P and NP

Sipser 7.2-7.3 (pages 256–270)

Polynomial time

\[ P = \bigcup_i \text{TIME}(n^i) \]

\[ \text{TIME}(n) \subset \text{TIME}(n^2) \subset \text{TIME}(n^3) \subset \ldots \]
“Practical” problems

• If \( n = 100 \)
  – \( n^3 = 1 \text{ billion} \)
  – \( 2^n > \# \text{atoms in the universe} \)

• Polynomial time is generally considered “practical” for a computer

• \( P \) is the class of “solvable” or “tractable” problems

How many tapes?

• Definition 7.12:
  \( P \) is the class of languages that are decidable in polynomial time by a deterministic single-tape Turing machine.

• But remember… we can convert from single-tape to multi-tape!
  – What was the time complexity conversion?
Polynomially equivalent models

\[
\begin{align*}
M & \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad \| \\
\quad & \quad a \quad a \quad a \quad \| \\
\quad & \quad b \quad a \quad \| \\
S & \quad # \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad # \quad a \quad a \quad # \quad b \quad a \quad # \quad \|
\end{align*}
\]

Finding your way

- \( \text{PATH} = \{<G,s,t>| \exists \text{ directed } s \text{ to } t \text{ path in } G\} \)
PATH ∈ P

• M = "On input <G,s,t>:
  1. Place a mark on node s.
  2. Repeat until no additional nodes are marked:
      3. Scan all edges of G.
         If (a,b) found from marked node to unmarked node, mark b.
  4. If t is marked, accept. Otherwise, reject.

Hamiltonian paths

• HAMPATH = {<G,s,t> | \exists Hamiltonian path from s to t}
Hamiltonian paths

- \( HAMPATH = \{ <G,s,t> \mid \exists \text{ Hamiltonian path from } s \text{ to } t \} \)

Checking for Hamiltonian paths

- Brute force method
- \( E = \text{"On input } <G,s,t>:\)
  1. Generate all orderings, \( p_1, p_2, ..., p_n \), of the nodes of \( G \)
  2. Check whether \( s = p_1 \) and \( t = p_n \)
  3. For each \( i=1 \) to \( n-1 \), check whether \( (p_i, p_{i+1}) \)
     is an edge of \( G \).
     If any are not, reject. Otherwise, accept."
Guessing a solution

- \textbf{N = "On input} \langle G, s, t \rangle:$
  1. Guess an orderings, \( p_1, p_2, \ldots, p_n \), of the nodes of \( G \)
  2. Check whether \( s = p_1 \) and \( t = p_n \)
  3. For each \( i = 1 \) to \( n-1 \), check whether \( (p_i, p_{i+1}) \)
     is an edge of \( G \).
     \textit{If any are not, reject. Otherwise, accept."}

Nondeterministic time complexity

- Definition 7.9: Let \( N \) be a NTM. The \textbf{running time} of \( N \) is a function \( f: N \rightarrow N \), where \( f(n) \) is the maximum number of steps that \( N \) uses on any branch of its computation on any input of length \( n \)

\[ f(n) \]
Nondeterministic time complexity classes

- Definition 7.21:
  \[ \text{NTIME}(t(n)) = \{ L \mid L \text{ is decided in } O(t(n)) \text{ time by an NTM} \} \]

Verifiers

- Definition 7.18:
  - A **verifier** for a language \( A \) is an algorithm \( V \), where \( A = \{ w \mid V \text{ accepts } <w,c> \text{ for some string } c \} \)
  - A polynomial time **verifier** runs in polynomial time in the length of \( w \)
  - A language \( A \) is **polynomially verifiable** if it has a polynomial time verifier
The class NP

- Definition 7.19:
  NP is the class of languages that have polynomial time verifiers

Nondeterminism and verifiers

- Theorem 7.20:
  A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine

- Corollary 7.19: $NP = \bigcup_k\text{NTIME}(n^k)$
The classes P and NP

\[ P = \bigcup_i \text{TIME}(n^i) \]

\[ NP = \bigcup_i \text{NTIME}(n^i) \]

The big question: \( P = NP ? \)

\[ P \subseteq NP \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME} \]

proper containment