Foundations of Computing

Lecture 23

Arkady Yerukhimovich

April 16, 2024

Final Exam

Final exam will be on Tuesday, May 7, 10:20-12:20.

Outline

Lecture 22 Review

- More $\mathcal{NP}$-complete problems
  - SAT
  - 3SAT
  - CLIQUE
  - VERTEX-COVER

Outline

3-Coloring

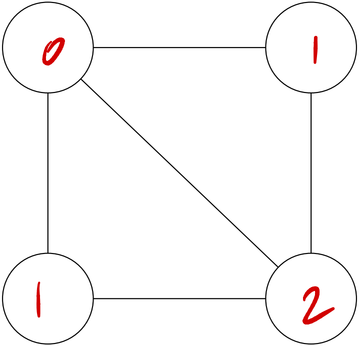
Definition

An undirected graph G is 3-colorable, if can assign colors $\{0, 1, 2\}$ to all nodes, such that no edges have the same color on both ends.

3-Coloring

Definition

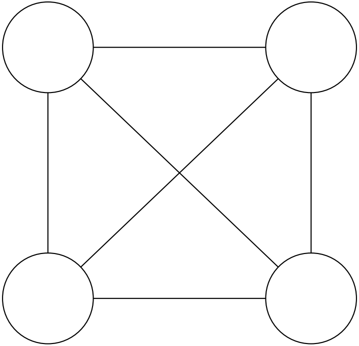
An undirected graph G is 3-colorable, if can assign colors $\{0, 1, 2\}$ to all nodes, such that no edges have the same color on both ends.



3-Coloring

Definition

An undirected graph G is 3-colorable, if can assign colors $\{0, 1, 2\}$ to all nodes, such that no edges have the same color on both ends.



Goal: Prove than 3-Coloring is $\mathcal{NP}$-Complete

3-Coloring is $\mathcal{NP}$-Complete

1. 3-Coloring $\in \mathcal{NP}$

3-Coloring is $\mathcal{NP}$-Complete

1. 3-Coloring $\in \mathcal{NP}$
2. 3-SAT $\leq_p$ 3-Coloring:

3-Coloring is $\mathcal{NP}$-Complete

1. 3-Coloring $\in \mathcal{NP}$
2. 3-SAT $\leq_p$ 3-Coloring:

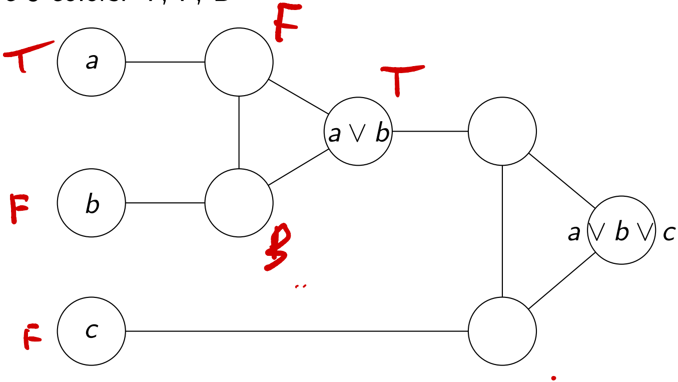
Main Tool

We need gadgets

$$\phi = \overbrace{(a \vee b \vee c)} \wedge (d \vee e \vee f)$$

Clause Gadget

We have 3 colors: T, F, B



Claim

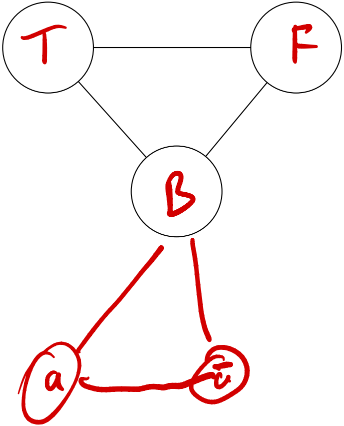
- If a, b, c are all colored F, then $a \vee b \vee c$ is colored F
- If at least one of a, b, c is colored T, then there is a coloring s.t. $a \vee b \vee c$ is colored T

