Overview

- Navigating the world of P and NP
  - 2SAT

- Beyond decision problems: NP-hardness
  - NP-hard reductions

- Acknowledgements
Subtleties

Similar looking problems, wildly different complexity:

- **Hamilton Cycle:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES, iff there is a cycle that visits every vertex exactly once
Subtleties

Similar looking problems, wildly different complexity:

- **Hamilton Cycle:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES, iff there is a *cycle* that visits every *vertex* exactly once

- **Euler Tour:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES iff there is *closed walk* traversing every *edge* exactly once
Subtleties

Similar looking problems, wildly different complexity:

- **Hamilton Cycle:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES, iff there is a cycle that visits every vertex exactly once

- **Euler Tour:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES iff there is closed walk traversing every edge exactly once

- Hamilton Cycle is NP-complete, whereas Euler tour has a linear time algorithm (depth-first search).

---

**Theorem (Euler’s theorem)**

- $G$ has eulerian tour iff every vertex has even degree.
- $G$ has eulerian path iff exactly 2 vertices have odd degree.
Subtleties

Similar looking problems, wildly different complexity:

- **Hamilton Cycle:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES, iff there is a *cycle* that visits every *vertex* exactly once

- **Euler Tour:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES iff there is *closed walk* traversing every *edge* exactly once

- Hamilton Cycle is NP-complete, whereas Euler tour has a linear time algorithm (depth-first search).

---

**Theorem (Euler’s theorem)**

*G* has eulerian tour iff every vertex has even degree.

*G* has eulerian path iff exactly 2 vertices have odd degree.

- Similar situation for hamiltonian path vs eulerian path!
Subtleties

Similar looking problems, wildly different complexity:

- **Hamilton Cycle:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES, iff there is a *cycle* that visits every *vertex* exactly once

- **Euler Tour:**
  - **Input:** undirected graph $G(V, E)$
  - **Output:** YES iff there is *closed walk* traversing every *edge* exactly once

- Hamilton Cycle is NP-complete, whereas Euler tour has a linear time algorithm (depth-first search).

---

**Theorem (Euler’s theorem)**

- $G$ has eulerian tour iff every vertex has even degree.
- $G$ has eulerian path iff exactly 2 vertices have odd degree.

- Similar situation for hamiltonian path vs eulerian path!
- In general, we need to be careful when distinguishing or making reductions between problems.
Navigating the world of P and NP
  - 2SAT

Beyond decision problems: NP-hardness
  - NP-hard reductions

Acknowledgements
2SAT

- **2SAT**
  - **Input:** 2CNF $\varphi(x_1, \ldots, x_n)$
  - **Output:** YES $\iff \varphi$ is satisfiable
2SAT

- **2SAT**
  - **Input:** 2CNF $\varphi(x_1, \ldots, x_n)$
  - **Output:** YES $\iff$ $\varphi$ is satisfiable

**Theorem**

2SAT is in P
2SAT

- **2SAT**
  - **Input:** 2CNF $\varphi(x_1, \ldots, x_n)$
  - **Output:** YES $\iff \varphi$ is satisfiable

**Theorem**

2SAT is in P

**Proof:** “implication graph”

Example: $(x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_3) \land (x_2 \lor \overline{x_3}) \land (x_1 \lor x_2)$
2SAT

- **2SAT**
  - **Input:** 2CNF $\varphi(x_1, \ldots, x_n)$
  - **Output:** YES $\iff$ $\varphi$ is satisfiable

**Theorem**

2SAT is in P

**Proof:** “implication graph”

Example: $(x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_3) \land (x_2 \lor \overline{x_3}) \land (x_1 \lor x_2)$

Let $G_\varphi([2n], E)$ be the directed graph generated by the implication graph process.
2SAT

- **2SAT**
  - **Input:** 2CNF $\varphi(x_1, \ldots, x_n)$
  - **Output:** YES $\iff \varphi$ is satisfiable

**Theorem**

2SAT is in P

**Proof:** “implication graph”

Example: $(x_1 \lor \overline{x}_2) \land (\overline{x}_1 \lor x_3) \land (x_2 \lor \overline{x}_3) \land (x_1 \lor x_2)$

Let $G_\varphi([2n], E)$ be the directed graph generated by the implication graph process.

