CMPSC 465
Data Structures and Algorithms
Spring 2022

Instructor: Chunhao Wang

NP and Computational Hardness

NP and Computational Hardness

P, NP, and NP-completeness (Kleinberg-Tardos, Section 8.3, 8.4)

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch.

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem $X \in$ **NP** to circuit-SAT

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem X $\in$ **NP** to circuit-SAT

We use the fact that X has a polynomial-time certifier $B(\cdot, \cdot)$

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem X $\in$ **NP** to circuit-SAT
We use the fact that X has a polynomial-time certifier $B(\cdot, \cdot)$

Main idea: any algorithm on inputs of fixed length can be simulated by a circuit,

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem X $\in$ **NP** to circuit-SAT
We use the fact that X has a polynomial-time certifier $B(\cdot, \cdot)$

Main idea: any algorithm on inputs of fixed length can be simulated by a circuit, i.e., circuit outputs 1 if and only if algorithm outputs yes and if the algorithm takes polynomial time then the circuit has polynomial size

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem X $\in$ **NP** to circuit-SAT
We use the fact that X has a polynomial-time certifier $B(\cdot, \cdot)$

Main idea: any algorithm on inputs of fixed length can be simulated by a circuit, i.e., circuit outputs 1 if and only if algorithm outputs yes and if the algorithm takes polynomial time then the circuit has polynomial size

To decide if $s \in$ X, we check if there exists a string t of length $p(|s|)$ s.t. $B(s, t) =$ yes

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem X $\in$ **NP** to circuit-SAT
We use the fact that X has a polynomial-time certifier $B(\cdot, \cdot)$

Main idea: any algorithm on inputs of fixed length can be simulated by a circuit, i.e., circuit outputs 1 if and only if algorithm outputs yes and if the algorithm takes polynomial time then the circuit has polynomial size

To decide if $s \in$ X, we check if there exists a string t of length $p(|s|)$ s.t. $B(s, t) =$ yes
We transform $B(s, \cdot)$ into a circuit C_s with s "hardwired" and $p(|s|)$ inputs for possible t's

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem X $\in$ **NP** to circuit-SAT
We use the fact that X has a polynomial-time certifier $B(\cdot, \cdot)$

Main idea: any algorithm on inputs of fixed length can be simulated by a circuit, i.e., circuit outputs 1 if and only if algorithm outputs yes and if the algorithm takes polynomial time then the circuit has polynomial size

To decide if $s \in$ X, we check if there exists a string t of length $p(|s|)$ s.t. $B(s, t) =$ yes
We transform $B(s, \cdot)$ into a circuit C_s with s "hardwired" and $p(|s|)$ inputs for possible t's

Ask if C_s is satisfiable. If yes, there exists such t so $s \in$ X.

The Cook-Levin Theorem

Theorem (Cook-Levin)

circuit-SAT is **NP***-complete*

Proof sketch. We need to reduce every problem X $\in$ **NP** to circuit-SAT
We use the fact that X has a polynomial-time certifier $B(\cdot, \cdot)$

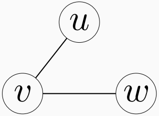
Main idea: any algorithm on inputs of fixed length can be simulated by a circuit, i.e., circuit outputs 1 if and only if algorithm outputs yes and if the algorithm takes polynomial time then the circuit has polynomial size

To decide if $s \in$ X, we check if there exists a string t of length $p(|s|)$ s.t. $B(s, t) =$ yes
We transform $B(s, \cdot)$ into a circuit C_s with s "hardwired" and $p(|s|)$ inputs for possible t's

Ask if C_s is satisfiable. If yes, there exists such t so $s \in$ X.
If no, there's such t that $B(s, t) =$ yes. So $s \notin$ X $\square$

