The Classes P and NP

Lecture 37
Sections 7.1 - 7.3

Robb T. Koether
Hampden-Sydney College

Mon, Nov 24, 2014
1 Assignment

2 Time Complexity Classes

3 Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time

4 Other Machine Models

5 The Class P
   - The PATH Problem
   - The CFL Membership Problem

6 Nondeterministic Algorithms
   - Generators
   - Verifiers

7 The Class NP
Assignment

Homework

- Read Section 7.1, pages 254 - 256.
- Read Section 7.2, page 256 - 263.
- Exercises 1, 2, 3, 4, 9, pages 294 - 295.
- Problem 12, page 295.
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
Definition (TIME complexity classes)

Let $t : \mathbb{N} \to \mathbb{R}^+$ be a function. The time complexity class $\text{TIME}(t(n))$ is the set of all languages that are decidable by a $O(t(n))$ Turing machine.
Examples

Examples (TIME classes)

- SHIFT is in \( \text{TIME}(n) \).
- COPY is in \( \text{TIME}(n^2) \).
- INCR and DECR are in \( \text{TIME}(n) \).
- Is COPY in \( \text{TIME}(n) \)?
- Are INCR and DECR in \( \text{TIME}(n^2) \)?
Examples

Questions

- If ADD uses INCR and DECR, then what is the time complexity class of ADD?
- If MULT uses ADD and DECR, then what is the time complexity class of MULT?
Example (TIME class of PRIME)

Let the input be an $n$-bit integer $k$.

Suppose a Turing machine uses the following strategy to factor the integer.

- Divide $k$ by each integer from 2 to $k - 1$.
- Assume division runs in $O(t(n))$.
- What will be the run time of this machine?
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
Programs that run in polynomial time are considered *feasible*. 
Programs that run in polynomial time are considered *feasible*. For example, sorting long lists is feasible.
Polynomial-Time Algorithms

- Programs that run in polynomial time are considered *feasible*.
- For example, sorting long lists is feasible.
- Programs that run in exponential time are considered *infeasible.*
Programs that run in polynomial time are considered \textit{feasible}.
For example, sorting long lists is feasible.
Programs that run in exponential time are considered \textit{infeasible}.
For example, factoring large integers is infeasible.
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
Comparison of Run Times

Example (Polynomial Time)

- Suppose we have Turing machines $M_1$, $M_2$, and $M_3$ whose run times are $O(n)$, $O(n^2)$, and $O(n^3)$, respectively.
- Suppose that each one takes 1 $\mu$s when the input size is $n = 1000$. 
### Example (Polynomial Time)

<table>
<thead>
<tr>
<th>Input size</th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1 $\mu$s</td>
<td>1 $\mu$s</td>
<td>1 $\mu$s</td>
</tr>
<tr>
<td>$10^4$</td>
<td>10 $\mu$s</td>
<td>100 $\mu$s</td>
<td>1 ms</td>
</tr>
<tr>
<td>$10^5$</td>
<td>100 $\mu$s</td>
<td>10 ms</td>
<td>1 s</td>
</tr>
<tr>
<td>$10^6$</td>
<td>1 ms</td>
<td>1 s</td>
<td>1000 s</td>
</tr>
<tr>
<td>$10^7$</td>
<td>10 ms</td>
<td>100 s</td>
<td>12 d</td>
</tr>
<tr>
<td>$10^8$</td>
<td>100 ms</td>
<td>2.8 h</td>
<td>32 y</td>
</tr>
<tr>
<td>$10^9$</td>
<td>1 s</td>
<td>12 d</td>
<td>32000 y</td>
</tr>
</tbody>
</table>
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
Comparison of Run Times

Example (Exponential Time)

- Suppose we have Turing machines $M_4$ and $M_5$ whose run times are $O(2^n)$ and $O(4^n)$, respectively.
- Suppose that each one takes $1 \, \mu s$ when the input size is $n = 1000$. 
### Example (Exponential Time)

