BILL AND NATHAN START RECORDING

HW07 Solutions

Hw 07, Problem 2a

$$\mathrm{CLIQ} = \{(G, k) : \ G \text{ has a clique of size } k \ \}.$$

$\omega(G)$ is the size of the largest clique in G.

Hw 07, Problem 2a

$$\mathrm{CLIQ} = \{(G, k) : \ G \text{ has a clique of size } k \ \}.$$

$\omega(G)$ is the size of the largest clique in G.

Show that if $\mathrm{CLIQ} \in \mathrm{P}$ then there is a poly time algorithm to compute $\omega(G)$. How many queries?

Hw 07, Problem 2a

$$\mathrm{CLIQ} = \{(G, k) : \ G \text{ has a clique of size } k \ \}.$$

$\omega(G)$ is the size of the largest clique in G.

Show that if $\mathrm{CLIQ} \in \mathrm{P}$ then there is a poly time algorithm to compute $\omega(G)$. How many queries?

Given G do the following.

(1) On Input G on n vertices, ask $\mathrm{CLIQ}\ (G, \frac{n}{2}) \in \mathrm{CLIQ}$.

Hw 07, Problem 2a

$$\mathrm{CLIQ} = \{(G, k) : \ G \text{ has a clique of size } k \ \}.$$

$\omega(G)$ is the size of the largest clique in G.

Show that if $\mathrm{CLIQ} \in \mathrm{P}$ then there is a poly time algorithm to compute $\omega(G)$. How many queries?

Given G do the following.

(1) On Input G on n vertices, ask $\mathrm{CLIQ}\ (G, \frac{n}{2}) \in \mathrm{CLIQ}$.

(2) Proceed by binary search to find k such that $(G, k) \notin \mathrm{CLIQ}$ and $(G, k+1) \in \mathrm{CLIQ}$.

Hw 07, Problem 2a

$$\mathrm{CLIQ} = \{(G, k) : \ G \text{ has a clique of size } k \ \}.$$

$\omega(G)$ is the size of the largest clique in G.

Show that if $\mathrm{CLIQ} \in \mathrm{P}$ then there is a poly time algorithm to compute $\omega(G)$. How many queries?

Given G do the following.

(1) On Input G on n vertices, ask $\mathrm{CLIQ}\ (G, \frac{n}{2}) \in \mathrm{CLIQ}$.

(2) Proceed by binary search to find k such that $(G, k) \notin \mathrm{CLIQ}$ and $(G, k+1) \in \mathrm{CLIQ}$.

(3) Output k.

Hw 07, Problem 2a

$$\mathrm{CLIQ} = \{(G, k) : \ G \text{ has a clique of size } k \ \}.$$

$\omega(G)$ is the size of the largest clique in G.

Show that if $\mathrm{CLIQ} \in \mathrm{P}$ then there is a poly time algorithm to compute $\omega(G)$. How many queries?

Given G do the following.

(1) On Input G on n vertices, ask $\mathrm{CLIQ}\ (G, \frac{n}{2}) \in \mathrm{CLIQ}$.

(2) Proceed by binary search to find k such that $(G, k) \notin \mathrm{CLIQ}$ and $(G, k+1) \in \mathrm{CLIQ}$.

(3) Output k.

The algorithm takes $\lg n + O(1)$ calls to CLIQ.

Hw 07, Problem 2b

$\mathrm{FCLIQ}(G)$ returns a clique of size $\omega(G)$.

Show that if $\mathrm{CLIQ} \in \mathrm{P}$ then there is a poly time algorithm to compute $\mathrm{FCLIQ}(G)$. How many queries?

Hw 07, Problem 2b

$\mathrm{FCLIQ}(G)$ returns a clique of size $\omega(G)$.

Show that if $\mathbf{CLIQ} \in \mathbf{P}$ then there is a poly time algorithm to compute $\mathbf{FCLIQ}(G)$. How many queries?