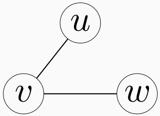
Example of such C_s

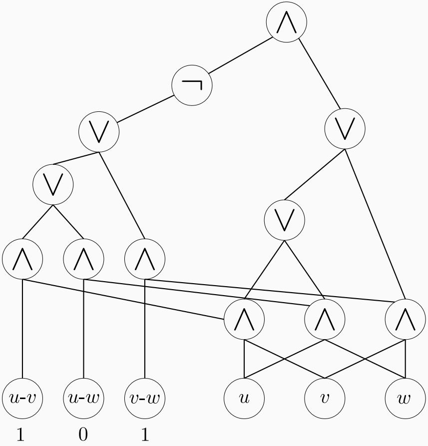
Decide if there's an IS of size 2



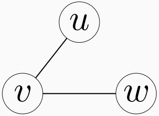
Example of such C_s

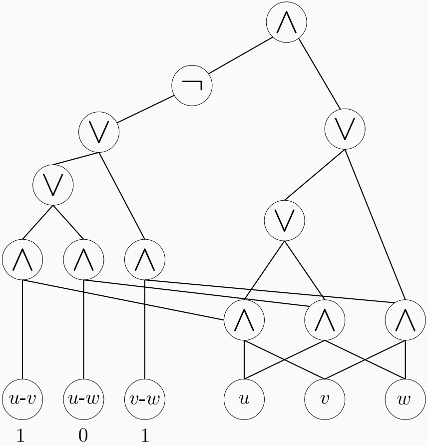
Decide if there's an IS of size 2





Decide if there's an IS of size 2

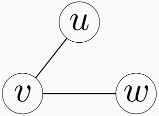


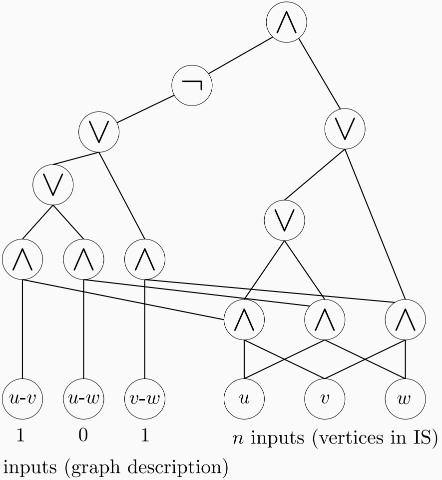


$\binom{n}{2}$ hardwired inputs (graph description)

