Computational Complexity

Lecture 2: NP-completeness and the Cook-Levin Theorem

Ronald de Haan
me@ronalddehaan.eu

University of Amsterdam

April 5, 2024

Recap

What we saw last time..

- Decision problems
- The complexity class P
- Nondeterministic Turing machines
- More complexity classes: EXP, NP, coNP
- Polynomial-time reductions
- NP-hardness and NP-completeness

What will we do today?

- Prove that NP-complete problems exist ☺
- The Cook-Levin Theorem
- Concrete reductions between problems
- Search vs. decision problems

Representing Turing machines as (binary) strings

- We can encode Turing machines into binary strings, such that:

  1. each string $s \in \{0,1\}^*$ represents some Turing machine $\mathbb{M}$
  2. each Turing machine $\mathbb{M}$ is represented by infinitely many strings $s \in \{0,1\}^*$
  3. given a TM $\mathbb{M}$, we can efficiently compute a string s that represents $\mathbb{M}$

- Idea:

  - Write out the tuple (Γ, Q, δ), together with starting and halting states, in an appropriate alphabet, and then encode into binary
  - Allow padding (cf. comments in programming languages)

Efficient universal Turing machine

Proposition

There exists a TM $\mathbb{U}$ such that for every $x, s \in \{0,1\}^*$ it holds that $\mathbb{U}(x, s) = \mathbb{M}_s(x)$, where $\mathbb{M}_s$ is the TM represented by the string s.

Moreover, if $\mathbb{M}_s$ halts on x in time T, then $\mathbb{U}(x, s)$ halts in time $C \cdot T \log T$, where C depends only on s (and not on x).

- $\mathbb{U}$ is an efficient universal Turing machine: it can simulate other TMs in an efficient way.

Our first NP-complete problem

Definition

The decision problem TM-SAT is defined as follows:

$$\text{TM-SAT} = \{\ (\alpha, x, 1^n, 1^t) \mid \textit{there exists } u \in \{0,1\}^n \textit{ such that } \mathbb{M}_\alpha \textit{ outputs 1 on input } (x, u) \textit{ within } t \textit{ steps}\ \}$$

Or, described in a different format:

Input: *A binary string* α*, a binary string* x*, a unary string* 1^n*, and a unary string* 1^t*.*

Question: *Does there exist a binary string* $u \in \{0,1\}^n$ *such that* $\mathbb{M}_\alpha$ *outputs 1 on input* (x, u) *within* t *steps?*

TM-SAT is NP-complete

Proposition

TM-SAT is NP-complete

Proof (sketch).

Membership in NP: guess u, and verify by simulating $\mathbb{M}_\alpha$.

NP-hardness:

Take an arbitrary $L \in \mathsf{NP}$. Then there exists a polynomial p and a TM $\mathbb{M}$ such that for all $x \in \{0,1\}^*$ there exists some $u \in \{0,1\}^{p(|x|)}$ such that $\mathbb{M}(x,u) = 1$ iff $x \in L$.

Let q be a polynomial bounding the running time of $\mathbb{M}$.

Take the reduction R from L to TM-SAT where:
$R(x) = (\mathrm{repr}(\mathbb{M}), x, 1^{p(|x|)}, 1^{q(|x|+p(|x|))})$ $\square$

Propositional logic

- Propositional logic formulas φ are built from *atomic propositions* $x_1, x_2, \ldots$ using Boolean operators $\wedge, \vee, \rightarrow, \neg$.
- For example, $\varphi_1 = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3)$.
- A *truth assignment* is a function $\alpha : \mathrm{Vars}(\varphi) \rightarrow \{0, 1\}$ that maps the atomic propositions to 1 (true) or 0 (false).
- For example, $\alpha_1 = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$.
- The truth $\varphi[\alpha]$ of a formula φ under a truth assignment α is defined inductively, following the standard meaning of the operators.
- For example, $\varphi_1[\alpha_1] = 0$.

