covers of size $\leq k$ (we've called this VC_k).

$(\forall H \in \mathrm{OBS}_k)$ determine if $H \preceq_{\mathrm{m}} G$ (Algorithm 3).

If $(\exists H \in \mathrm{OBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO.
If $(\forall H \in \mathrm{OBS}_k)[H \not\preceq_{\mathrm{m}} G]$ then output YES.

Algorithm 5 If **we have** $\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ then **we have** an FPT algorithm for:

Given a graph G find if it has a vertex cov of size k and if it does then output the actual vertex cov.

Use: have FPT algorithms for $\mathrm{VC}_1, \ldots, \mathrm{VC}_k$. This may be a HW.

One More Algorithm

One More Algorithm

Let $H_1, H_2, \ldots$ be a list of all graphs in increasing order of size (ties broken arbitrarily).

One More Algorithm

Let $H_1, H_2, \ldots$ be a list of all graphs in increasing order of size (ties broken arbitrarily).

Algorithm 6 We have an algorithm that:
on input i, output H_i. Don't care about time.

One More Algorithm

Let $H_1, H_2, \ldots$ be a list of all graphs in increasing order of size (ties broken arbitrarily).

Algorithm 6 We have an algorithm that:
on input i, output H_i. Don't care about time.

I WILL HAND OUT A SHEET WITH Statements of Algorithms 1,2,3,4,5,6.

Concepts Needed For The Constructive Algorithm

Concepts Needed For The Constructive Algorithm

Assume that we are trying to find out if $G \in \mathrm{VC}_k$ where $G = (V, E)$.

Concepts Needed For The Constructive Algorithm

Assume that we are trying to find out if $G \in \mathrm{VC}_k$ where $G = (V, E)$.

if we find an $H \notin \mathrm{VC}_k$, $H \preceq_{\mathrm{m}} G$, then **we know** $G \notin \mathrm{VC}_k$.

Concepts Needed For The Constructive Algorithm

Assume that we are trying to find out if $G \in \mathrm{VC}_k$ where $G = (V, E)$.

if we find an $H \notin \mathrm{VC}_k$, $H \preceq_{\mathrm{m}} G$, then **we know** $G \notin \mathrm{VC}_k$.

if we find $U \subseteq V$, U is a vertex cov of size k, then **we know** $G \in \mathrm{VC}_k$.

Concepts Needed For The Constructive Algorithm

Assume that we are trying to find out if $G \in \mathrm{VC}_k$ where $G = (V, E)$.

if we find an $H \notin \mathrm{VC}_k$, $H \preceq_{\mathrm{m}} G$, then **we know** $G \notin \mathrm{VC}_k$.

if we find $U \subseteq V$, U is a vertex cov of size k, then **we know** $G \in \mathrm{VC}_k$.

Our constructive algorithm will, given G, achieve one of those two.

Constructive Algorithm for VC_k

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)
    if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)
    if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k
    if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

if $H_i \in \mathrm{VC}_j - \mathrm{VC}_{j-1}$ then put H_i into PHOBS_{j-1}.

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

if $H_i \in \mathrm{VC}_j - \mathrm{VC}_{j-1}$ then put H_i into PHOBS_{j-1}.

$\vdots$

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

if $H_i \in \mathrm{VC}_j - \mathrm{VC}_{j-1}$ then put H_i into PHOBS_{j-1}.

$\vdots$

if $H_i \in \mathrm{VC}_2 - \mathrm{VC}_1$ then put H into PHOBS_1.

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

if $H_i \in \mathrm{VC}_j - \mathrm{VC}_{j-1}$ then put H_i into PHOBS_{j-1}.

$\vdots$

if $H_i \in \mathrm{VC}_2 - \mathrm{VC}_1$ then put H into PHOBS_1.

(Comment: Is PHOBS_j an obs set for VC_j?

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

if $H_i \in \mathrm{VC}_j - \mathrm{VC}_{j-1}$ then put H_i into PHOBS_{j-1}.

$\vdots$

if $H_i \in \mathrm{VC}_2 - \mathrm{VC}_1$ then put H into PHOBS_1.