Example of such C_s

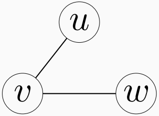
Decide if there's an IS of size 2

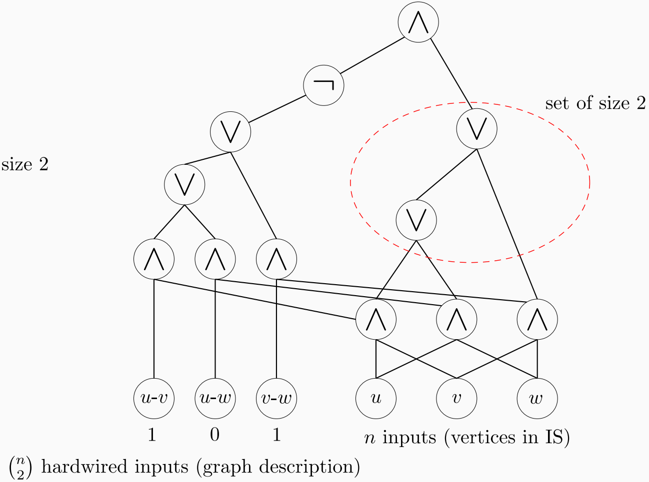




Example of such C_s

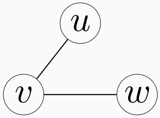
Decide if there's an IS of size 2

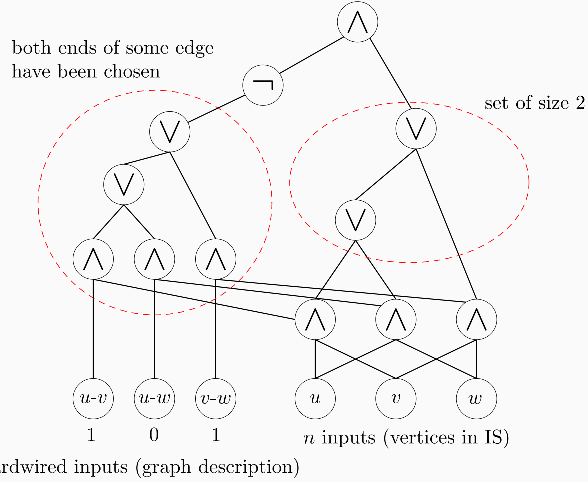




Example of such C_s

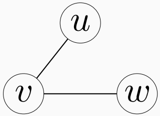
Decide if there's an IS of size 2

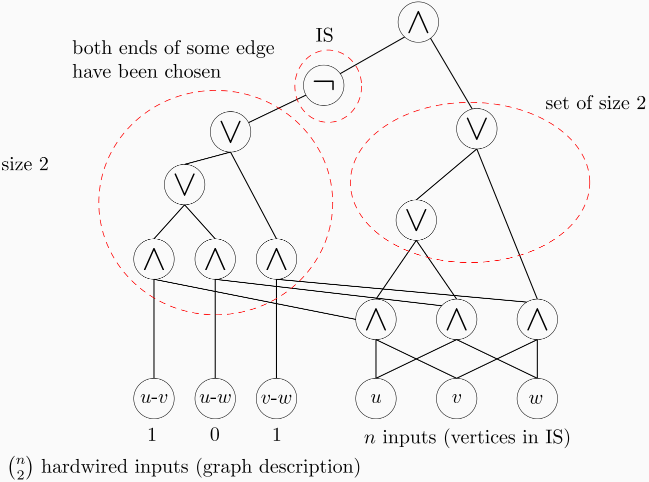




Example of such C_s

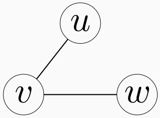
Decide if there's an IS of size 2

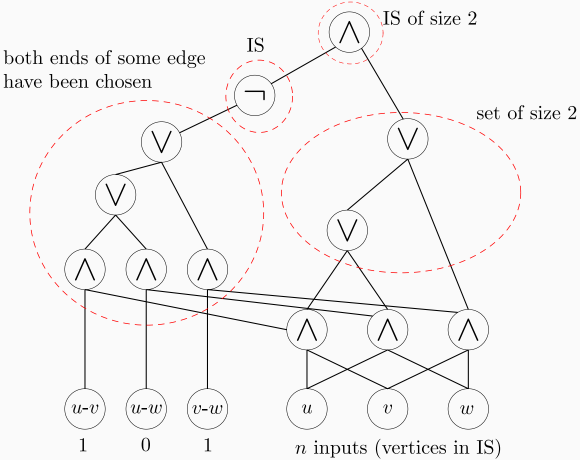




Example of such C_s

Decide if there's an IS of size 2





$\binom{n}{2}$ hardwired inputs (graph description)

Proving NP-completeness

Recipe for proving Y is **NP**-complete

Recipe for proving Y is **NP**-complete

Step 1: Prove $Y \in \mathbf{NP}$

Proving NP-completeness

Recipe for proving Y is **NP**-complete

Step 1: Prove Y $\in$ **NP**

Step 2: Choose an **NP**-complete problem X

Proving NP-completeness

Recipe for proving Y is **NP**-complete

Step 1: Prove Y $\in$ **NP**

Step 2: Choose an **NP**-complete problem X

Step 3: Prove X $\leq_P$ Y

Proving NP-completeness

Recipe for proving Y is **NP**-complete

Step 1: Prove $\mathrm{Y} \in \mathbf{NP}$

Step 2: Choose an **NP**-complete problem X

Step 3: Prove $\mathrm{X} \leq_P \mathrm{Y}$

Observation

If X *is* **NP***-complete,* $\mathrm{Y} \in \mathbf{NP}$*, and* $\mathrm{X} \leq_P \mathrm{Y}$*, then* Y *is* **NP***-complete*

Recipe for proving Y is **NP**-complete

Step 1: Prove Y $\in$ **NP**

Step 2: Choose an **NP**-complete problem X

Step 3: Prove X $\leq_P$ Y

Observation

If X *is* **NP***-complete,* Y $\in$ **NP***, and* X $\leq_P$ Y*, then* Y *is* **NP***-complete*

Proof.