Run BFS or DFS from each literal $y$, and call it bad if for some $i \in [n]$, the BFS from $y$ visits both $x_i, \overline{x}_i$
2SAT

- **2SAT**
  - **Input**: 2CNF \( \varphi(x_1, \ldots, x_n) \)
  - **Output**: YES \( \iff \varphi \) is satisfiable

Theorem

2SAT is in P

- **Proof**: “implication graph”

  Example: \((x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_3) \land (x_2 \lor \overline{x_3}) \land (x_1 \lor x_2)\)

Let \( G_\varphi([2n], E) \) be the directed graph generated by the implication graph process.

Run BFS or DFS from each literal \( y \), and call it *bad* if for some \( i \in [n] \), the BFS from \( y \) visits both \( x_i, \overline{x_i} \)

If for some \( i \in [n] \), both \( x_i \) and \( \overline{x_i} \) are bad, then return NO. Otherwise, return YES.
Navigating the world of P and NP
  - 2SAT

Beyond decision problems: NP-hardness
  - NP-hard reductions

Acknowledgements
NP-hardness

- Often times we want to know whether a non-decision problem (say optimization problem or search problem) is hard
NP-hardness

- Often times we want to know whether a non-decision problem (say optimization problem or search problem) is hard.
- In these cases, since the problems are not decision problems, they will not belong to NP.
NP-hardness

- Often times we want to know whether a *non-decision problem* (say optimization problem or search problem) is hard.
- In these cases, since the problems are not decision problems, they will not belong to NP.
- However, can still apply our original reasoning:
  - want to prove that problem $B$ (non-decision problem) is hard.
  - Can select an NP-complete problem $A$ and show that “if we can solve $B$ efficiently, then we can solve $A$ efficiently”.
  - In other words:

$$A \leq_T B$$
NP-hardness

- Often times we want to know whether a *non-decision problem* (say optimization problem or search problem) is hard.
- In these cases, since the problems are not decision problems, they will not belong to NP.
- However, can still apply our original reasoning:
  - want to prove that problem $B$ (non-decision problem) is hard.
  - Can select an NP-complete problem $A$ and show that “if we can solve $B$ efficiently, then we can solve $A$ efficiently”
  - In other words:
    \[ A \leq_T B \]
- The above is our definition of **NP-hardness**:
  Problem $B$ is *NP-hard* if there is NP-complete problem $A$ such that $A \leq_T B$. 
Examples of NP-hard problems

- MAX-CLIQUE
  - **Input:** graph $G(V, E)$
  - **Output:** maximum size of a clique in $G$
Examples of NP-hard problems

- **MAX-CLIQUE**
  - **Input**: graph $G(V, E)$
  - **Output**: maximum size of a clique in $G$

- **MIS**:
  - **Input**: graph $G(V, E)$
  - **Output**: maximum independent set in $G$
Examples of NP-hard problems

- MAX-CLIQUE
  - **Input:** graph $G(V, E)$
  - **Output:** maximum size of a clique in $G$

- MIS:
  - **Input:** graph $G(V, E)$
  - **Output:** maximum independent set in $G$

- MIN-Vertex-Cover:
  - **Input:** graph $G(V, E)$
  - **Output:** size of minimum vertex cover in $G$
Examples of NP-hard problems

- **MAX-CLIQUE**
  - **Input:** graph $G(V, E)$
  - **Output:** maximum size of a clique in $G$

- **MIS:**
  - **Input:** graph $G(V, E)$
  - **Output:** maximum independent set in $G$

- **MIN-Vertex-Cover:**
  - **Input:** graph $G(V, E)$
  - **Output:** size of minimum vertex cover in $G$

