Topic 23: Complexity Theory, P, and NP

(CIS 34)

CPS 230, Fall 2001

- Area of theoretical computer science.
- Broad Goals:
  - Establish (upper and) lower bounds on the number of steps it takes to solve a computational problem, using any algorithm.
  - Identify potentially hard computational problems.
  - Identify common traits among such problems.
  - Accumulate evidence for hardness.

The Clique problem

Def. Let $G = (V, E)$ be an undirected graph. A **clique** of $G$ is a set $V' \subseteq V$ such that $orall u, v \in V'$ we have $(u, v) \in E$.

i.e. **clique** is a complete subgraph of $G$.

Optimization problem for clique:

Given $G$, find size of largest clique.

Algorithm for clique problem

```
for j ← |V| downto 1
  do list all subsets of $j$ vertices
     check if clique

Time = $\Omega(2^{|V|})$ — VERY SLOW!
```

Is there a “tractable” algorithm?

What do we mean by “tractable”?

Polynomial-time algorithms

Def. An algorithm $A$ runs in **polynomial time** if $\exists$ positive constants $c$ and $k$ such that $A$’s running time is at most $cn^k$ on any problem instance of size $n$.

Q. What do we mean by “size $n$”?

A. Any problem instance can be **encoded** as a sequence of 0’s and 1’s.

Assume all numbers in encoding are in binary.

**Size** of problem instance = length of encoding. Multiple arguments are coded into a single string.

Choice of the encoding doesn’t matter as long as can compute one from another in **polynomial time**.
Example: Factoring

Given number \( n \), determine \( m \) that divides \( n \)

Suppose number \( n \) encoded in unary: \(11...1\)

Any factor \( m \) can be no greater than \( \sqrt{n} \)

Thus, check all numbers from \( 2 \ldots \sqrt{n} \) (\( O(\sqrt{n}) \) checks)

Division in unary takes linear time \( O(n) \)

Thus factoring takes time \( O(n^{3/2}) \!\)

However, given binary encoding of \( L \) bits,
we must do \( \sqrt{n} = 2^{L/2} \) divisions!

Bottom line: specific encoding usually irrelevant
(we can solve problem on \( \langle G \rangle_1 \) in polynomial time
iff we can solve problem on \( \langle G \rangle_2 \) in polynomial time)

Normally presume “standard” encoding; that is,
using strings over a finite alphabet.
Typically need to be careful only when representation
sizes differ by an exponential factor.

Polynomial-time algorithms

Most problems seen so far have polynomial-time algo-

\[ \text{Q. Does clique problem have a polynomial-time algo-
ritm?} \]

\[ \text{A. No one knows. Most would be surprised if it did.} \]

Decision problem

Decision: yes/no (1/0) output

Decision problem for clique:

Given \( \langle G, k \rangle \),
\( G \) is an undirected graph, \( k \) is integer
does \( G \) have a clique of size \( k \)?

Complexity theory deals mostly with
decision problems.

Theorem (for Clique):

Optimization problem solvable in polynomial time

iff decision problem solvable in polynomial time.

Proof (for clique)

\( (\text{Opt} \Rightarrow \text{Dec}) \)
Find max clique size.
Compare with \( k \).

\( (\text{Dec} \Rightarrow \text{Opt}) \) Binary search.
Natural question: true for other problems?
Yes... when the thing to be optimized
has a polynomial number of possible values
(with respect to the size of the input instance).
Formal languages

**Def.** A formal language is a set of binary strings.

**Example 1:**
\[ \{0, 1\}^* = \{ \text{all binary strings} \} \]

**Example 2:**
Every decision problem can be viewed as a language

\[ \text{CLIQUE} = \{(G, k) : G \text{ has a clique of size } k\} \]

(i.e. CLIQUE is the set of binary encodings of all the graphs \( G \) that have a clique of size \( k \).)

Let \( Q \) be a decision problem

\( Q \) is entirely characterized by problem instances that produce a 1 (yes) answer

Can view \( Q \) as a language

\[ L = \{ x \in \Sigma^* : Q(x) = 1 \} \]

where \( \Sigma \) is encoding alphabet.

For binary encoding \( \Sigma = \{0, 1\} \).

---

**Example:**

decision problem & corresponding language PATH

Decision problem:

Given graph \( G = (E, V) \), two nodes \( u \) and \( v \), and an integer \( k \), is there a path from \( u \) to \( v \) in graph \( G \) of length \( \leq k \)?