Given G do the following:

(1) Use the alg above to find $k = \omega(G)$. ($\lg(n) + O(1)$.)

Hw 07, Problem 2b

$\mathrm{FCLIQ}(G)$ returns a clique of size $\omega(G)$.

Show that if $\mathbf{CLIQ} \in \mathbf{P}$ then there is a poly time algorithm to compute $\mathbf{FCLIQ}(G)$. How many queries?

Given G do the following:

(1) Use the alg above to find $k = \omega(G)$. ($\lg(n) + O(1)$.)

(2) Next goto next page.

Hw 07, Problem 2b (cont)

(1) We have G and know $k = \omega(G)$. Want to find cliq of size k.

Hw 07, Problem 2b (cont)

(1) We have G and know $k = \omega(G)$. Want to find cliq of size k.

(2) For each node v ask the question $\mathrm{CLIQ}(G - \{v\}, k)$.

Hw 07, Problem 2b (cont)

(1) We have G and know $k = \omega(G)$. Want to find cliq of size k.

(2) For each node v ask the question $\mathrm{CLIQ}(G - \{v\}, k)$.

(3a) If YES then we know that v IS NOT NEEDED for the max clique, so remove v and proceed to the next vertex.

Hw 07, Problem 2b (cont)

(1) We have G and know $k = \omega(G)$. Want to find cliq of size k.

(2) For each node v ask the question $\mathrm{CLIQ}(G - \{v\}, k)$.

(3a) If YES then we know that v IS NOT NEEDED for the max clique, so remove v and proceed to the next vertex.

(3b) If NO then we know v IS NEEDED so keep it.

Hw 07, Problem 2b (cont)

(1) We have G and know $k = \omega(G)$. Want to find cliq of size k.

(2) For each node v ask the question $\mathrm{CLIQ}(G - \{v\}, k)$.

(3a) If YES then we know that v IS NOT NEEDED for the max clique, so remove v and proceed to the next vertex.

(3b) If NO then we know v IS NEEDED so keep it.

The graph you are left with is a clique of size k.

Hw 07, Problem 2b (cont)

(1) We have G and know $k = \omega(G)$. Want to find cliq of size k.

(2) For each node v ask the question $\mathrm{CLIQ}(G - \{v\}, k)$.

(3a) If YES then we know that v IS NOT NEEDED for the max clique, so remove v and proceed to the next vertex.

(3b) If NO then we know v IS NEEDED so keep it.

The graph you are left with is a clique of size k.

Queries $\lg n + O(1)$ to find $\omega(G)$. n to the find cliq.

$$n + \lg(n) + O(1) \text{ queries}$$

Hw 07, Problem 2c

Find a function f such that the following holds:
If you could compute $\mathrm{FCLIQ}(G)$ with $f(n)$ queries to CLIQ then P=NP.

Hw 07, Problem 2c

Find a function f such that the following holds:
If you could compute $\mathrm{FCLIQ}(G)$ with $f(n)$ queries to CLIQ then P=NP.
$f(n) = \lg n$.
Assume that there is a poly time algorithm for $\mathrm{FCLIQ}(G)$ that makes $\lg n$ queries to CLIQ.

Hw 07, Problem 2c

Find a function f such that the following holds:
If you could compute $\mathrm{FCLIQ}(G)$ with $f(n)$ queries to CLIQ then P=NP.
$f(n) = \lg n$.
Assume that there is a poly time algorithm for $\mathrm{FCLIQ}(G)$ that makes $\lg n$ queries to CLIQ.
(1) On input G run the algorithm, simulate both the YES and NO answer for all queries.

Hw 07, Problem 2c

Find a function f such that the following holds:
If you could compute $\mathrm{FCLIQ}(G)$ with $f(n)$ queries to CLIQ then P=NP.
$f(n) = \lg n$.
Assume that there is a poly time algorithm for $\mathrm{FCLIQ}(G)$ that makes $\lg n$ queries to CLIQ.
(1) On input G run the algorithm, simulate both the YES and NO answer for all queries.
(2) Since there are $\lg n$ queries, there are $\leq 2^{\lg n} = n$ diff outputs.