Variable Gadget

Goal: Need to color variables T or F

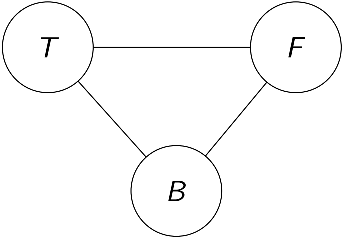
Variable Gadget

Goal: Need to color variables T or F



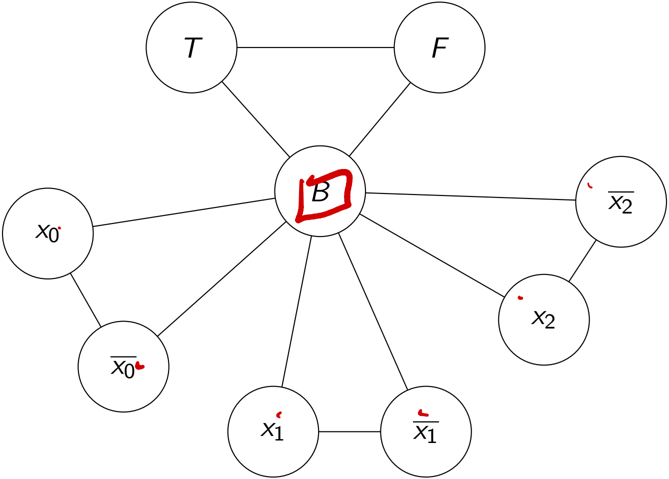
Variable Gadget

Goal: Need to color variables T or F

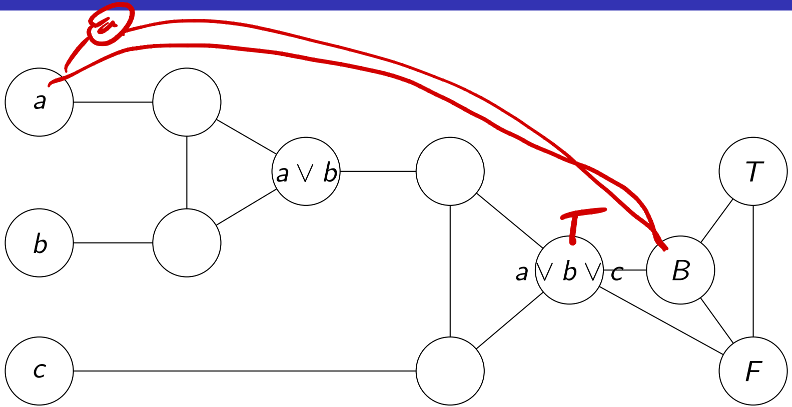


Variable Gadget

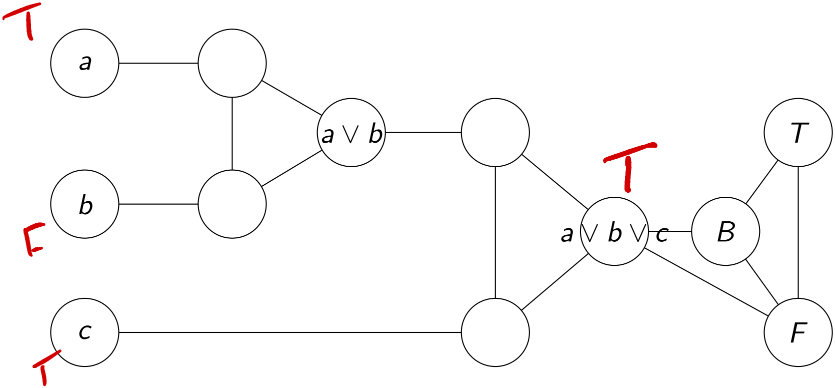
Goal: Need to color variables T or F



Putting it All Together



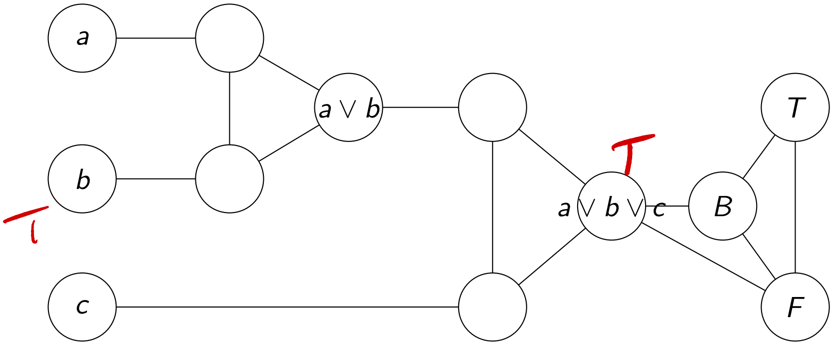
Putting it All Together



Claim

1. If ϕ is satisfiable, G is 3-colorable

Putting it All Together



Claim

1. If ϕ is satisfiable, G is 3-colorable
2. If G is 3-colorable than ϕ is satisfiable

Outline

Ladner's Theorem

- Recall that we know that $\mathcal{P} \subseteq \mathcal{NP}$

Ladner's Theorem

- Recall that we know that $\mathcal{P} \subseteq \mathcal{NP}$
- Suppose that $\mathcal{P} \neq \mathcal{NP}$:

Ladner's Theorem

- Recall that we know that $\mathcal{P} \subseteq \mathcal{NP}$
- Suppose that $\mathcal{P} \neq \mathcal{NP}$:

  Question: Are all languages either easy or very hard?

Is there an $L \in NP$ s.t.

$L \notin P$ and

L is not NP-complete

Ladner's Theorem

- Recall that we know that $\mathcal{P} \subseteq \mathcal{NP}$
- Suppose that $\mathcal{P} \neq \mathcal{NP}$:

  Question: Are all languages either easy or very hard?

  Math version: Is there an $L \in \mathcal{NP}$, s.t. $L \notin \mathcal{P}$ and L is not $\mathcal{NP}$-Complete?

Ladner's Theorem

If $\mathcal{P} \neq \mathcal{NP}$ then there exists an $L \in \mathcal{NP}$ s.t.