Corresponding language:

\[ \text{PATH} = \{ (G, u, v, k) : G = (V, E) \text{ is an undirected graph, } u, v \in V, \]

\( k \geq 0 \) is an integer, and there exists a path from \( u \) to \( v \) in \( G \) whose length is at most \( k \} . \]

---

**Algorithms and Decision Problems**

An algorithm \( A \) accepts \( x \in \{0, 1\}^* \) if \( A(x) = 1 \)

An algorithm \( A \) rejects \( x \in \{0, 1\}^* \) if \( A(x) = 0 \)

Might do neither (i.e., loop forever)

Language \( L \) is accepted by \( A \) if

\[ L = \{ x \in \{0, 1\}^* : A(x) = 1 \} \]
**Algorithms and Decision Problems**

Language $L$ is **decided** by $A$ if

\[ x \in L \Rightarrow A(x) = 1 \text{ and } \quad x \notin L \Rightarrow A(x) = 0 \]

Accepted $\Rightarrow$ might run for ever, output something, etc.

Decided $\Rightarrow$ always halts and outputs 0 or 1

---

**Class P**

**Def.** An algorithm $A$ runs in **polynomial time**

if $\exists$ positive constants $c$ and $k$

such that $A$’s running time is at most $cn^k$

on any problem instance of size $n$.

\[ P = \{ L \subseteq \{0,1\}^* : \exists A \text{ such that } L \text{ is accepted by } A \text{ in polynomial time } \} \]

$P$: class of languages accepted by a polynomial-time algorithm.

---

**Theorem**

$P = \{ L \subseteq \{0,1\}^* : \exists A \text{ such that } L \text{ is decided by } A \text{ in polynomial time } \}$

**Proof:**

(decided) $\Rightarrow$ (accepted) Trivial

(accepted) $\Rightarrow$ (decided) ”simulation argument:"

Construct $A'$ deciding $L$ as follows:

Let $A$ be some algorithm that accepts $L$ in polynomial time.

Thus, $\exists$ cons $c, k \geq 0$ such that

$\forall x \in L$, $A$ accepts $L$ in $\leq c|x|^k$ steps.

Define $A'$ as follows: It simulates $A$

If $A$ outputs 1 within $c|x|^k$ steps, $A'$ outputs 1.

Else $A'$ outputs 0.

$A'$ runs in polynomial time

(i.e., always halts, even if $A$ loops).

Is this proof constructive?

---

**Q.** Is CLIQUE $\in P$?

**A.** No one knows.

Most would be surprised if it is, since CLIQUE is NP-complete (next time).
**Class NP**

Is it easier to find a clique of size \( k \) or to check whether a given subset of \( k \) vertices forms a clique?

**Def.** A 2-argument algorithm \( A \)

verifies an input \( x \in \{0,1\}^* \)

if \( \exists y \in \{0,1\}^* \) such that \( A(x,y) = 1 \).

The language verified by \( A \) is

\[
L = \{ x \in \{0,1\}^* : \exists y \in \{0,1\}^* \text{ such that } A(x,y) = 1 \}
\]

We call \( y \) a certificate for \( x \in L \).

Polynomial-time verification:

\[
|y| = O(|x|^{O(1)}) \text{ and } A \text{ is polynomial time.}
\]

---

**Example:**

\( \text{CLIQUE} \in \text{NP} \).

Input 1: \( \langle G, k \rangle \)

Input 2: \( \langle V' \rangle \) such that \( V' \subseteq V \)

Verification algorithm checks that \( |V'| = k \),

and that every pair of vertices in \( V' \) is connected by an edge in \( E \).

If so, it outputs 1; else it outputs 0.

\( \langle G, k \rangle \) can be verified *iff* \( G \) has a clique of size \( k \).

Verification algorithm runs in polynomial time.

Thus, \( \text{CLIQUE} \in \text{NP} \).

"Verification Set": Pairing of input, something else

- Recognizable in polynomial time
- \( G \) can be verified \( \iff \) \( G \) has property

---

**Theorem:** \( P \subseteq \text{NP} \)

**Proof:**

Let \( L \) be in \( P \Rightarrow \exists \) a polynomial-time algorithm \( A \) that decides \( L \).

This polynomial-time algorithm \( A \) can be converted to a 2-argument verification algorithm that ignores its verification argument.

**Big Question:** Is \( P = \text{NP} \)?

Open question for \( \approx 25 \) years.

We do know that \( \text{CLIQUE} \) is in a sense the hardest problem in \( \text{NP} \).

If we can solve \( \text{CLIQUE} \) in polynomial time then every problem in \( \text{NP} \) can be solved in polynomial time!

\( \text{CLIQUE} \) is \( \text{NP-complete} \) (next time).