• Easy problems vs. Hard Problems
• P vs. NP
• Cook-Levin Theorem

- Easy problems vs. hard problems

- Easy problem: solvable in polynomial time \((O(n), \Theta(n \log n), O(n^2))\)

- Hard problem: problems that (we believe) cannot be solved in polynomial time.

- Eulerian path vs. Hamiltonian path

- Given graph \(G\) (undirected), decide whether there is a path from \(s\) to \(t\) that uses every
  (a) edge exactly once.
  (b) vertex

(a) Eulerian path  (b) Hamiltonian path

- Claim: There is an Eulerian path iff graph \(G\) is connected and every vertex (except for \(s, t\)) has even degree, \(s, t\) have odd degree.
  \(\Rightarrow\) Eulerian Path is easy.

- Hamiltonian path is hard (formalized later).

- Decision Problem

- Problems that require a yes/no answer.

  decision problems vs. optimization problems

  Is there a spanning tree with cost \(\leq 100\)?

  \(\implies\) MST

  - Approach: classify decision problems into complexity classes.
- P: easy problems: can be solved in polynomial time.

- NP (nondeterministic polynomial time). Given an input \( X \), there is a polynomial time verifier \( V \) s.t. given a "proof" \( C \), if answer to \( X \) is yes, there is a proof \( C \) s.t. \( V(x,c) = true \) if answer to \( X \) is no, then for any proof \( C \), \( V(x,c) = false \).

- NP: easy to verify whether a solution is correct, but potentially hard to find a solution.

- P \( \leq \) NP (need no proof, \( V \) is just the algorithm)

- co NP: "negation" of NP, can verify the answer is no.

- Example: Whether \( X \) is a prime, can prove \( X \) is a prime by writing \( X = y \cdot z \), \( y, z \neq 1 \)

  "not Hamiltonian path"

- P vs. NP: Is \( P = NP \)?

  - Solving homework problems / finding a proof are in NP.

  - Harder problems

  - Example: chess \( (PSPACE) \)

- Q: How do we decide problem A is harder than problem B?

  - A: Reductions

  - We say there is a polynomial time reduction from B to A, if for any instance \( X \) of B, there is a polynomial time algorithm \( f \) that maps \( x \) to \( y = f(x) \). \( y \) is an instance of A, and answer for \( y \) is the same as answer for \( x \).

  - In this case we say A is harder (or at least not simpler) than B.
\((A \geq B)\)

\[
\begin{array}{c}
A \\
\Rightarrow \\
f(x) < f(x)
\end{array}
\]

If we have an algorithm for \(A\) (\(Q\)), then \(Q(f(x))\) is an algorithm for \(B\).

- NP-hardness: Problem \(A\) is NP-hard, if for any NP problem \(B\), can reduce \(B\) to \(A\).

\[A \geq B \text{ for any } B \in \text{NP}\]

If \(A\) is also in NP, then we call \(A\) NP-complete.

- If \(A\) can be solved in poly time, then \(P = \text{NP}\).
- Hamiltonian Path is an NP complete problem.

- Cook-Levin Theorem: For any problem \(L\) in NP, there is a polynomial time reduction from \(L\) to CIRCUIT-SAT (SAT, 3-SAT).