Propositional satisfiability

Definition

The decision problem Formula-SAT is defined as follows:

$$\text{Formula-SAT} = \{\, \varphi \mid \varphi \textit{ is a propositional logic formula and there exists a satisfying truth assignment } \alpha \textit{ for } \varphi \,\}$$

Or, described in a different format:

Input:	*A propositional logic formula* φ.
Question:	*Is* φ *satisfiable?*

Propositional satisfiability of CNF formulas

Definition

The decision problem CNF-SAT is defined as follows:

$$\text{CNF-SAT} = \{\ \varphi \mid \ \varphi \textit{ is a propositional logic formula in CNF and there exists a satisfying truth assignment } \alpha \textit{ for } \varphi\ \}$$

Or, described in a different format:

Input: *A propositional logic formula φ in CNF.*

Question: *Is φ satisfiable?*

- Conjunctive Normal Form (CNF): a conjunction of disjunctions of literals.
- For example: $\varphi_1 = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$

The Cook-Levin Theorem

Theorem (Cook 1971, Levin 1969)

CNF-SAT *is* NP*-complete.*

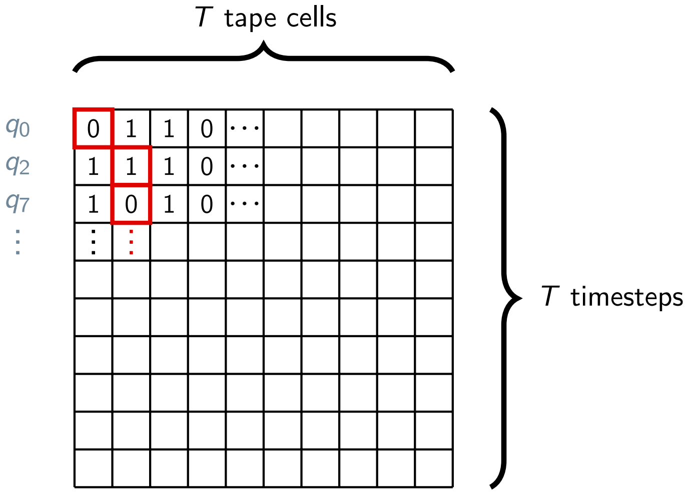
Polynomial-time computation in a picture

For a single-tape TM

For each $t, i \in \{1, \ldots, T\}$
and each $\gamma \in \Gamma$:
introduce a proposition $\boldsymbol{c_{t,i,\gamma}}$

For each $t, i \in \{1, \ldots, T\}$:
introduce a proposition $\boldsymbol{h_{t,i}}$

For each $t \in \{1, \ldots, T\}$
and each $q \in Q$:
introduce a proposition $\boldsymbol{s_{t,q}}$



Proof of Cook-Levin Theorem

- Take an arbitrary $L \in \mathrm{NP}$. Then there exist polynomials $p, q : \mathbb{N} \to \mathbb{N}$ and a TM $\mathbb{M}$ running in time $q(n)$ such that for each $x \in \{0,1\}^*$:

  $x \in L$ if and only if there exists $u \in \{0,1\}^{p(|x|)}$ such that $\mathbb{M}(x, u) = 1$.

- W.l.o.g., assume that $\mathbb{M}$ is single-tape and that q_{acc} and q_{rej} are 'sinks'

- Take $T = q(|x| + p(|x|))$. That is, $T \geq$ running time of $\mathbb{M}(x, u)$.

- We will construct a formula φ (over the variables $c_{t,i,\gamma}$, $h_{t,i}$, $s_{t,q}$) that is satisfiable if and only if $x \in L$

- φ is the conjunction of several clauses (see next slides).

Proof of Cook-Levin Theorem (ct'd)

- Initialize tape contents:
  - (c_{1,i,x_i}) for $1 \leq i \leq |x|$
  - $(c_{1,i,0} \vee c_{1,i,1})$ for $|x| < i \leq |x| + p(|x|)$
  - $(c_{1,i,\square})$ for $|x| + p(|x|) < i \leq T$
- Other initial conditions:
  - $(h_{1,1})$
  - $(s_{1,q_{\mathbf{start}}})$

Proof of Cook-Levin Theorem (ct'd)

- At most one symbol per cell (at each time):
  - $(\neg c_{t,i,\gamma} \vee \neg c_{t,i,\gamma'})$ for $1 \leq i, t \leq T$ and all $\gamma, \gamma' \in \Gamma$ with $\gamma \neq \gamma'$
- At most one tape head position at each time:
  - $(\neg h_{t,i} \vee \neg h_{t,i'})$ for $1 \leq i, i', t \leq T$ with $i \neq i'$
- At most one state at each time:
  - $(\neg s_{t,q} \vee \neg s_{t,q'})$ for $1 \leq t \leq T$ and $q, q' \in Q$ with $q \neq q'$

Proof of Cook-Levin Theorem (ct'd)

- Correct transitions.