Let W be any problem in **NP**. Then W $\leq_P$ X $\leq_P$ Y implies that W $\leq_P$ Y. Therefore, Y is **NP**-complete $\square$

3-SAT is NP-complete

Theorem

3-SAT is **NP***-complete*

3-SAT is NP-complete

Theorem

3-SAT is **NP***-complete*

Proof sketch.

3-SAT is NP-complete

Theorem

3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**.

3-SAT is NP-complete

Theorem

3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Theorem

3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,

3-SAT is NP-complete

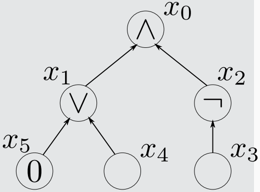
Theorem

3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,



3-SAT is NP-complete

Theorem

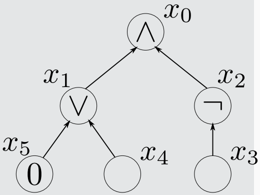
3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,

- $x_2 = \bar{x}_3$: add 2 clauses, $(x_2 \vee x_3)$, $(\bar{x}_2 \vee \bar{x}_3)$



3-SAT is NP-complete

Theorem

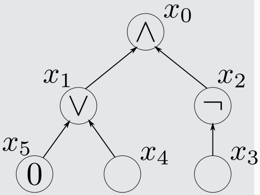
3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,

- $x_2 = \bar{x}_3$: add 2 clauses, $(x_2 \vee x_3)$, $(\bar{x}_2 \vee \bar{x}_3)$
- $x_1 = x_5 \vee x_4$: add 3 clauses, $(x_1 \vee \bar{x}_4)$, $(x_1 \vee \bar{x}_5)$, $(\bar{x}_1 \vee x_4 \vee x_5)$



3-SAT is NP-complete

Theorem

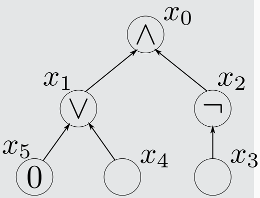
3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,

- $x_2 = \bar{x}_3$: add 2 clauses, $(x_2 \vee x_3)$, $(\bar{x}_2 \vee \bar{x}_3)$
- $x_1 = x_5 \vee x_4$: add 3 clauses, $(x_1 \vee \bar{x}_4)$, $(x_1 \vee \bar{x}_5)$, $(\bar{x}_1 \vee x_4 \vee x_5)$
- $x_0 = x_1 \wedge x_2$: add 3 clauses, $(\bar{x}_0 \vee x_1)$, $(\bar{x}_0 \vee x_2)$, $(x_0 \vee \bar{x}_1 \vee \bar{x}_2)$



3-SAT is NP-complete

Theorem

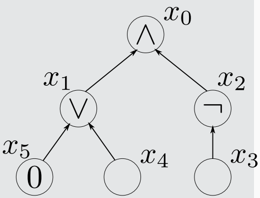
3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,

- $x_2 = \bar{x}_3$: add 2 clauses, $(x_2 \vee x_3)$, $(\bar{x}_2 \vee \bar{x}_3)$
- $x_1 = x_5 \vee x_4$: add 3 clauses, $(x_1 \vee \bar{x}_4)$, $(x_1 \vee \bar{x}_5)$, $(\bar{x}_1 \vee x_4 \vee x_5)$
- $x_0 = x_1 \wedge x_2$: add 3 clauses, $(\bar{x}_0 \vee x_1)$, $(\bar{x}_0 \vee x_2)$, $(x_0 \vee \bar{x}_1 \vee \bar{x}_2)$
- hardwired input $x_5 = 0$: add clause $(\bar{x}_5)$



3-SAT is NP-complete

Theorem

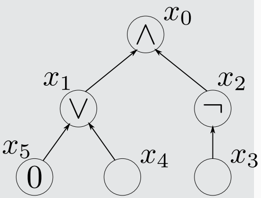
3-SAT is **NP***-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,