- **TSP-OPT:**
  - **Input:** complete graph $G(V, E, d)$ where $d : E \rightarrow \mathbb{R}_{\geq 0}$
  - **Output:** hamiltonian cycle in $G$ of minimum total distance
Navigating the world of P and NP
  - 2SAT

Beyond decision problems: NP-hardness
  - NP-hard reductions

Acknowledgements
Non-Trivial NP-hardness reduction

- (unweighted) **MAX-CUT**
  - **Input:** graph $G(V, E)$
  - **Output:** a cut $S \subseteq V$ with maximum $|\delta(S)|$
Non-Trivial NP-hardness reduction

- (unweighted) **MAX-CUT**
  - **Input:** graph $G(V, E)$
  - **Output:** a cut $S \subseteq V$ with maximum $|\delta(S)|$

**Theorem**

MAX-CUT is NP-hard
Non-Trivial NP-hardness reduction

**Theorem**

\[ \text{MAX-CUT is NP-hard} \]

**Proof:** reduction from MIS. Let \( G(V, E) \) be the input graph.
Non-Trivial NP-hardness reduction

**Theorem**

MAX-CUT is NP-hard

**Proof:** reduction from MIS. Let $G(V, E)$ be the input graph.

- **Vertex gadget:**
  - add vertex $x$
  - for each $v \in V$, add edge $\{x, v\}$
Non-Trivial NP-hardness reduction

Theorem

MAX-CUT is NP-hard

**Proof:** reduction from MIS. Let $G(V, E)$ be the input graph.

- **Vertex gadget:**
  - add vertex $x$
  - for each $v \in V$, add edge $\{x, v\}$

- **Edge gadget:** for each edge $e = \{u, v\}$
  - add vertices $u_e, v_e$,
  - and edges: $\{x, u_e\}, \{x, v_e\}, \{u, u_e\}, \{v, v_e\}, \{u_e, v_e\}$,
Theorem

MAX-CUT is NP-hard

Proof: reduction from MIS. Let $G(V, E)$ be the input graph.

- **Vertex gadget:**
  - add vertex $x$
  - for each $v \in V$, add edge $\{x, v\}$

- **Edge gadget:** for each edge $e = \{u, v\}$
  - add vertices $u_e, v_e$,
  - and edges: $\{x, u_e\}, \{x, v_e\}, \{u, u_e\}, \{v, v_e\}, \{u_e, v_e\}$,

Edge gadget $H_e$:
Non-Trivial NP-hardness reduction

**Theorem**

**MAX-CUT is NP-hard**

**Proof:** reduction from MIS. Let \( G(V, E) \) be the input graph.

- **Vertex gadget:**
  - add vertex \( x \)
  - for each \( v \in V \), add edge \( \{x, v\} \)

- **Edge gadget:** for each edge \( e = \{u, v\} \)
  - add vertices \( u_e, v_e \),
  - and edges: \( \{x, u_e\}, \{x, v_e\}, \{u, u_e\}, \{v, v_e\}, \{u_e, v_e\}, \)

- Edge gadget \( H_e \):
  - Let \( H(U, F) \) be graph given by:
    - \( U = V \uplus \{x\} \uplus \{u_e, v_e\}_{u,v}=:e\in E \)
    - \( F = \{\{x, w\}\}_{w\in U\setminus\{x\}} \uplus \{\{u_e, v_e\}\}_{e\in E} \uplus \{\{u, u_e\}, \{v, v_e\}\}_{u,v}=:e\in E \)

Note that \( H \) does not have any edges from \( G \)
Claim 1: $G$ contains independent set $I \subseteq V$ with $|I| = k \Rightarrow$ there is cut $S \subseteq U$ in $H$ such that

$$|\delta(S)| \geq k + 4 \cdot |E|$$
Claim 1: \( G \) contains independent set \( I \subset V \) with \( |I| = k \Rightarrow \) there is cut \( S \subset U \) in \( H \) such that

\[
|\delta(S)| \geq k + 4 \cdot |E|
\]

1. Start with \( S = I \).
2. For each edge \( e = \{u, v\} \in E \) do
   - if \( u \in I, v \notin I \), then add \( v \) to \( S \)
   - if \( u \notin I, v \in I \), then add \( u \) to \( S \)
   - if \( u, v \notin I \), then add \( u, v \) to \( S \).