Hw 07, Problem 2c

Find a function f such that the following holds:
If you could compute $\mathrm{FCLIQ}(G)$ with $f(n)$ queries to CLIQ then P=NP.
$f(n) = \lg n$.
Assume that there is a poly time algorithm for $\mathrm{FCLIQ}(G)$ that makes $\lg n$ queries to CLIQ.
(1) On input G run the algorithm, simulate both the YES and NO answer for all queries.
(2) Since there are $\lg n$ queries, there are $\leq 2^{\lg n} = n$ diff outputs.
(3) Hence there will be n **candidates** for the largest clique. (Some may be garbage- not even cliques!)

Hw 07, Problem 2c

Find a function f such that the following holds:
If you could compute $\mathrm{FCLIQ}(G)$ with $f(n)$ queries to CLIQ then P=NP.
$f(n) = \lg n$.
Assume that there is a poly time algorithm for $\mathrm{FCLIQ}(G)$ that makes $\lg n$ queries to CLIQ.
(1) On input G run the algorithm, simulate both the YES and NO answer for all queries.
(2) Since there are $\lg n$ queries, there are $\leq 2^{\lg n} = n$ diff outputs.
(3) Hence there will be n **candidates** for the largest clique. (Some may be garbage- not even cliques!)
(4) Output the largest candidate that is a clique.

Hw 07, Problem 3a

Let A, B, C be sets of strings.

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \leq B$ and $B \leq C$ then $A \leq C$.

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \leq B$ and $B \leq C$ then $A \leq C$.
$A \leq B$ by a poly red via program M_1; time p_1, a poly.

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \leq B$ and $B \leq C$ then $A \leq C$.
$A \leq B$ by a poly red via program M_1; time p_1, a poly.
$B \leq C$ by a poly red via program M_2; time p_2, a poly.

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \leq B$ and $B \leq C$ then $A \leq C$.
$A \leq B$ by a poly red via program M_1; time p_1, a poly.
$B \leq C$ by a poly red via program M_2; time p_2, a poly.
Here is a reduction $A \leq C$:

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \le B$ and $B \le C$ then $A \le C$.
$A \le B$ by a poly red via program M_1; time p_1, a poly.
$B \le C$ by a poly red via program M_2; time p_2, a poly.
Here is a reduction $A \le C$:

(1) Given x, compute $y = M_1(x)$. Time $p_1(|x|)$. Note $|y| \le p_1(|x|)$.

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \leq B$ and $B \leq C$ then $A \leq C$.
$A \leq B$ by a poly red via program M_1; time p_1, a poly.
$B \leq C$ by a poly red via program M_2; time p_2, a poly.
Here is a reduction $A \leq C$:

(1) Given x, compute $y = M_1(x)$. Time $p_1(|x|)$. Note $|y| \leq p_1(|x|)$.
(2) Then compute $z = M_2(y)$. Time $\leq p_2(|y|) \leq p_2(p_1(|x|))$.

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \leq B$ and $B \leq C$ then $A \leq C$.
$A \leq B$ by a poly red via program M_1; time p_1, a poly.
$B \leq C$ by a poly red via program M_2; time p_2, a poly.
Here is a reduction $A \leq C$:

(1) Given x, compute $y = M_1(x)$. Time $p_1(|x|)$. Note $|y| \leq p_1(|x|)$.
(2) Then compute $z = M_2(y)$. Time $\leq p_2(|y|) \leq p_2(p_1(|x|))$.
Poly closed under composition, so reduction is poly time.