- $x_2 = \bar{x}_3$: add 2 clauses, $(x_2 \vee x_3)$, $(\bar{x}_2 \vee \bar{x}_3)$
- $x_1 = x_5 \vee x_4$: add 3 clauses, $(x_1 \vee \bar{x}_4)$, $(x_1 \vee \bar{x}_5)$, $(\bar{x}_1 \vee x_4 \vee x_5)$
- $x_0 = x_1 \wedge x_2$: add 3 clauses, $(\bar{x}_0 \vee x_1)$, $(\bar{x}_0 \vee x_2)$, $(x_0 \vee \bar{x}_1 \vee \bar{x}_2)$
- hardwired input $x_5 = 0$: add clause $(\bar{x}_5)$
- output: $x_0 = 1$: add clause (x_0)



3-SAT is NP-complete

Theorem

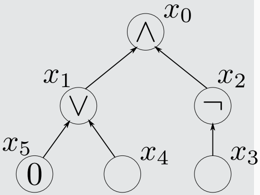
*3-SAT is **NP**-complete*

Proof sketch.

We have seen that 3-SAT is in **NP**. Now we show circuit-SAT $\leq_P$ 3-SAT

Given a circuit, create a 3-SAT variable x_i for each circuit element i, e.g.,

- $x_2 = \bar{x}_3$: add 2 clauses, $(x_2 \vee x_3)$, $(\bar{x}_2 \vee \bar{x}_3)$
- $x_1 = x_5 \vee x_4$: add 3 clauses, $(x_1 \vee \bar{x}_4)$, $(x_1 \vee \bar{x}_5)$, $(\bar{x}_1 \vee x_4 \vee x_5)$
- $x_0 = x_1 \wedge x_2$: add 3 clauses, $(\bar{x}_0 \vee x_1)$, $(\bar{x}_0 \vee x_2)$, $(x_0 \vee \bar{x}_1 \vee \bar{x}_2)$
- hardwired input $x_5 = 0$: add clause $(\bar{x}_5)$
- output: $x_0 = 1$: add clause (x_0)



Turn clauses of length < 3 into clauses of length exactly 3 $\square$

Other NP-complete problems

From last lecture:

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

- Hamilton cycle. Given $G = (V, E)$ undirected. Is there a simple cycle that contains every vertex in V?

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

- Hamilton cycle. Given $G = (V, E)$ undirected. Is there a simple cycle that contains every vertex in V?
  3-SAT $\leq_P$ Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

- Hamilton cycle. Given $G = (V, E)$ undirected. Is there a simple cycle that contains every vertex in V?
  3-SAT $\leq_P$ Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle
- Travelling Salesman (TSP)

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

- Hamilton cycle. Given $G = (V, E)$ undirected. Is there a simple cycle that contains every vertex in V?
  3-SAT $\leq_P$ Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle
- Travelling Salesman (TSP)
  Given a set of cities, distances $d(u, v)$, a number D, is there a tour of length $\leq D$?

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

- Hamilton cycle. Given $G = (V, E)$ undirected. Is there a simple cycle that contains every vertex in V?
  3-SAT $\leq_P$ Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle
- Travelling Salesman (TSP)
  Given a set of cities, distances $d(u, v)$, a number D, is there a tour of length $\leq D$?
  Hamiltonian Cycle $\leq_P$ TSP

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

- Hamilton cycle. Given $G = (V, E)$ undirected. Is there a simple cycle that contains every vertex in V?
  3-SAT $\leq_P$ Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle
- Travelling Salesman (TSP)
  Given a set of cities, distances $d(u, v)$, a number D, is there a tour of length $\leq D$?
  Hamiltonian Cycle $\leq_P$ TSP

and many more...

Other NP-complete problems

From last lecture:

- Independent Set is **NP**-complete
- Vertex Cover is **NP**-complete

Other **NP**-complete problems:

- Hamilton cycle. Given $G = (V, E)$ undirected. Is there a simple cycle that contains every vertex in V?
  3-SAT $\leq_P$ Directed Hamiltonian Cycle $\leq_P$ Hamiltonian Cycle
- Travelling Salesman (TSP)
  Given a set of cities, distances $d(u, v)$, a number D, is there a tour of length $\leq D$?
  Hamiltonian Cycle $\leq_P$ TSP

and many more...

Want to learn more about this topic? Take CMPSC 464