   In all above cases, add four of five edge gadget \( H_e \) edges
Proof of Correctness - Part 1

- **Claim 1:** $G$ contains independent set $I \subset V$ with $|I| = k \Rightarrow$ there is cut $S \subset U$ in $H$ such that

$$|\delta(S)| \geq k + 4 \cdot |E|$$

1. Start with $S = I$.
2. For each edge $e = \{u, v\} \in E$ do
   - if $u \in I$, $v \notin I$, then add $v_e$ to $S$
   - if $u \notin I$, $v \in I$, then add $u_e$ to $S$
   - if $u, v \notin I$, then add $u_e, v_e$ to $S$.

   In all above cases, add four of five edge gadget $H_e$ edges

Analyzing the cut given by $S$:
- For every $w \in I$, the edge $\{x, w\}$ is cut by $S$
Proof of Correctness - Part 1

- **Claim 1:** $G$ contains independent set $I \subseteq V$ with $|I| = k \Rightarrow$ there is cut $S \subseteq U$ in $H$ such that

\[ |\delta(S)| \geq k + 4 \cdot |E| \]

1. Start with $S = I$.
2. For each edge $e = \{u, v\} \in E$ do
   - if $u \in I$, $v \notin I$, then add $v_e$ to $S$
   - if $u \notin I$, $v \in I$, then add $u_e$ to $S$
   - if $u, v \notin I$, then add $u_e, v_e$ to $S$.

   In all above cases, add four of five edge gadget $H_e$ edges

Analyzing the cut given by $S$:

- For every $w \in I$, the edge $\{x, w\}$ is cut by $S$
- For every edge $\{u, v\} =: e \in E$, exactly 4 edges of $H_e$ are cut.
Proof of Correctness - Part 2

- **Claim 2:** Given cut $S \subset U$ in $H$ with

  $$|\delta(S)| \geq k + 4 \cdot |E|$$

  then $G$ contains independent set $I \subset V$ of size $\geq k$. 
Claim 2: Given cut $S \subset U$ in $H$ with
\[ |\delta(S)| \geq k + 4 \cdot |E| \]
then $G$ contains independent set $I \subset V$ of size $\geq k$.

W.l.o.g. can assume $x \notin S$ (otherwise take complement $V \setminus S$)
Claim 2: Given cut $S \subset U$ in $H$ with

$$|\delta(S)| \geq k + 4 \cdot |E|$$

then $G$ contains independent set $I \subset V$ of size $\geq k$.

- W.l.o.g. can assume $x \notin S$ (otherwise take complement $V \setminus S$)
- Let $I = S \cap V$ (vertices in $G$)
Proof of Correctness - Part 2

- **Claim 2:** Given cut $S \subset U$ in $H$ with
  \[ |\delta(S)| \geq k + 4 \cdot |E| \]
  then $G$ contains independent set $I \subset V$ of size $\geq k$.
  - W.l.o.g. can assume $x \notin S$ (otherwise take complement $V \setminus S$)
  - Let $I = S \cap V$ (vertices in $G$)
  - If $u, v \in I$ are s.t. $\{u, v\} =: e \in E$, then $S$ cuts at most 3 edges of $H_e$
Proof of Correctness - Part 2

Claim 2: Given cut $S \subset U$ in $H$ with

$$|\delta(S)| \geq k + 4 \cdot |E|$$

then $G$ contains independent set $I \subset V$ of size $\geq k$.