(Comment: Is PHOBS_j an obs set for VC_j? We will pretend it is.)

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

if $H_i \in \mathrm{VC}_j - \mathrm{VC}_{j-1}$ then put H_i into PHOBS_{j-1}.

$\vdots$

if $H_i \in \mathrm{VC}_2 - \mathrm{VC}_1$ then put H into PHOBS_1.

(Comment: Is PHOBS_j an obs set for VC_j? We will pretend it is.)
(Comment: Next slide is what we do after processing an i.)

Constructive Algorithm for VC_k

1) Input $G = (V, E)$. Let $n = |V|$. Maintain sets PHOBS_1 PHOBS_2, ..., PHOBS_k, initially $\emptyset$.
(For all $1 \leq i \leq k$, PHOBS_i is a possibly phony OBS_i.)

2) **For** $i = 1$ to ∞ (Algorithm 6, Algorithm 2)

if $H_i \notin \mathrm{VC}_k$ then put H_i into PHOBS_k

if $H_i \in \mathrm{VC}_k - \mathrm{VC}_{k-1}$ then put H_i into PHOBS_{k-1}.

$\vdots$

if $H_i \in \mathrm{VC}_j - \mathrm{VC}_{j-1}$ then put H_i into PHOBS_{j-1}.

$\vdots$

if $H_i \in \mathrm{VC}_2 - \mathrm{VC}_1$ then put H into PHOBS_1.

(Comment: Is PHOBS_j an obs set for VC_j? We will pretend it is.)
(Comment: Next slide is what we do after processing an i.)
(Comment: Still in the For Loop)

Pretending!

Pretending!

This is a comment, not part of the code.

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

I run it on ϕ.

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

I run it on ϕ.

If it says NO then I don't trust it.

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

I run it on ϕ.

If it says NO then I don't trust it.

If it says YES then I use it to find a sat assignment $\vec{v}$ for ϕ

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

I run it on ϕ.

If it says NO then I don't trust it.

If it says YES then I use it to find a sat assignment $\vec{v}$ for ϕ

I then compute $\phi(\vec{v})$.

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

I run it on ϕ.

If it says NO then I don't trust it.

If it says YES then I use it to find a sat assignment $\vec{v}$ for ϕ

I then compute $\phi(\vec{v})$. If its T then I output YES

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

I run it on ϕ.

If it says NO then I don't trust it.

If it says YES then I use it to find a sat assignment $\vec{v}$ for ϕ

I then compute $\phi(\vec{v})$. If its T then I output YES

and I am confident of my output!

Pretending!

This is a comment, not part of the code.

How do we use a possibly false algorithm?

By using it to provide a **witness for membership** and then **verifying** the witness.

A student has an possibly-false algorithm for SAT

I run it on ϕ.

If it says NO then I don't trust it.

If it says YES then I use it to find a sat assignment $\vec{v}$ for ϕ

I then compute $\phi(\vec{v})$. If its T then I output YES
and I am confident of my output!

We will achieve certainty for both $G \in \mathrm{VC}_k$ and $G \notin \mathrm{VC}_k$.

Might Find $G \notin \mathrm{VC}_k$ (cont)

Might Find $G \notin \mathrm{VC}_k$ (cont)

$\forall H \in \mathrm{PHOBS}_k$ determine if $H \preceq_{\mathrm{m}} G$ (Use Alg 3).

Might Find $G \notin \mathrm{VC}_k$ (cont)

$\forall H \in \mathrm{PHOBS}_k$ determine if $H \preceq_{\mathrm{m}} G$ (Use Alg 3).

If $(\exists H \in \mathrm{PHOBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO and stop.

Might Find $G \notin \mathrm{VC}_k$ (cont)

$\forall H \in \mathrm{PHOBS}_k$ determine if $H \preceq_{\mathrm{m}} G$ (Use Alg 3).

If $(\exists H \in \mathrm{PHOBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO and stop.

(Comment: Since $H \notin \mathrm{VC}_k$ and $H \preceq_{\mathrm{m}} G$, we **know** $G \notin \mathrm{VC}_k$.)