<table>
<thead>
<tr>
<th>Input size</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>$M_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1.03 µs</td>
<td>1 µs</td>
<td>1 µs</td>
</tr>
<tr>
<td>1010</td>
<td>1.06 µs</td>
<td>1.02 ms</td>
<td>1.05 s</td>
</tr>
<tr>
<td>1020</td>
<td>1.09 µs</td>
<td>1.05 s</td>
<td>13 d</td>
</tr>
<tr>
<td>1030</td>
<td>1.12 µs</td>
<td>1073 s</td>
<td>37,000 y</td>
</tr>
<tr>
<td>1040</td>
<td>1.15 µs</td>
<td>13 d</td>
<td>$38 \times 10^9$ y</td>
</tr>
<tr>
<td>1050</td>
<td>1.18 µs</td>
<td>36 y</td>
<td>$40 \times 10^{15}$ y</td>
</tr>
<tr>
<td>1060</td>
<td>1.21 µs</td>
<td>37,000 y</td>
<td>$42 \times 10^{21}$ y</td>
</tr>
</tbody>
</table>
Outline

1 Assignment

2 Time Complexity Classes

3 Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time

4 Other Machine Models

5 The Class P
   - The PATH Problem
   - The CFL Membership Problem

6 Nondeterministic Algorithms
   - Generators
   - Verifiers

7 The Class NP
Example (Factorial Time)

- Finally, suppose we have a Turing machine $M_6$ whose run time is $O(n!)$.
- Suppose that it takes $1 \mu s$ when the input size is $n = 1000$. 
Comparison of Run Times

Example (Factorial Time)

<table>
<thead>
<tr>
<th>Input size</th>
<th>$M_5$</th>
<th>$M_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1 µs</td>
<td>1 µs</td>
</tr>
<tr>
<td>1001</td>
<td>4 µs</td>
<td>1.001 ms</td>
</tr>
<tr>
<td>1002</td>
<td>16 µs</td>
<td>1.003 s</td>
</tr>
<tr>
<td>1003</td>
<td>64 µs</td>
<td>16.8 m</td>
</tr>
<tr>
<td>1004</td>
<td>256 µs</td>
<td>11.7 d</td>
</tr>
<tr>
<td>1005</td>
<td>1.024 ms</td>
<td>32.2 y</td>
</tr>
<tr>
<td>1006</td>
<td>4.096 ms</td>
<td>32,381 y</td>
</tr>
</tbody>
</table>
Theorem

If a problem $P \in \text{TIME}(t(n))$ for a $k$-tape Turing machine and $t(n) \geq n$, then $P \in \text{TIME}(t^2(n))$ for a single-tape Turing machine.
Theorem

If a problem $P \in \text{TIME}(t(n))$ for a nondeterministic Turing machine, then $P \in \text{TIME}(2^{O(t(n))})$ for a deterministic Turing machine.
The factoring problem can be solved \textit{nondeterministically} in $O(n)$ time, where $n$ equals the length of the number. (How?)

Therefore, the factoring problem can be solved \textit{deterministically} in $2^{O(n)}$ time.
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
The class \( P \) is the class of all languages (i.e., problems) that are decidable in polynomial time by a deterministic single-tape Turing machine.

\[
P = \bigcup_{k=0}^{\infty} \text{TIME}(n^k).
\]
We have seen that if a problem can be decided by a multitape Turing machine in polynomial time $O(t(n))$, then it can be decided by a single-tape Turing machine in time $O(t^2(n))$, which is still polynomial.

However, we have also seen that a problem that can be solved nondeterministically in polynomial time can be solved deterministically in exponential time, but not necessarily in polynomial time.

Factoring an integer is an example (as far as we know).
Outline

1. Assignment
2. Time Complexity Classes
   - Polynomial Time
   - Exponential Time
   - Factorial Time
3. Comparison of Run Times
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
The PATH Problem

Given a directed graph $G$ and two vertices $s$ and $t$, does there exist a directed path in $G$ from $s$ to $t$?
The PATH Problem

Example (The PATH Problem)

- Let $G$ be the graph.
- Let $m$ be the number of vertices in $G$.
- Let $e(i, j)$ be true if there is an edge from vertex $i$ to vertex $j$ and false otherwise.
- Let $s$ and $t$ be the starting and ending vertices.
The PATH Problem

Example (The PATH Problem)