  For $1 \leq i, t \leq T - 1$, $\gamma \in \Gamma$, and $q \in Q$:

  - $(c_{t,i,\gamma} \wedge h_{t,i} \wedge s_{t,q}) \rightarrow (c_{t+1,i,\gamma'} \wedge h_{t+1,i} \wedge s_{t+1,q'})$ if $\delta(q, \gamma) = (q', \gamma', \mathsf{S})$
  - $(c_{t,i,\gamma} \wedge h_{t,i} \wedge s_{t,q}) \rightarrow (c_{t+1,i,\gamma'} \wedge h_{t+1,i+1} \wedge s_{t+1,q'})$ if $\delta(q, \gamma) = (q', \gamma', \mathsf{R})$
  - $(c_{t,i,\gamma} \wedge h_{t,i} \wedge s_{t,q}) \rightarrow (c_{t+1,i,\gamma'} \wedge h_{t+1,i-1} \wedge s_{t+1,q'})$ if $\delta(q, \gamma) = (q', \gamma', \mathsf{L})$

Proof of Cook-Levin Theorem (ct'd)

- No change when the tape head is away:
  - $(c_{t,i,\gamma} \wedge \neg h_{t,i}) \rightarrow c_{t+1,i,\gamma}$ for $1 \leq t \leq T-1$, $1 \leq i \leq T$ and $\gamma \in \Gamma$
- The machine must accept:
  - $s_{T,q_{\mathbf{acc}}}$

Proof of Cook-Levin Theorem (ct'd)

- The formula φ is satisfiable if and only if there exists some $u \in \{0,1\}^{p(|x|)}$ such that $\mathbb{M}(x,u) = 1$, and thus if and only if $x \in L$.

- The conjuncts of φ can be equivalently rewritten as clauses (of size ≤ 4)
  - $(a \wedge b \wedge c) \rightarrow (d \wedge e \wedge f) \quad \mapsto$
    $(\neg a \vee \neg b \vee \neg c \vee d) \wedge (\neg a \vee \neg b \vee \neg c \vee e) \wedge (\neg a \vee \neg b \vee \neg c \vee f)$

- Computing φ takes polynomial time.
  - Polynomial number of atomic propositions and clauses

3SAT

Definition

The decision problem 3SAT is defined as follows:

$$\text{3SAT} = \{\ \varphi \mid \varphi \textit{ is a propositional logic formula in 3CNF and there exists a satisfying truth assignment } \alpha \textit{ for } \varphi\ \}$$

Or, described in a different format:

Input:	*A propositional logic formula φ in 3CNF.*
Question:	*Is φ satisfiable?*

- 3CNF: each clause (disjunction) contains at most 3 literals

3SAT is NP-complete

Theorem (Cook 1971, Levin 1969)

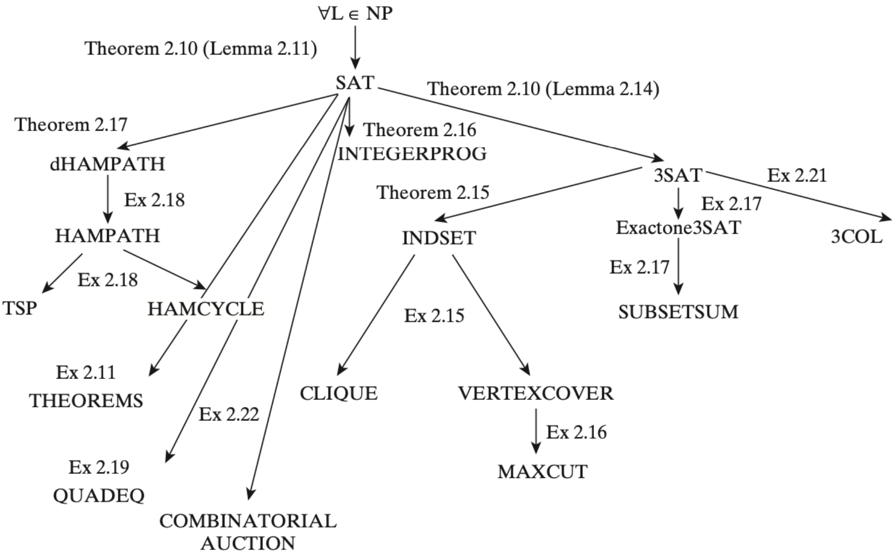
3SAT *is* NP*-complete.*

- The formula that we constructed is in 4CNF. So 4SAT is NP-complete. We give a polynomial-time reduction from 4SAT to 3SAT.
- We replace each clause $c = (\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4)$ of length 4 by:

  $$(\ell_1 \vee \ell_2 \vee z_c) \wedge (\neg z_c \vee \ell_3 \vee \ell_4),$$

  where z_c is a fresh variable.
- The resulting formula φ' is satisfiable if and only if the original formula φ is satisfiable.