Might Find $G \notin \mathrm{VC}_k$ (cont)

$\forall H \in \mathrm{PHOBS}_k$ determine if $H \preceq_{\mathrm{m}} G$ (Use Alg 3).

If $(\exists H \in \mathrm{PHOBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO and stop.

(Comment: Since $H \notin \mathrm{VC}_k$ and $H \preceq_{\mathrm{m}} G$, we **know** $G \notin \mathrm{VC}_k$.)

(Comment: If we have $(\forall H \in \mathrm{PHOBS})[H \not\preceq_{\mathrm{m}} G]$ then we don't know anything since PHOBS's might not be OBS's.)

Might Find $G \notin \mathrm{VC}_k$ (cont)

$\forall H \in \mathrm{PHOBS}_k$ determine if $H \preceq_{\mathrm{m}} G$ (Use Alg 3).

If $(\exists H \in \mathrm{PHOBS}_k)[H \preceq_{\mathrm{m}} G]$ then output NO and stop.

(Comment: Since $H \notin \mathrm{VC}_k$ and $H \preceq_{\mathrm{m}} G$, we **know** $G \notin \mathrm{VC}_k$.)

(Comment: If we have $(\forall H \in \mathrm{PHOBS})[H \not\preceq_{\mathrm{m}} G]$ then we don't know anything since PHOBS's might not be OBS's.)

So now what do we do?

Go to the next slide.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Use Algorithm 1 to check if U is a vertex cov of size $\leq k$.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Use Algorithm 1 to check if U is a vertex cov of size $\leq k$.

If Yes then then output YES and Stop.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Use Algorithm 1 to check if U is a vertex cov of size $\leq k$.

If Yes then then output YES and Stop.

(Comment: We **know** that $G \in \mathrm{VC}_k$ since we have a vertex cov of size $\leq k$.)

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Use Algorithm 1 to check if U is a vertex cov of size $\leq k$.

If Yes then then output YES and Stop.

(Comment: We **know** that $G \in \mathrm{VC}_k$ since we have a vertex cov of size $\leq k$.)

Case 2: Algorithm 5 says NO.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Use Algorithm 1 to check if U is a vertex cov of size $\leq k$.

If Yes then then output YES and Stop.

(Comment: We **know** that $G \in \mathrm{VC}_k$ since we have a vertex cov of size $\leq k$.)

Case 2: Algorithm 5 says NO.

We do not know anything.

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Use Algorithm 1 to check if U is a vertex cov of size $\leq k$.

If Yes then then output YES and Stop.

(Comment: We **know** that $G \in \mathrm{VC}_k$ since we have a vertex cov of size $\leq k$.)

Case 2: Algorithm 5 says NO.

We do not know anything.

Got to the next iteration of the For Loop

Might Find $G \in \mathrm{VC}_k$ (cont)

(Comment: We will Pretend $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$ really is $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$. If we find a vertex cover we will verify it—or fail to).)

(Comment: Recall that Algorithm 5 takes $(\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k)$ and uses them to find to **find** the a vertex cov of size $\leq k$ if such exists. We will run with $(\mathrm{PHOBS}_1, \ldots, \mathrm{PHOBS}_k)$.

Run Algorithm 5 using the PHOBS_i's.

Case 1: Algorithm 5 says YES and outputs U.

Use Algorithm 1 to check if U is a vertex cov of size $\leq k$.

If Yes then then output YES and Stop.

(Comment: We **know** that $G \in \mathrm{VC}_k$ since we have a vertex cov of size $\leq k$.)

Case 2: Algorithm 5 says NO.

We do not know anything.

Got to the next iteration of the For Loop

End For

Why Does The Algorithm Work?

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

If the algorithm outputs YES then the algorithm it is correct: it found an actual vertex covers of size k.

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

If the algorithm outputs YES then the algorithm it is correct: it found an actual vertex covers of size k.

Why must the algorithm stop?

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

If the algorithm outputs YES then the algorithm it is correct: it found an actual vertex covers of size k.

Why must the algorithm stop?

