

# CSC363 Tutorial 11

almost done!

Paul “sushi\_enjoyer” Zhang

University of Chungi

March 31, 2021



# Learning objectives this tutorial

By the end of this tutorial, you should...

Have one more NP-complete problem added to your “NP-complete problems” toolkit.

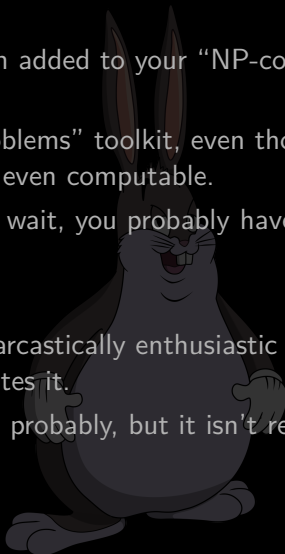
Add a problem to your “NP-hard problems” toolkit, even though it’s kinda useless since the problem isn’t even computable.

Probably work on assignment 5? Oh wait, you probably have other courses with higher priority... :(

Be scared for the final exam! D:

Feel like sushi\_enjoyer is just being sarcastically enthusiastic about CSC363 material when he himself hates it.

Big Chungus certified readings: chapter 8 probably, but it isn’t really necessary tbh.



do you like proving NP-completeness? D:

well too bad! you'll have to do it for the upcoming problem set.



# Set cover

We will now describe the **set cover problem** and prove it is NP-complete, because why not.

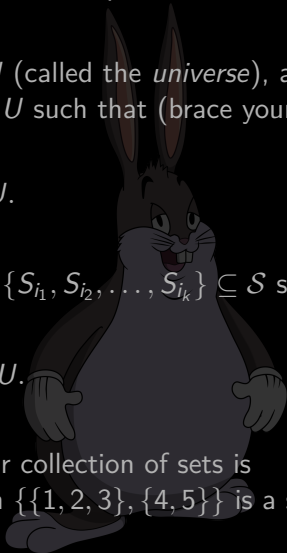
Suppose we are given a set of elements  $U$  (called the *universe*), and a collection  $\mathcal{S} = \{S_1, \dots, S_n\}$  of subsets of  $U$  such that (brace yourself,  $\cup$ )

$$\bigcup_{i=1}^n S_i = U.$$

A **set cover** of  $U$  is a subcollection  $\mathcal{S}' = \{S_{i_1}, S_{i_2}, \dots, S_{i_k}\} \subseteq \mathcal{S}$  such that

$$\bigcup_{m=1}^k S_{i_m} = U.$$

For example, if  $U = \{1, 2, 3, 4, 5\}$ , and our collection of sets is  $\mathcal{S} = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$ , then  $\{\{1, 2, 3\}, \{4, 5\}\}$  is a set cover.



# Set cover

**Task:** Let  $U = \{\text{🍣}, \text{🍷}, \text{🥕}, \text{🥔}, \text{🥕}, \text{🍷}\}$ , and

$$\mathcal{S} = \{\{\text{🍣}, \text{🥕}\}, \{\text{🍷}, \text{🥕}, \text{🍷}\}, \{\text{🍷}, \text{🥕}, \text{🥕}\}, \{\text{🥔}, \text{🍷}\}, \{\text{🥔}\}, \{\text{🍣}, \text{🥕}, \text{🥔}\}\}.$$

Find the smallest set cover of  $U$ .

**Answer:** The smallest set cover is  $\{\{\text{🍷}, \text{🥕}, \text{🍷}\}, \{\text{🍣}, \text{🥕}, \text{🥔}\}\}$ , which is of size 2.



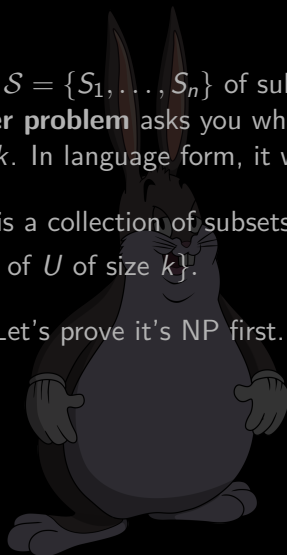
# Set cover is NP-complete!