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \le B$ and $B \le C$ then $A \le C$.
$A \le B$ by a poly red via program M_1; time p_1, a poly.
$B \le C$ by a poly red via program M_2; time p_2, a poly.
Here is a reduction $A \le C$:

(1) Given x, compute $y = M_1(x)$. Time $p_1(|x|)$. Note $|y| \le p_1(|x|)$.
(2) Then compute $z = M_2(y)$. Time $\le p_2(|y|) \le p_2(p_1(|x|))$.
Poly closed under composition, so reduction is poly time.
Note that
$x \in A$ iff $y \in B$ iff $z \in C$

Hw 07, Problem 3a

Let A, B, C be sets of strings.
Show that if $A \leq B$ and $B \leq C$ then $A \leq C$.
$A \leq B$ by a poly red via program M_1; time p_1, a poly.
$B \leq C$ by a poly red via program M_2; time p_2, a poly.
Here is a reduction $A \leq C$:

(1) Given x, compute $y = M_1(x)$. Time $p_1(|x|)$. Note $|y| \leq p_1(|x|)$.
(2) Then compute $z = M_2(y)$. Time $\leq p_2(|y|) \leq p_2(p_1(|x|))$.
Poly closed under composition, so reduction is poly time.
Note that
$x \in A$ iff $y \in B$ iff $z \in C$
So we have a reduction $A \leq C$.

HW 07, Problem 3b

Show that if $A \leq B$ and $B \in \mathrm{P}$ then $A \in \mathrm{P}$

HW 07, Problem 3b

Show that if $A \leq B$ and $B \in \mathrm{P}$ then $A \in \mathrm{P}$

Omitted, but similar to the the prior problem.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

Is $\mathrm{COL}_4 \leq \mathrm{COL}_3$? Vote.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

Is $\mathrm{COL}_4 \leq \mathrm{COL}_3$? Vote.

YES.
$\mathrm{COL}_4 \leq \mathrm{SAT}$ by Cook-Levin.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

Is $\mathrm{COL}_4 \leq \mathrm{COL}_3$? Vote.

YES.
$\mathrm{COL}_4 \leq \mathrm{SAT}$ by Cook-Levin. $\mathrm{SAT} \leq \mathrm{COL}_3$ we showed in class.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

Is $\mathrm{COL}_4 \leq \mathrm{COL}_3$? Vote.

YES.
$\mathrm{COL}_4 \leq \mathrm{SAT}$ by Cook-Levin. $\mathrm{SAT} \leq \mathrm{COL}_3$ we showed in class.

SO by Prior Problem, $\mathrm{COL}_4 \leq \mathrm{COL}_3$.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \leq \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

Is $\mathrm{COL}_4 \leq \mathrm{COL}_3$? Vote.

YES.
$\mathrm{COL}_4 \leq \mathrm{SAT}$ by Cook-Levin. $\mathrm{SAT} \leq \mathrm{COL}_3$ we showed in class.

SO by Prior Problem, $\mathrm{COL}_4 \leq \mathrm{COL}_3$.

This reduction is INSANE.

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \le \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

Is $\mathrm{COL}_4 \le \mathrm{COL}_3$? Vote.

YES.
$\mathrm{COL}_4 \le \mathrm{SAT}$ by Cook-Levin. $\mathrm{SAT} \le \mathrm{COL}_3$ we showed in class.

SO by Prior Problem, $\mathrm{COL}_4 \le \mathrm{COL}_3$.

This reduction is INSANE.
Is there a sane reduction? YES!

HW 07 Problem 4a

$$\mathrm{COL}_k = \{G : G \text{ is } k\text{-colorable}\}.$$

Show that $\mathrm{COL}_3 \le \mathrm{COL}_4$.

Given G, form G' by adding a vertex and connecting it to all of the original vertices.

G is 3-col iff G' is 4-col.

Is $\mathrm{COL}_4 \le \mathrm{COL}_3$? Vote.

YES.
$\mathrm{COL}_4 \le \mathrm{SAT}$ by Cook-Levin. $\mathrm{SAT} \le \mathrm{COL}_3$ we showed in class.