$\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ are all **finite**.

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

If the algorithm outputs YES then the algorithm it is correct: it found an actual vertex covers of size k.

Why must the algorithm stop?

$\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ are all **finite**.

All graphs in the OBS_i's appear in (say) the first 10^{10} graphs.

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

If the algorithm outputs YES then the algorithm it is correct: it found an actual vertex covers of size k.

Why must the algorithm stop?

$\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ are all **finite**.

All graphs in the OBS_i's appear in (say) the first 10^{10} graphs.

The **for** loop will have real OBS's somewhere before $i = 10^{10}$

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

If the algorithm outputs YES then the algorithm it is correct: it found an actual vertex covers of size k.

Why must the algorithm stop?

$\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ are all **finite**.

All graphs in the OBS_i's appear in (say) the first 10^{10} graphs.

The **for** loop will have real OBS's somewhere before $i = 10^{10}$

In the first iteration where all of the PHOBS's are really OBS's, the algorithm will halt and be correct.

Why Does The Algorithm Work?

If the algorithm outputs NO then the algorithm is correct: there is a graph $H \notin \mathrm{VC}_k$ such that $H \preceq_{\mathrm{m}} G$, so $G \notin \mathrm{VC}_k$.

If the algorithm outputs YES then the algorithm it is correct: it found an actual vertex covers of size k.

Why must the algorithm stop?

$\mathrm{OBS}_1, \ldots, \mathrm{OBS}_k$ are all **finite**.

All graphs in the OBS_i's appear in (say) the first 10^{10} graphs.

The **for** loop will have real OBS's somewhere before $i = 10^{10}$

In the first iteration where all of the PHOBS's are really OBS's, the algorithm will halt and be correct.

It may stop much earlier than that, and be correct.

Why is The Algorithm FPT

Why is The Algorithm FPT

As noted above the **for** loop will go for at most 10^{10} steps.

Why is The Algorithm FPT

As noted above the **for** loop will go for at most 10^{10} steps.

Hence all of the H_i's have ≤ 100 vertices.

Why is The Algorithm FPT

As noted above the **for** loop will go for at most 10^{10} steps.

Hence all of the H_i's have ≤ 100 vertices.

Within the for loop everything we do is FPT.

Back to Arguing

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT.

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!
First GMT gave us an algorithm that **couldn't** be coded.

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!
First GMT gave us an algorithm that **couldn't** be coded.
Then you obtained an algorithm you **wouldn't** want to code up.

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!
First GMT gave us an algorithm that **couldn't** be coded.
Then you obtained an algorithm you **wouldn't** want to code up.

Bill My shirt agrees with you.

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!
First GMT gave us an algorithm that **couldn't** be coded.
Then you obtained an algorithm you **wouldn't** want to code up.

Bill My shirt agrees with you.

Ryan's Friend So, now what?

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!
First GMT gave us an algorithm that **couldn't** be coded.
Then you obtained an algorithm you **wouldn't** want to code up.

Bill My shirt agrees with you.

Ryan's Friend So, now what?

Bill I agree that the algorithms I presented are

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!
First GMT gave us an algorithm that **couldn't** be coded.
Then you obtained an algorithm you **wouldn't** want to code up.

Bill My shirt agrees with you.

Ryan's Friend So, now what?

Bill I agree that the algorithms I presented are
two tacos shy of a combination plate.

Back to Arguing

Bill I have shown there there is an $O(n^3)$ algorithm for VC_{1000} that you can actual code. So VC_k is FPT. **Yeah!**

Ryan's Friend FPT is bullshit!
First GMT gave us an algorithm that **couldn't** be coded.
Then you obtained an algorithm you **wouldn't** want to code up.

Bill My shirt agrees with you.

Ryan's Friend So, now what?

Bill I agree that the algorithms I presented are

two tacos shy of a combination plate.

But not all is lost. See next slide

Lets End with Good News

Bill Once theorists learned that $\exists$ an FPT algorithm using GMT they found practical FPT algorithms that do not use GMT.