Now given a universal set  $U$ , a collection  $\mathcal{S} = \{S_1, \dots, S_n\}$  of subsets of  $U$ , and a natural number  $k$ , the **set cover problem** asks you whether it is possible to find a set cover for  $U$  of size  $k$ . In language form, it would be

Set-Cover =  $\{(U, \mathcal{S}, k) : U \text{ is a set, } \mathcal{S} \text{ is a collection of subsets of } U,$   
and there is a set cover of  $U$  of size  $k\}$ .

Turns out this problem is NP-complete! Let's prove it's NP first.

**Task:** Prove that Set-Cover is NP.



# Set cover is NP-complete!

**Task:** Prove that Set-Cover is NP.

**Answer:** We can build a poly-time verifier  $V$  that checks whether a given subcollection  $\mathcal{S}'$  of  $\mathcal{S}$  is a set cover for  $U$ .

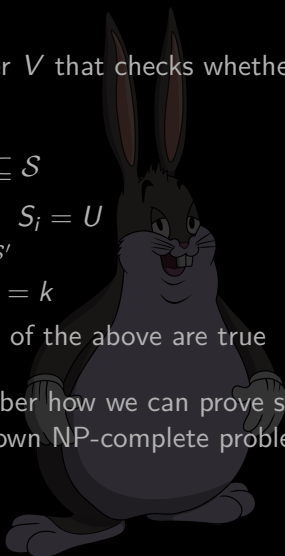
$V(U, \mathcal{S}, k, \mathcal{S}')$  : Check if  $\mathcal{S}' \subseteq \mathcal{S}$

Check if  $\bigcup_{S_i \in \mathcal{S}'} S_i = U$

Check if  $|\mathcal{S}'| = k$

Accept iff all of the above are true

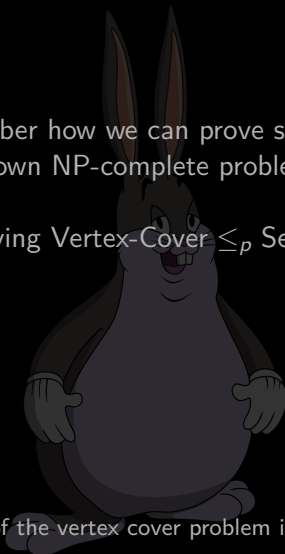
Now we prove it is NP-complete. Remember how we can prove something is NP-complete by showing that some known NP-complete problem reduces to it?



# Set cover is NP-complete!

Now we prove it is NP-complete. Remember how we can prove something is NP-complete by showing that some known NP-complete problem reduces to it?

**Task:** Show that  $\text{Set-Cover} \in \text{NP}$  by proving  $\text{Vertex-Cover} \leq_p \text{Set-Cover}$ .<sup>1</sup>



---

<sup>1</sup>Recall: this involves converting an instance of the vertex cover problem into an instance of set cover problem in poly-time.



# Set cover is NP-complete!

**Answer:** Suppose we are given an instance  $(G, k)$  of the vertex cover problem. We may transform it into a set cover problem  $(U_G, \mathcal{S}_G, k_G)$  with the property that

$$(G, k) \in \text{Vertex-Cover} \Leftrightarrow (U_G, \mathcal{S}_G, k_G) \in \text{Set-Cover}.$$

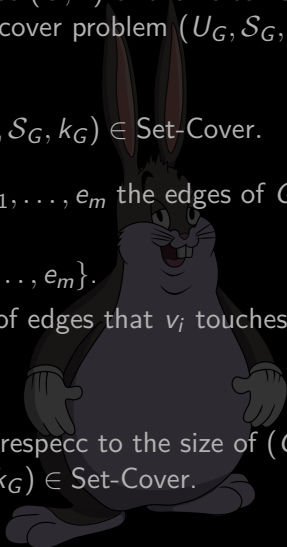
Let  $v_1, \dots, v_n$  be the vertices of  $G$ , and  $e_1, \dots, e_m$  the edges of  $G$ . Define  $U_G, \mathcal{S}_G, k_G$  as follows:

$U_G$  will consist of all the edges  $\{e_1, \dots, e_m\}$ .

For each vertex  $v_i$ , let  $S_i$  be the set of edges that  $v_i$  touches. Then let  $\mathcal{S}_G = \{S_1, \dots, S_n\}$ .