SO by Prior Problem, $\mathrm{COL}_4 \le \mathrm{COL}_3$.

This reduction is INSANE.
Is there a sane reduction? YES!
`https://arxiv.org/abs/1407.5128`

HW07, Problem 5a

$$\mathrm{VC}_{17} = \{G : \text{There is a vertex cover of size} \leq 17\}.$$

HW07, Problem 5a

$$\mathrm{VC}_{17} = \{G : \text{There is a vertex cover of size} \leq 17\}.$$

Show that $\mathrm{VC}_{17} \in \mathrm{P}$ and give the run time.

HW07, Problem 5a

$$\mathrm{VC}_{17} = \{G : \text{There is a vertex cover of size} \leq 17\}.$$

Show that $\mathrm{VC}_{17} \in \mathrm{P}$ and give the run time.
Given G:

HW07, Problem 5a

$$\mathrm{VC}_{17} = \{G : \text{There is a vertex cover of size} \leq 17\}.$$

Show that $\mathrm{VC}_{17} \in \mathrm{P}$ and give the run time.
Given G:

(1) check every combo of 17 vertices to see if it is a vertex cover.

HW07, Problem 5a

$$\mathrm{VC}_{17} = \{G : \text{There is a vertex cover of size} \leq 17\}.$$

Show that $\mathrm{VC}_{17} \in \mathrm{P}$ and give the run time.
Given G:

(1) check every combo of 17 vertices to see if it is a vertex cover.

(2) There are $\binom{n}{17} \leq n^{17}$ such combo, and for each combo, there are $\leq n^2$ edges to check, so our alg runs in n^{19} time.

HW07, Problem 5b

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

Find a function f such that the following is true.
VC_k *can be solved in time* $f(n, k)$.

HW07, Problem 5b

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

Find a function f such that the following is true.
VC_k *can be solved in time* $f(n, k)$.
Given G
(1) check every combo of k vertices to see if it is a vertex cover.

HW07, Problem 5b

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

Find a function f such that the following is true.
VC_k *can be solved in time* $f(n, k)$.
Given G
(1) check every combo of k vertices to see if it is a vertex cover.
(2) There are $\binom{n}{k} \leq n^k$ such combo, and for each combo, there are $\leq n^k$ edges to check, so our alg runs in n^{k+2} time.

HW07, Problem 5b

HW07, Problem 5c

HW07, Problem 5c

Vote for one of the following.

HW07, Problem 5c

Vote for one of the following.

1. There is an algorithm for VC_k where the deg doesn't dep on k.

HW07, Problem 5c

Vote for one of the following.

1. There is an algorithm for VC_k where the deg doesn't dep on k.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.

HW07, Problem 5c

Vote for one of the following.

1. There is an algorithm for VC_k where the deg doesn't dep on k.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.
3. Neither (a) or (b) is known (Guido will determine which one this summer.)

HW07, Problem 5c

Vote for one of the following.

1. There is an algorithm for VC_k where the deg doesn't dep on k.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.
3. Neither (a) or (b) is known (Guido will determine which one this summer.)

Answer is (1).

HW07, Problem 5c

Vote for one of the following.

1. There is an algorithm for VC_k where the deg doesn't dep on k.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.
3. Neither (a) or (b) is known (Guido will determine which one this summer.)

Answer is (1).

See next slide.

$\mathbf{VC}_k$

Given (G, k) form the following tree.
(1) Root has G and an edge (u, v).
Key either u or v has to be in the VC.

$\mathbf{VC}_k$

Given (G, k) form the following tree.
(1) Root has G and an edge (u, v).
Key either u or v has to be in the VC.

(2) Left child is $G - \{u\}$, Right child is $G - \{v\}$.
For both pick an edge and do Left-Right as in step (1).

$\mathbf{VC}_k$

Given (G, k) form the following tree.
(1) Root has G and an edge (u, v).
Key either u or v has to be in the VC.