The web of reductions



3COL is NP-complete

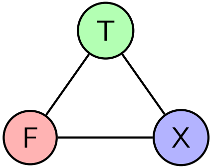
Theorem (Karp 1972)

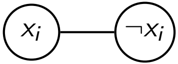
3COL *is* NP*-complete.*

- We will show NP-hardness by reduction from 3SAT.

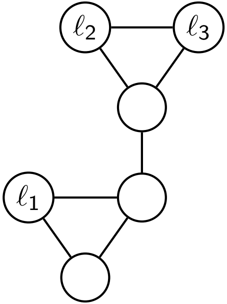
Gadgets

Gadgets





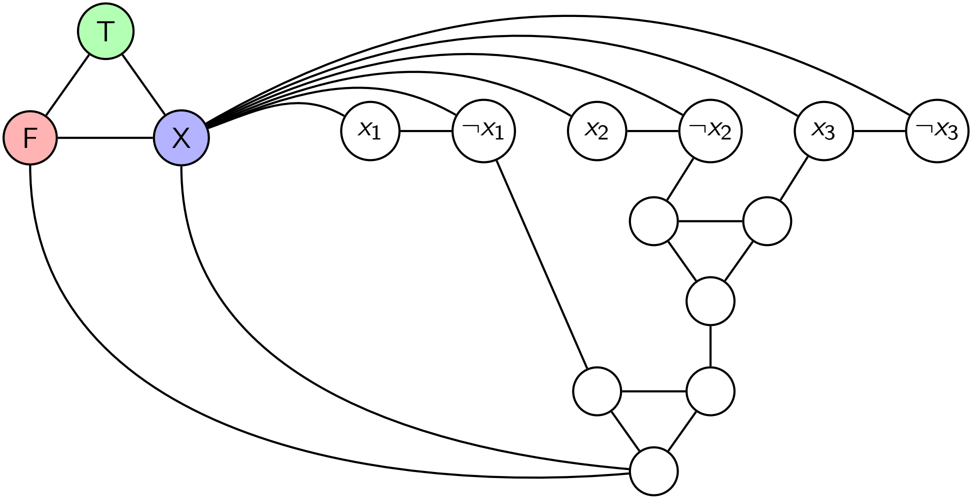
for each variable x_i



for each clause c_j