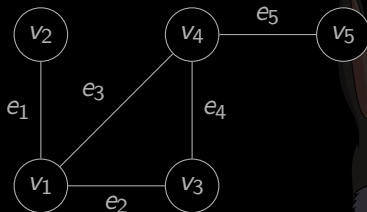
$k_G = k$ .

This transformation takes poly-time with respect to the size of  $(G, k)$ . We claim  $(G, k) \in \text{Vertex-Cover} \Leftrightarrow (U_G, \mathcal{S}_G, k_G) \in \text{Set-Cover}$ .



# Set cover is NP-complete!

We'll "prove"  $(G, k) \in \text{Vertex-Cover} \Leftrightarrow (U_G, \mathcal{S}_G, k_G) \in \text{Set-Cover}$  via example.<sup>2</sup> Suppose  $k = 2$  and  $G$  is the following graph:



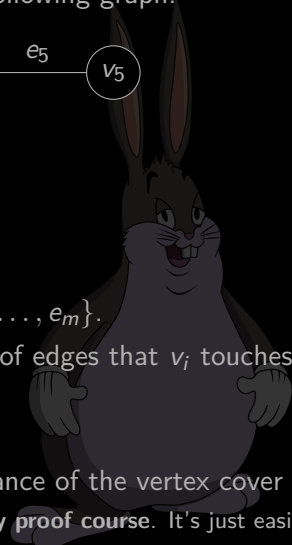
$U_G$  will consist of all the edges  $\{e_1, \dots, e_m\}$ .

For each vertex  $v_i$ , let  $S_i$  be the set of edges that  $v_i$  touches. Then let  $\mathcal{S}_G = \{S_1, \dots, S_n\}$ .

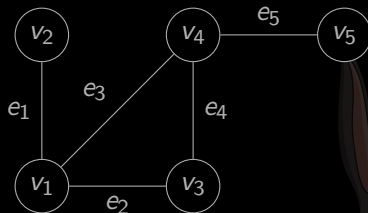
$k_G = k$ .

**Task:** Find  $U_G$ ,  $\mathcal{S}_G$ , and  $k_G$  for this instance of the vertex cover problem.

<sup>2</sup>Please, please, please, **do not do this in any proof course**. It's just easier for illustrate with an example.



# Set cover is NP-complete!



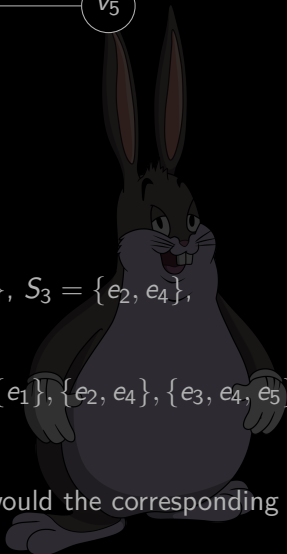
$$U_G = \{e_1, e_2, e_3, e_4, e_5\}.$$

We have  $S_1 = \{e_1, e_2, e_3\}$ ,  $S_2 = \{e_1\}$ ,  $S_3 = \{e_2, e_4\}$ ,  $S_4 = \{e_3, e_4, e_5\}$ ,  $S_5 = \{e_5\}$ . So

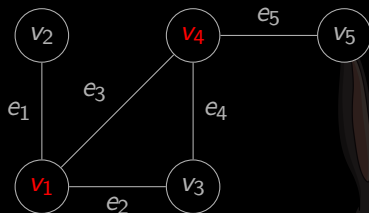
$$\mathcal{S}_G = \{S_1, \dots, S_5\} = \{\{e_1, e_2, e_3\}, \{e_1\}, \{e_2, e_4\}, \{e_3, e_4, e_5\}, \{e_5\}\}.$$

$$k_G = 2 \text{ since } k = 2.$$

**Task:** Find a vertex cover for  $G$ . What would the corresponding set cover be?



# Set cover is NP-complete!



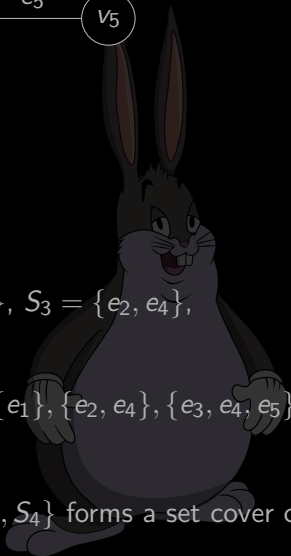
$$U_G = \{e_1, e_2, e_3, e_4, e_5\}.$$

We have  $S_1 = \{e_1, e_2, e_3\}$ ,  $S_2 = \{e_1\}$ ,  $S_3 = \{e_2, e_4\}$ ,  
 $S_4 = \{e_3, e_4, e_5\}$ ,  $S_5 = \{e_5\}$ . So