- W.l.o.g. can assume $x \notin S$ (otherwise take complement $V \setminus S$)
- Let $I = S \cap V$ (vertices in $G$)
- If $u, v \in I$ are s.t. $\{u, v\} =: e \in E$, then $S$ cuts at most 3 edges of $H_e$
- Otherwise, we saw in part 1 how to get 4 edges of $H_e$ across the cut.
Claim 2: Given cut $S \subset U$ in $H$ with

$$|\delta(S)| \geq k + 4 \cdot |E|$$

then $G$ contains independent set $I \subset V$ of size $\geq k$.

- W.l.o.g. can assume $x \not\in S$ (otherwise take complement $V \setminus S$)
- Let $I = S \cap V$ (vertices in $G$)
- If $u, v \in I$ are s.t. $\{u, v\} =: e \in E$, then $S$ cuts at most 3 edges of $H_e$
- Otherwise, we saw in part 1 how to get 4 edges of $H_e$ across the cut.
- Letting $e(I)$ be number of edges between elements of $I$ in $G$:

$$|\delta(S)| = |I| + \sum_{e \in E} |\delta_{H_e}(S)| \leq |I| + 3e(I) + 4(|E| - e(I)) = |I| + 4|E| - e(I)$$
Proof of Correctness - Part 2

- **Claim 2:** Given cut $S \subset U$ in $H$ with

$$|\delta(S)| \geq k + 4 \cdot |E|$$

then $G$ contains independent set $I \subset V$ of size $\geq k$.

- W.l.o.g. can assume $x \notin S$ (otherwise take complement $V \setminus S$)
- Let $I = S \cap V$ (vertices in $G$)
- If $u, v \in I$ are s.t. $\{u, v\} = e \in E$, then $S$ cuts at most 3 edges of $H_e$
- Otherwise, we saw in part 1 how to get 4 edges of $H_e$ across the cut.
- Letting $e(I)$ be number of edges between elements of $I$ in $G$:

$$|\delta(S)| = |I| + \sum_{e \in E} |\delta_{H_e}(S)| \leq |I| + 3e(I) + 4(|E| - e(I)) = |I| + 4|E| - e(I)$$

- As $|\delta(S)| \geq k + 4|E|$, we have

$$|I| \geq k + e(I)$$
Proof of Correctness - Part 2

- **Claim 2:** Given cut \( S \subset U \) in \( H \) with

\[
|\delta(S)| \geq k + 4 \cdot |E|
\]

then \( G \) contains independent set \( I \subset V \) of size \( \geq k \).

- W.l.o.g. can assume \( x \notin S \) (otherwise take complement \( V \setminus S \))
- Let \( I = S \cap V \) (vertices in \( G \))
- If \( u, v \in I \) are s.t. \( \{u, v\} =: e \in E \), then \( S \) cuts at most 3 edges of \( H_e \)
- Otherwise, we saw in part 1 how to get 4 edges of \( H_e \) across the cut.
- Letting \( e(I) \) be number of edges between elements of \( I \) in \( G \):

\[
|\delta(S)| = |I| + \sum_{e \in E} |\delta_{H_e}(S)| \leq |I| + 3e(I) + 4(|E| - e(I)) = |I| + 4|E| - e(I)
\]

- As \( |\delta(S)| \geq k + 4|E| \), we have

\[
|I| \geq k + e(I)
\]

- So for each \( u, v \in I \) with \( \{u, v\} \in E \), we can afford to remove one of the endpoints from \( S \), decreasing \( |I| \) by one. After \( e(I) \) removals, get our independent set.
Acknowledgement

Based on

- [Erickson 2019, Chapter 12]
- Debmalya’s Lecture 22
  
References I

Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford (2009)
 MIT Press

Dasgupta, Sanjay and Papadimitriou, Christos and Vazirani, Umesh (2006)
Algorithms

Erickson, Jeff (2019)
Algorithms
https://jeffe.cs.illinois.edu/teaching/algorithms/

Kleinberg, Jon and Tardos, Eva (2006)
Algorithm Design.
 Addison Wesley