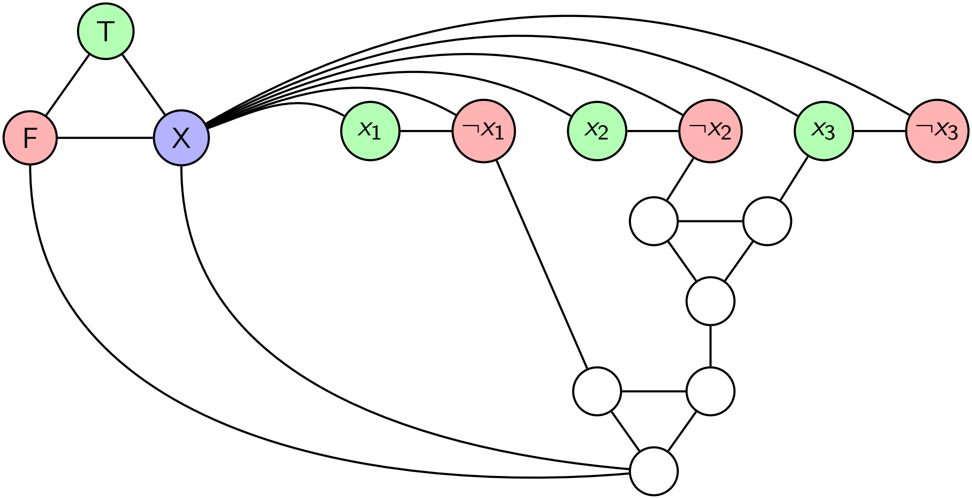
Example

$\varphi = (\neg x_1 \vee \neg x_2 \vee x_3)$



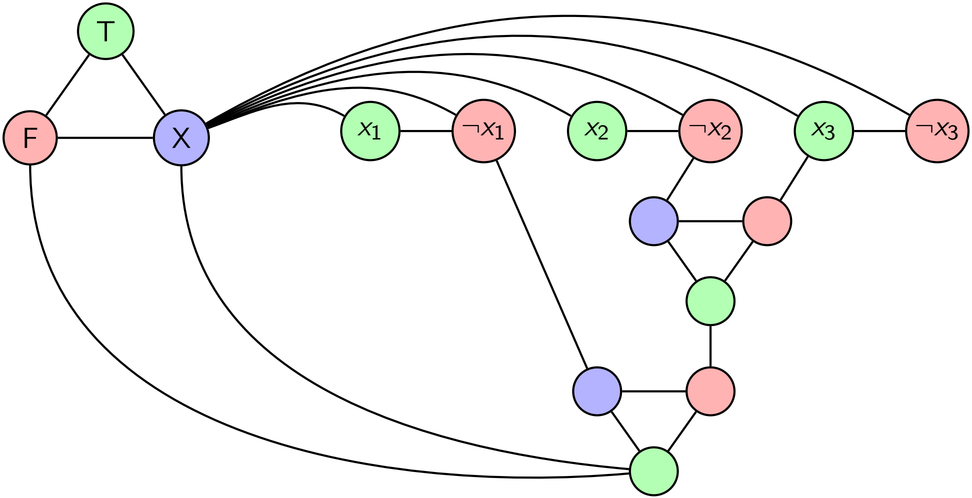
Example

$\varphi = (\neg x_1 \vee \neg x_2 \vee x_3)$, $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1\}$



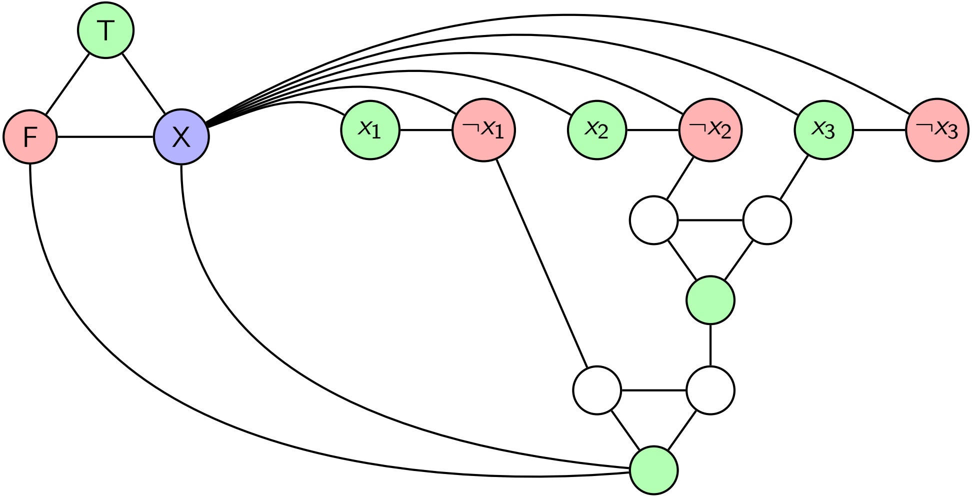
Example

$\varphi = (\neg x_1 \vee \neg x_2 \vee x_3)$, $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1\}$



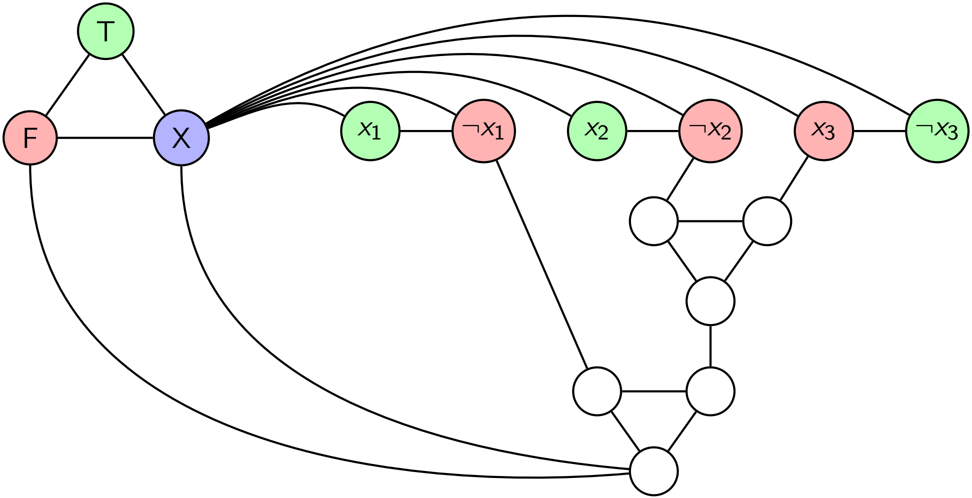
Example

$\varphi = (\neg x_1 \vee \neg x_2 \vee x_3)$, $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1\}$



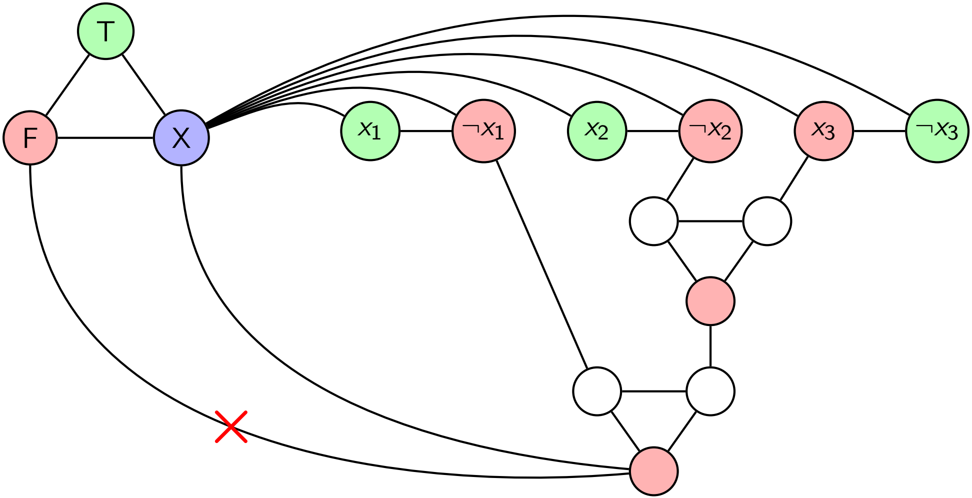
Example

$\varphi = (\neg x_1 \vee \neg x_2 \vee x_3)$, $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$



Example

$\varphi = (\neg x_1 \vee \neg x_2 \vee x_3)$, $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$



Search vs. decision

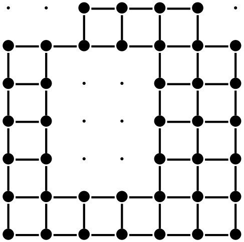
- Does NP-completeness tell us something useful about the search problems on which our decision problems are based?

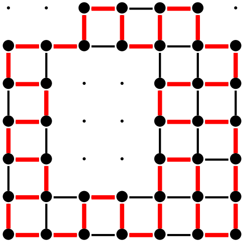
Proposition

Suppose that $\mathsf{P} = \mathsf{NP}$. Then for every $L \in \mathsf{NP}$ and each verifier $\mathbb{M}$ for L, there exists a polynomial-time Turing machine $\mathbb{B}$ that on input $x \in L$ outputs a certificate u for x.

Hamiltonian cycles in grid graphs

For the homework..



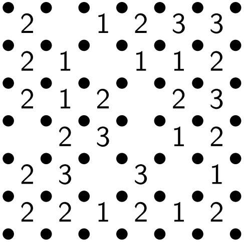


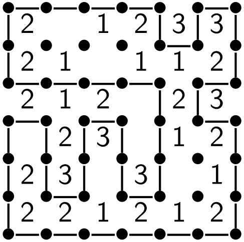
- A grid graph G..

..and a Hamiltonian cycle in G.

Slitherlink

For the homework..





- A Slitherlink instance I..

..and a solution for I.

Recap

- Prove that NP-complete problems exist ☺
- The Cook-Levin Theorem
- Concrete reductions between problems
- Search vs. decision problems

Next time

- Diagonalization arguments
- Time Hierarchy Theorems
- $\mathsf{P} \neq \mathsf{EXP}$