$$\mathcal{S}_G = \{S_1, \dots, S_5\} = \{\{e_1, e_2, e_3\}, \{e_1\}, \{e_2, e_4\}, \{e_3, e_4, e_5\}, \{e_5\}\}.$$

$$k_G = 2 \text{ since } k = 2.$$

$v_1, v_4$  form a vertex cover of  $G$ .  $\mathcal{S}' = \{S_1, S_4\}$  forms a set cover of  $U$ .



# Break time!

No sushi juice this time. But you get to ask me one question, about pretty much anything (as long as it's appropriate i guess lol).



# no more brake time with uwu

Alright so we now have one more problem that we know is NP-complete.  
I'm so excited! Anyone? ;-;

Let's add an NP-hard problem to the back of our memory! This one is  
actually a bit tricky to prove though...

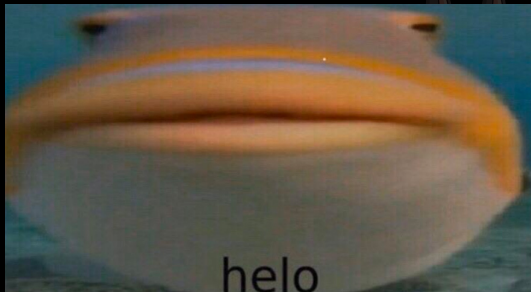
**Task:** What does HP stand for?



# no more brake time with uwu

**Task:** What does HP stand for?

**Answer:** Helo Phish.



$HP = \{(M, w) : M \text{ is a Turing machine that halts on input } w\}.$

We will prove HP is NP-Hard by showing  $3SAT \leq_p HP$ .<sup>3</sup>

---

<sup>3</sup>In fact, any computable language  $A$  satisfies  $A \leq_p HP$ ! You can just adapt the proof I'm about to show.

# HP is NP-complete

$HP = \{(M, w) : M \text{ is a Turing machine that halts on input } w\}$ .

We will construct the following reduction of 3SAT to HP. Suppose  $\varphi$  is a given instance of 3SAT. Construct the following Turing machine  $M$ :

$M(\varphi)$  : Check whether  $\varphi \in 3SAT$  via brute force.

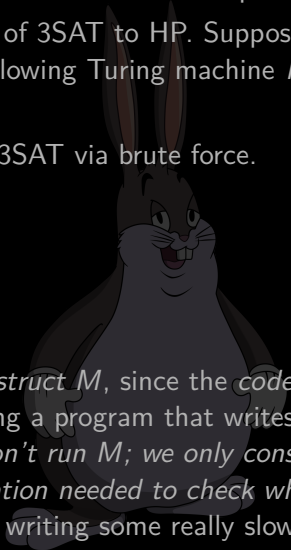
If  $\varphi \in 3SAT$ :

Accept

Else:

Loop

Notice that *it takes constant time to construct  $M$* , since the code of  $M$  doesn't depend on  $\varphi$  at all. It's like writing a program that writes a fixed Python script into a text file. Also, *we don't run  $M$ ; we only construct it, and bypass the exponential time computation needed to check whether  $\varphi \in 3SAT$  via brute force.* Again, it's like writing some really slow code to a text file but not running it.





# HP is NP-complete

$HP = \{(M, w) : M \text{ is a Turing machine that halts on input } w\}$ .

We will construct the following reduction of 3SAT to HP. Suppose  $\varphi$  is a given instance of 3SAT. Construct the following Turing machine  $M$ :

$M(\varphi)$  : Check whether  $\varphi \in 3SAT$  via brute force.

If  $\varphi \in 3SAT$ :

Accept

Else:

Loop

**Task:** Show  $\varphi \in 3SAT \Leftrightarrow (M, \varphi) \in HP$ , where  $M$  is as above. Then convince yourself that we can replace 3SAT with any computable language, and the same proof would work.



buy

helo\_fish.jpg is sad to see you go ;-;  
only one more week left! D:

