Time complexity (a bird's-eye-view tour)

- Section 7.1: time complexity, asymptotic upper bounds.
- Section 7.2: polynomial time, P
- Section 7.3: NP, polynomial verifiers, nondeterministic machines.
- Sections 7.4, 7.5: NP-Completeness

Decidability vs. Complexity
Time complexity classes

\[ \text{TIME}(t(n)) = \{ L \mid L \text{ is decidable by a deterministic Turing machine running in } O(t(n)) \} \]

- **Exponential**
  \[ \text{EXPTIME} = \bigcup_{k} \text{TIME}(2^{n^k}) \]

- **Polynomial**
  \[ P = \bigcup_{k} \text{TIME}(n^k) \]

- **Logarithmic**
  May not need to read all of input

Brute-force search

Invariant under many models of TMs
Which Turing machine model?

deterministic computation

non-deterministic computation

q₀

q_{rej}

q_{acc}

q_{rej}

q₀

q_{rej}

q_{acc}

q_{acc}
Time complexity

For M a deterministic decider, its **running time** or **time complexity** is the function $f: \mathbb{N} \to \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps M takes before halting, over all inputs of length n.}$$

For M a **nondeterministic decider**, its **running time** or **time complexity** is the function $f: \mathbb{N} \to \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps M takes before all branches of computation halting, over all inputs of length n.}$$
Time complexity classes

\[ \text{TIME} ( t(n) ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \} \]
\[ \text{deterministic, single-tape TM } \]

\[ \text{NTIME} ( t(n) ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \} \]
\[ \text{nondeterministic, single-tape TM } \]

Is \( \text{TIME}(n^2) \) a subset of \( \text{TIME}(n^3) \)?

A. Yes
B. No
C. Not enough information to decide
D. I don't know
Time complexity classes

\[
\text{TIME} \ (t(n)) = \{ L \mid L \text{ is decidable by } O(t(n)) \text{ deterministic, single-tape TM} \}
\]

\[
\text{NTIME} \ (t(n)) = \{ L \mid L \text{ is decidable by } O(t(n)) \text{ nondeterministic, single-tape TM} \}
\]

Is \( \text{TIME}(n^2) \) a subset of \( \text{NTIME}(n^2) \)?

A. Yes
B. No
C. Not enough information to decide
D. I don't know
Time complexity classes

\[ \text{TIME} \ ( t(n) ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \} \]
\[ \quad \text{deterministic, single-tape TM} \]

\[ \text{NTIME} \ ( t(n) ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \} \]
\[ \quad \text{nondeterministic, single-tape TM} \]

Is \( \text{NTIME}(n^2) \) a subset of \( \text{TIME}(n^2) \)?

A. Yes
B. No
C. Not enough information to decide
D. I don't know
"Feasible" i.e. P

- Can't use nondeterminism
- Can use multiple tapes

Often need to be "more clever" than naïve / brute force approach

Examples

PATH = \{<G, s, t> \mid G \text{ is digraph with } n \text{ nodes there is path from } s \text{ to } t\}
RELPRIME = \{<x, y> \mid x \text{ and } y \text{ are relatively prime integers}\}

Use Euclidean Algorithm to show in P

L(G) = \{w \mid w \text{ is generated by } G\} \text{ where } G \text{ is any CFG}

Use Dynamic Programming to show in P
"Verifiable" i.e. NP

- Can be decided by a nondeterministic machine in polynomial time
- Solution can be verified by a deterministic machine in polynomial time
- Best-known procedures to decide on a deterministic machine are exponential time, brute-force solutions
- Is it possible to decide by a deterministic machine in polynomial time?

\[ NP = \bigcup_{k} NTIME(n^k) \]
P = NP or P ≠ NP
# P vs. NP

**Problems in P**
- (Membership in any) CFL
- PATH
- \(E_{DFA}\)
- \(EQ_{DFA}\)
- Addition, multiplication of integers
- ...

**Problems in NP**
- Any problem in P
- HAMPATH
- VERTEX-COVER
- CLIQUE
- SAT
- ...

...
Examples in NP for graphs  

Sipser p. 264, 268, 284

HAMPATH = \{ <G,s,t> \mid G \text{ is digraph with a path from } s \text{ to } t \text{ that goes through every node exactly once} \}
Examples in NP for graphs  \[ \text{Sipser p. 264,268,284} \]

\[
\text{VERTEX-COVER} = \{ <G,k> \mid G \text{ is an undirected graph with a } k\text{-node vertex cover} \}
\]
Examples in NP for graphs  

Sipser p. 264, 268, 284

CLIQUE = \{ <G,k> | G is an undirected graph with a k-clique \}

How many possible k-cliques are there in a graph with n vertices?

A. \( n^k \)
B. \( k^n \)
C. \( C(n,k) \)
D. \( P(n,k) \)
E. I don’t know
Examples in NP for logic  

Sipser p. 271

SAT = \{ <\varphi> | \varphi \text{ is a satisfiable Boolean formula} \}
1970s Stephen Cook and Leonid Levin independently and in parallel lay foundations of **NP-completeness**

**Definition:** A language $B$ is **NP-complete** if

- it is in NP and
- every $A$ in NP is polynomial time reducible to it.

**Implication:** If an NP-complete problem has a polynomial time algorithm, then all NP problems are polynomial time solvable.

**Cook-Levin Theorem:** SAT is NP-complete.
Reductions to the rescue

1970s Stephen Cook and Leonid Levin independently and in parallel laid foundations of NP-completeness.

Definition: A language B is NP-complete if
- it is in NP and
- every A in NP is polynomial time reducible to it.

Implication: If an NP-complete problem has a polynomial time solution, then all NP problems are polynomial time solvable.

Cook-Levin Theorem: SAT is NP-complete.

What would it mean to find a polynomial time solution to an NP-complete problem?

A. P is a subset of NP
B. P is a proper subset of NP
C. P equals NP
D. P does not equal NP
E. I don't know
Reduction: 3SAT to CLIQUE

Cook-Levin Theorem: SAT is NP-complete.

To prove a different problem is NP-complete, show a reduction from a problem known to be NP-complete.

Next: 3SAT reduces to CLIQUE. Therefore, CLIQUE is NP-complete.

HW: CLIQUE reduces to SEPARATE. Therefore, SEPARATE is NP-complete.
Reduction: 3SAT to CLIQUE

3cnf-formula:
- Several *clauses* connected with AND
- Each clause has three *literals* connected with OR
- A literal is a Boolean variable or its negation

\[
(x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)
\]

3SAT = \{ <\varphi> \mid \varphi \text{ is a satisfiable 3cnf-formula} \}
Reduction: 3SAT to CLIQUE

3cnf-formula:
• Several clauses connected with AND
• Each clause has three literals connected with OR
• A literal is a Boolean variable or its negation

3SAT = \{ <\varphi> \mid \varphi \text{ is a satisfiable 3cnf-formula} \}

Which assignment of variables makes this 3cnf-formula true?
A.  x_1=\text{T}, x_2=\text{T}, x_3=\text{F}
B.  x_1=\text{T}, x_2=\text{F}, x_3=\text{F}
C.  x_1=\text{T}, x_2=\text{T}, x_3=\text{F}
D.  None of the above
E.  More than one of the above

\( (x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3) \)
Reduction: 3SAT to CLIQUE

Given a 3cnf-formula $\phi$ with $k$ clauses, define a graph $G$:

- Vertices are the literals in each clause
- Edges between all vertices except:
  - Two literals in the same clause
  - Two literals that are the negation of one another

\[
(x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)
\]
Reduction: 3SAT to CLIQUE

Sipser Thm 7.32

\[(x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor x_2 \lor x_3)\]

\[\varphi \text{ is satisfiable } \iff G \text{ has a k-clique}\]
Reduction: 3SAT to CLIQUE

Which assignment of variables makes this 3cnf-formula true?
A. $x_1=T, x_2=T, x_3=F$
B. $x_1=T, x_2=F, x_3=F$
C. $x_1=T, x_2=T, x_3=F$
D. None of the above
E. More than one of the above

What are the corresponding $k$-cliques?
Conclude: CLIQUE is NP-complete

Homework:
• Similar proof to show CLIQUE reduces to SEPARATE.
• How does solving one of these problems enable you to solve the other?
• Conclude: SEPARATE is NP-complete.