1. $L \notin \mathcal{P}$, and
2. L is not $\mathcal{NP}$-Complete

Ladner's Theorem

- Recall that we know that $\mathcal{P} \subseteq \mathcal{NP}$
- Suppose that $\mathcal{P} \neq \mathcal{NP}$:

  Question: Are all languages either easy or very hard?

  Math version: Is there an $L \in \mathcal{NP}$, s.t. $L \notin \mathcal{P}$ and L is not $\mathcal{NP}$-Complete?

Ladner's Theorem

If $\mathcal{P} \neq \mathcal{NP}$ then there exists an $L \in \mathcal{NP}$ s.t.

1. $L \notin \mathcal{P}$, and
2. L is not $\mathcal{NP}$-Complete

Comment: All languages useful for crypto are such $\mathcal{NP}$-intermediate languages

Proof

A Useful Language

$$SAT_H = \{\phi 01^{n^{\boxed{H(n)}}} \mid \phi \in SAT, n = |\phi|\}$$

Proof

$$SAT_H := \{ \phi 0 1^{n^c} \mid \phi \in SAT, |\phi| = n \}$$

$$n + 1 + n^c$$

A Useful Language

$$SAT_H = \{\phi 0 1^{n^{H(n)}} \mid \phi \in SAT, n = |\phi|\}$$

1. If $H(n) = n$, then $SAT_H \in \mathcal{P}$
2. If $H(n) \leq c$, then SAT_H is $\mathcal{NP}$-Complete
3. We will define H to be in between these two cases

$SAT_H \in P \Rightarrow \exists$ poly f. s.t. $M(x)$ run in time $\leq f(|x|)$

$$|x| = n^n + n + 1$$

Defining $H(n)$

Let $M_1, M_2, \ldots$ be an enumeration of all TM's (can do this since TM's are countable)

$H(n)$

- Smallest $i \leq \log\log n$ s.t. for all x, $|x| \leq \log n$, $M_i(x)$ halts in $i|x|^i$ steps and accepts iff $x \in SAT_H$
- If no such M_i exists, $H(n) = \log\log n$

Defining $H(n)$

Let $M_1, M_2, \ldots$ be an enumeration of all TM's (can do this since TM's are countable)

$H(n)$

- Smallest $i \leq \log\log n$ s.t. for all x, $|x| \leq \log n$, $M_i(x)$ halts in $i|x|^i$ steps and accepts iff $x \in SAT_H$
- If no such M_i exists, $H(n) = \log\log n$

1. $H(n)$ is computable since can enumerate all short x

Defining $H(n)$

Let $M_1, M_2, \ldots$ be an enumeration of all TM's (can do this since TM's are countable)

$H(n)$

- Smallest $i \leq \log\log n$ s.t. for all x, $|x| \leq \log n$, $M_i(x)$ halts in $i|x|^i$ steps and accepts iff $x \in SAT_H$
- If no such M_i exists, $H(n) = \log\log n$

1. $H(n)$ is computable since can enumerate all short x
2. Claim: $SAT_H \in \mathcal{P}$ iff $H(n) < c$ for all n

Defining $H(n)$

Let $M_1, M_2, \ldots$ be an enumeration of all TM's (can do this since TM's are countable)

$H(n)$

- Smallest $i \leq \log\log n$ s.t. for all x, $|x| \leq \log n$, $M_i(x)$ halts in $i|x|^i$ steps and accepts iff $x \in SAT_H$
- If no such M_i exists, $H(n) = \log\log n$

1. $H(n)$ is computable since can enumerate all short x
2. Claim: $SAT_H \in \mathcal{P}$ iff $H(n) < c$ for all n
   ($\Rightarrow$) By definition of $\mathcal{P}$, there is machine M_k that decides SAT_H in kn^k steps so $H(n) = k$

Defining $H(n)$

Let $M_1, M_2, \ldots$ be an enumeration of all TM's (can do this since TM's are countable)

$H(n)$

- Smallest $i \leq \log\log n$ s.t. for all x, $|x| \leq \log n$, $M_i(x)$ halts in $i|x|^i$ steps and accepts iff $x \in SAT_H$
- If no such M_i exists, $H(n) = \log\log n$

1. $H(n)$ is computable since can enumerate all short x
2. Claim: $SAT_H \in \mathcal{P}$ iff $H(n) < c$ for all n
   ($\Rightarrow$) By definition of $\mathcal{P}$, there is machine M_k that decides SAT_H in kn^k steps so $H(n) = k$
   ($\Leftarrow$) If $H(n) < c$, then there is infinitely long stretch where $H(x) = i$. But, then M_i decides SAT_H.

Completing the proof

Claim

$SAT_H \in \mathcal{P}$ iff $H(n) < c$ for all n

Completing the proof

Claim

$SAT_H \in \mathcal{P}$ iff $H(n) < c$ for all n

$\phi 0 1^{n^c}$

1. $SAT_H \notin \mathcal{P}$:
   - Suppose it is in $\mathcal{P}$, then $H(n) < c$
   - Can reduce any SAT formula to SAT_H formula by padding with $H(n)$ 1s
   - But, SAT is $\mathcal{NP}$-Complete, contradiction!

Completing the proof

Claim

$SAT_H \in \mathcal{P}$ iff $H(n) < c$ for all n

1. $SAT_H \notin \mathcal{P}$:
   - Suppose it is in $\mathcal{P}$, then $H(n) < c$
   - Can reduce any SAT formula to SAT_H formula by padding with $H(n)$ 1s
   - But, SAT is $\mathcal{NP}$-Complete, contradiction!
2. SAT_H is not $\mathcal{NP}$-Complete

   $|\phi 01^{H(n)}| < n^c$

   - Assume it is, then $SAT \leq_p SAT_H$
   - Reduction maps ψ of length n to $\phi 01^{H(n)}$ of length n^c, but $H(n) \to \infty$ so this is super-poly in size of ϕ
   - Hence $|\phi| << n$, so have reduced solving long formula to solving a much shorter one.
   - Repeat this enough times to make $|\phi| = O(1)$ and solve.