Lets End with Good News

Bill Once theorists learned that $\exists$ an FPT algorithm using GMT they found practical FPT algorithms that do not use GMT.

Ryan's Friend How good?

Lets End with Good News

Bill Once theorists learned that $\exists$ an FPT algorithm using GMT they found practical FPT algorithms that do not use GMT.

Ryan's Friend How good?
Bill There is a simple $O(2^k n^2)$ algorithm. The best known is in $O(kn + 1.274^k)$. Both have very small constants in the O-of. The second one is useable and I could code it up.

Lets End with Good News

Bill Once theorists learned that $\exists$ an FPT algorithm using GMT they found practical FPT algorithms that do not use GMT.

Ryan's Friend How good?
Bill There is a simple $O(2^k n^2)$ algorithm. The best known is in $O(kn + 1.274^k)$. Both have very small constants in the O-of. The second one is useable and I could code it up.

Ryan's Friend You could code it up? In what computer language?

Lets End with Good News

Bill Once theorists learned that $\exists$ an FPT algorithm using GMT they found practical FPT algorithms that do not use GMT.

Ryan's Friend How good?
Bill There is a simple $O(2^k n^2)$ algorithm. The best known is in $O(kn + 1.274^k)$. Both have very small constants in the O-of. The second one is useable and I could code it up.

Ryan's Friend You could code it up? In what computer language?

Bill The language HSS.

Lets End with Good News

Bill Once theorists learned that $\exists$ an FPT algorithm using GMT they found practical FPT algorithms that do not use GMT.

Ryan's Friend How good?
Bill There is a simple $O(2^k n^2)$ algorithm. The best known is in $O(kn + 1.274^k)$. Both have very small constants in the O-of. The second one is useable and I could code it up.

Ryan's Friend You could code it up? In what computer language?

Bill The language HSS.

Ryan's Friend Oh. I don't know that one. Is that some fancy new language that David van Horn made up?

Lets End with Good News

Bill Once theorists learned that $\exists$ an FPT algorithm using GMT they found practical FPT algorithms that do not use GMT.

Ryan's Friend How good?
Bill There is a simple $O(2^k n^2)$ algorithm. The best known is in $O(kn + 1.274^k)$. Both have very small constants in the O-of. The second one is useable and I could code it up.

Ryan's Friend You could code it up? In what computer language?

Bill The language HSS.

Ryan's Friend Oh. I don't know that one. Is that some fancy new language that David van Horn made up?

Bill HSS stands for High School Student.

Summary

Summary

From the GMT many problems that were not known to be in P, are now in P.

Summary

From the GMT many problems that were not known to be in P, are now in P.

Using GMT some problems that took $O(n^k)$ where k was some parameter of the problem, are now in $O(f(k)n^{O(1)})$ for some function f. These problems are called **Fixed Parameter Tractable (FPT)**

Summary

From the GMT many problems that were not known to be in P, are now in P.

Using GMT some problems that took $O(n^k)$ where k was some parameter of the problem, are now in $O(f(k)n^{O(1)})$ for some function f. These problems are called **Fixed Parameter Tractable (FPT)**

For many of these problems techniques that do not use GMT were developed.

Summary

From the GMT many problems that were not known to be in P, are now in P.

Using GMT some problems that took $O(n^k)$ where k was some parameter of the problem, are now in $O(f(k)n^{O(1)})$ for some function f. These problems are called **Fixed Parameter Tractable (FPT)**

For many of these problems techniques that do not use GMT were developed.

The GMT algorithms are still important in that once you know that a problem has a better-than-you-thought algorithm, you know that its plausible to look for a practical algorithm.

Summary

From the GMT many problems that were not known to be in P, are now in P.

Using GMT some problems that took $O(n^k)$ where k was some parameter of the problem, are now in $O(f(k)n^{O(1)})$ for some function f. These problems are called **Fixed Parameter Tractable (FPT)**

For many of these problems techniques that do not use GMT were developed.

The GMT algorithms are still important in that once you know that a problem has a better-than-you-thought algorithm, you know that its plausible to look for a practical algorithm.

There is a complexity theory to show problems are likely not FPT.