Course contents

- Classical resources: Time, Space, Non-determinism.
- Alternation & the Polynomial Hierarchy.
- Non-uniform complexity & Lower bounds.
- Randomness and its power.
- Proofs, Interaction, Knowledge.
- Quantum computation.

Complexity Theory

Basic iteration:

- Identify resource; pick a gross bound on resource.
- Find points = problems.
- Draw arrows (reductions). $A \rightarrow B$, or $A \leq B$ if $A$ reduces to $B$.
- In some cases rule out arrows.

Hopefully, get a map of all computational problems and complexities involved.

Classically ...

Resources Time, Space, Non-determinism.

Stopping points Logarithmic, Polynomial, Exponential.

Reductions?

- Karp vs. Turing.
- Logspace vs. Polynomial time.

Turing reductions & Relativization

Definition: $L_1 \leq^p_T L_2$ if there exists a polynomial time Turing machine $M$ that with access to an oracle for $L_2$ can solve the problem $L_1$.

Languages vs. Problems Problems are general functions; Languages are Boolean functions. Turing reductions work generally. Their most powerful usage is to reduce general problems to languages.

Exercise Reduce search-SAT to SAT.

Relativization $M$ above is an oracle Turing machine since it invokes an oracle $O$ occasionally. Notation to describe this
duo: $M^O$. Here our focus was on what can $O$ be used to do, when we vary $M$. In relativization, we often fix $M$ (or the class it comes from) and vary $O$ to see what can be done. Will see more next lecture.

**Food for thought** Why need Karp reductions?
(Hint: two famed classes would be indistinguishable under Turing reductions.)

---

**Classical classes**

- Logarithmic space $L$.
- Polynomial time $P$.
- Polynomial space $PSPACE$.
- Exponential time $E/EXP$.
- etc.

---

**Basic results**

- Time Hierarchy theorem.
- Space Hierarchy theorem.
- Blum’s speedup theorem.
- Any one remembers exact form?
- Diagonalization - Tool #1 in proving absence of arrows.
Food for thought

- Given language in NP, can we decide if it is in P or not?
- Is every language in NP either in P or NP-complete?
- Is there a NTIME hierarchy theorem? What goes wrong with the usual proof?
- Is linear time a reasonable notion? How about nearly linear time?

Some other basic results

- Time(\(t\)) in NTIME(\(t\)) in Space(\(t\)) (actually can do a bit better)!
- Space(\(s\)) in Time(\(2^n\)).
- Technically harder results:
  - NSPACE(\(s\)) in SPACE(\(s^2\)).
  - NSPACE(\(s\)) in coNSPACE(O(\(s\))).
- Will prove above later today.

Big questions

\[ P = \text{NP?} \]

1. Belief: \( P \neq \text{NP} \).
2. Stronger belief: \( \text{NP} \neq \text{co-NP} \).
3. Weaker beliefs:
   (a) \( P \neq \text{PSPACE} \).
   (b) SAT not in L.
   (c) SAT not in nearly Linear Time.
4. Another belief: \( L \neq P \).

We know at least one of 3(a) or 4 is true!

Will show one more such statement (hopefully).

Rest of lecture

Quick Review of

- Savitch’s theorem.
- Immerman-Szelepcsényi theorem.
Savitch’s theorem

Thm: For all space constructible \( s(n) \geq \log n \), \( \text{NSPACE}(s(n)) \subseteq \text{SPACE}s^2(n) \).

Simplifying assumptions:

- Suffices to consider the case \( s = \log n \).
- Suffices to show that STCON can be solved in space \( O(\log^2 n) \).
- STCON:
  
  Given: Directed graph \( G \), vertices \( s, t \).
  
  YES instances: There is a directed path from \( s \) to \( t \) in \( G \).
- Suffices to let \( n \) be power of 2: \( n = 2^k \).

Basic Lemma on Space

Basic Lemma: If \( f, g : \{0,1\}^n \rightarrow \{0,1\}^n \) can be computed in space \( s_1 \) and \( s_2 \) respectively, then \( f \circ g : \{0,1\}^n \rightarrow \{0,1\}^n \) can be computed in space \( s_1 + s_2 \) (no big-Oh’s).

Proof: Omitted.

Lemma: Given \( A \), the matrix \( A^{2^\ell} \) can be computed in space \( \ell \log n \).

Proof: Induction using Basic Lemma.

Savitch’s theorem follows.

STCON Algorithm

- Let \( A \) be the adjacency matrix of \( G \).
- Suffices to compute \( A^n \), where \( A \cdot B \) denote Boolean matrix multiplication and \( A^n = A \cdot A^{n-1} \).

Immerman–Szelepscenyi Theorem

Thm: For all space constructible \( s(n) \geq \log n \), \( \text{co-NSPACE}(s(n)) \subseteq \text{NSPACE}O(s(n)) \).

Idea:

- Suffices to prove \( \text{co-STCON} \) in \( NL \).
- Key quantities:

\[
\Gamma_\ell(s) = \{ v \in V | \exists \text{ path } w. \text{ length } \leq \ell \text{ from } s \text{ to } v \} \]

\[
\text{COUNT}(s, \ell) = |\Gamma_\ell(s)|
\]

- Central subroutine: \( \text{CHECK}(u, \ell, \text{COUNT}) \).

Guarantee: If \( \text{COUNT} = \text{COUNT}(s, \ell - 1) \), then \( \text{OUTPUT} = \text{TRUE} \) iff there is no path from \( s \) to \( u \) of length \( \leq \ell \).
NL=coNL: Proof

Lemma 1: co-STCON in NL if CHECK in NL.

Proof:

- Inductively, compute \( \text{COUNT}(s, \ell) \) given \( \text{COUNT}(s, \ell - 1) \) as follows:
  - Initialize \( \text{COUNT}(s, \ell = 0) \).
  - For each \( u \in V \) guess if \( v \in \Gamma_\ell(s) \).
  - If Guess=YES, verify the guess and increment \( \text{COUNT}(s, \ell - 1) \).
  - If Guess=NO, use \( \text{CHECK}(u, \ell, \text{COUNT}(s, \ell - 1)) \) to verify guess.

NL=coNL: Proof (contd.)

Lemma 2: \( \text{CHECK}(u, \ell, \text{COUNT}) \in \text{NL} \).

Algorithm:

- Initialize \( \text{COUNT-OFAR} = 0 \);
- For every \( v \in V \) do:
  - Guess if \( v \in \Gamma_{\ell-1}(s) \).
  - If Guess=NO, do nothing;
  - If Guess=YES, (1) verify guess, (2) increment \( \text{COUNT-OFAR} \), and (3) verify \((v, u)\) is not an edge.
- Verify \( \text{COUNT-OFAR} = \text{COUNT} \).
- Return(TRUE).

“Verify COND” \( \equiv \) Abort if COND is FALSE.

Next lecture

- Relativization.
- Baker Gill Solovay theorem.
- ?