(2) Left child is $G - \{u\}$, Right child is $G - \{v\}$.
For both pick an edge and do Left-Right as in step (1).

(3) Grow the tree to k levels.

$\mathbf{VC}_k$

Given (G, k) form the following tree.
(1) Root has G and an edge (u, v).
Key either u or v has to be in the VC.

(2) Left child is $G - \{u\}$, Right child is $G - \{v\}$.
For both pick an edge and do Left-Right as in step (1).

(3) Grow the tree to k levels.

(4) If any of tree-nodes is the empty graph then have a VC of size k.
If none do then there is no VC of size k.

$\mathbf{VC}_k$

Given (G, k) form the following tree.
(1) Root has G and an edge (u, v).
Key either u or v has to be in the VC.

(2) Left child is $G - \{u\}$, Right child is $G - \{v\}$.
For both pick an edge and do Left-Right as in step (1).

(3) Grow the tree to k levels.

(4) If any of tree-nodes is the empty graph then have a VC of size k.
If none do then there is no VC of size k.

$\leq 2^k$ tree-nodes. Each one takes $O(n)$ to process. Time $O(2^k n)$.

$\mathbf{VC}_k$

Given (G, k) form the following tree.
(1) Root has G and an edge (u, v).
Key either u or v has to be in the VC.

(2) Left child is $G - \{u\}$, Right child is $G - \{v\}$.
For both pick an edge and do Left-Right as in step (1).

(3) Grow the tree to k levels.

(4) If any of tree-nodes is the empty graph then have a VC of size k.
If none do then there is no VC of size k.

$\leq 2^k$ tree-nodes. Each one takes $O(n)$ to process. Time $O(2^k n)$.

Can we do even better? Next Slide.

Better?

We have run time $O(2^k n)$. We think of $k \ll n$.

Better?

We have run time $O(2^k n)$. We think of $k \ll n$.

Vote for one of the following.

Better?

We have run time $O(2^k n)$. We think of $k \ll n$.

Vote for one of the following.

1. ($\exists$) alg for VC_k and, time $n^{O(1)} + f(k)$ for some f.

Better?

We have run time $O(2^k n)$. We think of $k \ll n$.

Vote for one of the following.

1. ($\exists$) alg for VC_k and, time $n^{O(1)} + f(k)$ for some f.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.

Better?

We have run time $O(2^k n)$. We think of $k \ll n$.

Vote for one of the following.

1. ($\exists$) alg for VC_k and, time $n^{O(1)} + f(k)$ for some f.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.
3. Neither (a) or (b) is known (Guido will determine which one this summer.)

Better?

We have run time $O(2^k n)$. We think of $k \ll n$.

Vote for one of the following.

1. ($\exists$) alg for VC_k and, time $n^{O(1)} + f(k)$ for some f.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.
3. Neither (a) or (b) is known (Guido will determine which one this summer.)

Answer is (1).

Better?

We have run time $O(2^k n)$. We think of $k \ll n$.

Vote for one of the following.

1. ($\exists$) alg for VC_k and, time $n^{O(1)} + f(k)$ for some f.
2. If $\mathrm{P} \neq \mathrm{NP}$ then there is no such algorithm.
3. Neither (a) or (b) is known (Guido will determine which one this summer.)

Answer is (1).

See next slide.

$\mathbf{VC}_k$

Given G notice that

if vertex v has degree $\geq k+1$ then it HAS to be in the VC

1. Input $G = (V, E)$. Let $A = \emptyset$. Let $t = k$.
2. For $v \in V$
   2.1 If v has degree $\geq t+1$ then
   $A := A \cup \{v\}$,
   $t := t - 1$,
   $G := G - \{v\}$.
3. If the graph is empty, you are done, and A is your VC.
4. If the graph is not empty then you have a graph with every vertex degree $\leq k$. Leave the rest to you.

Run time is

$$n + \alpha^k$$

More refined approaches have gotten $\alpha = 1.34$.