Takeaway

If $\mathcal{P} \neq \mathcal{NP}$, then $\mathcal{NP}$-intermediate languages exist!

Outline

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Consider the following language:

UNSAT

$$\text{UNSAT} = \{\langle\phi\rangle \mid \phi \text{ is not satisfiable}\}$$

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Consider the following language:

UNSAT

$$\text{UNSAT} = \{\langle \phi \rangle \mid \phi \text{ is not satisfiable}\}$$

- For all possible assignments $w \in \{0,1\}^{|\phi|}$, $\phi(w) = 0$

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Consider the following language:

UNSAT

$$\text{UNSAT} = \{\langle\phi\rangle \mid \phi \text{ is not satisfiable}\}$$

- For all possible assignments $w \in \{0,1\}^{|\phi|}$, $\phi(w) = 0$
- Is this in $\mathcal{NP}$?

Are All Problems in $\mathcal{NP}$?

Question

Do all languages have poly-size proofs?

Consider the following language:

UNSAT

$$\text{UNSAT} = \{\langle\phi\rangle \mid \phi \text{ is not satisfiable}\}$$

- For all possible assignments $w \in \{0,1\}^{|\phi|}$, $\phi(w) = 0$
- Is this in $\mathcal{NP}$?
- We define complexity class co-$\mathcal{NP}$ to contain all such languages that are complements of languages in $\mathcal{NP}$

$\underline{\Phi}$ All ϕ, $|\phi| = n$ UNSAT = Φ / SAT

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{NP}$

$L \in \mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ there exists a witness w s.t. $V(x, w) = 1$

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{NP}$

$L \in \mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ there exists a witness w s.t. $V(x, w) = 1$

co-$\mathcal{NP}$

$L \in$ co-$\mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ for all w, $V(x, w) = 0$

$\mathcal{P}$, $\mathcal{NP}$ and co-$\mathcal{NP}$

$\mathcal{P}$

$L \in \mathcal{P}$ if there exists poly-time DTM M s.t $M(x) = [x \in L]$

$\mathcal{NP}$

$L \in \mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ there exists a witness w s.t. $V(x, w) = 1$

co-$\mathcal{NP}$

$L \in$ co-$\mathcal{NP}$ if there exists poly-time DTM V s.t. for $x \in L$ for all w, $V(x, w) = 0$

Question:

Can you prove that $x \in L$, when $L \in$co-$\mathcal{NP}$?

Proving that $x \in L$ for $L \in$co-$\mathcal{NP}$

The Problem

Suppose, I am given an input formula ϕ and I want to prove that ϕ is not satisfiable.

Proving that $x \in L$ for $L \in$co-$\mathcal{NP}$

The Problem

Suppose, I am given an input formula ϕ and I want to prove that ϕ is not satisfiable.

- It is widely believed that there is no poly-size, efficiently verifiable proof w that you could give for UNSAT

Proving that $x \in L$ for $L \in$co-$\mathcal{NP}$

The Problem

Suppose, I am given an input formula ϕ and I want to prove that ϕ is not satisfiable.

- It is widely believed that there is no poly-size, efficiently verifiable proof w that you could give for UNSAT
- $\mathcal{NP} \neq$ co-$\mathcal{NP}$

The Complexity Zoo

- There are many other complexity classes

The Complexity Zoo

- There are many other complexity classes
- We know some relationships between classes

The Complexity Zoo

- There are many other complexity classes
- We know some relationships between classes
- But, most big questions (e.g., $\mathcal{P} = \mathcal{NP}$, $\mathcal{NP} =$ co-$\mathcal{NP}$, etc.) are still not known!!!

The Complexity Zoo

- There are many other complexity classes
- We know some relationships between classes
- But, most big questions (e.g., $\mathcal{P} = \mathcal{NP}$, $\mathcal{NP} =$ co-$\mathcal{NP}$, etc.) are still not known!!!

Complexity Zoo

The complexity zoo (`https://complexityzoo.net/Complexity_Zoo`) now has 547 complexity classes.