- Begin at $s$ and let $M = \{s\}$.
- Add to $M$ all vertices that are adjacent to $s$ (one edge away).
- Next, add to $M$ all vertices that are adjacent to those vertices that are already in $M$.
- Repeat the previous step until no new vertices are added to $M$.
- See whether $t \in M$. 
The PATH Problem

Example (The PATH Problem)

Is \( t \in M \)? No.
Example (The PATH Problem)

Is \( t \in M \)? No.
Example (The PATH Problem)

Is $t \in M$? No.
Example (The PATH Problem)

Is $t \in M$? No.
The PATH Problem

Example (The PATH Problem)

Is $t \in M$? Yes.
Example (The PATH Problem)

Is $t \in M$? No.
The PATH Problem

Example (The PATH Problem)

Is $t \in M$? No.
The PATH Problem

Example (The PATH Problem)

Is $t \in M$? No.
Example (The PATH Problem)

Is $t \in M$? No.
Example (The PATH Problem)

Is \( t \in M \)? No.
Example (The PATH Problem)

PATH(graph G, vertex s, vertex t)
{
    M = {s};
    M_old = {};
    while (M != M_old)
    {
        M_old = M;
        for (i = 0; i < m; i++)
            for (j = 0; j < m; j++)
                if (e(i, j) \in E && i \in M && j \notin M)
                    M = M \cup \{j\};
    }
    if (t \in M)
        accept();
    else
        reject();
}
An Algorithm for PATH

Example (The PATH Problem)

- We can analyze each step in the function $\text{PATH()}$.
- Each step is done in polynomial time.
- Thus, $\text{PATH} \in \mathbb{P}$. 
Outline

1 Assignment
2 Time Complexity Classes
3 Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4 Other Machine Models
5 The Class P
   - The PATH Problem
   - The CFL Membership Problem
6 Nondeterministic Algorithms
   - Generators
   - Verifiers
7 The Class NP
The CFL Membership Problem

Given a context-free language \( L \) and a word \( w \), is \( w \) in \( L \)?

- We may assume that we have a grammar for \( L \) in Chomsky Normal Form.
- Let \( m \) be the length of the word \( w \).
The CFL Membership Problem

Example (The CFL Membership Problem)

- Earlier we saw an algorithm for testing whether a word $w$ is derivable from a CNF grammar $G$.
- The algorithm pursued every possible derivation of length $2m - 1$.
- There problem is, the number of such derivations may be exponential in $m$. (Why?)
- We need a different algorithm.
Our strategy will be to begin with all one-symbol substrings of $w$ that are derivable from $G$.

These come directly from the rules of the form $A \rightarrow a$.

Then proceed inductively.
Example (The CFL Membership Problem)

- Let $w = x_1 x_2 x_3 \ldots x_m$.
- Set up a table of size $m \times m$.
- For all $i \leq j$, table entry $(i, j)$ is the set of all variables $A$ in $G$ from which the substring $x_i \ldots x_j$ is derivable.
- For example,
  - Table entry $(1, 1) = \{ A \in V \mid A \Rightarrow^* x_1 \}$.
  - Table entry $(3, 5) = \{ A \in V \mid A \Rightarrow^* x_3 x_4 x_5 \}$.
  - Table entry $(1, m) = \{ A \in V \mid A \Rightarrow^* w \}$. 
Example (The CFL Membership Problem)

- Basic step:
  - Initialize table entries \((i, i)\) to the set of all variables \(A\) for which there is a rule \(A \rightarrow x_i\).
Example (The CFL Membership Problem)

- Let the grammar be

  \[ S \rightarrow SS \mid aSb \mid bSa \mid \varepsilon \]

- In CNF it is

  \[ S \rightarrow SS \mid AX \mid YA \mid AB \mid BA \]
  \[ X \rightarrow SB \]
  \[ Y \rightarrow BS \]
  \[ A \rightarrow a \]
  \[ B \rightarrow b \]
Example (The CFL Membership Problem)

For the word $w = \text{abbaba}$
Example (The CFL Membership Problem)

- **Inductive step:**
  - For each $j \geq 1$, for each $i < m$, for each rule $A \rightarrow BC$, and for each $k$ from $i$ to $i + j - 1$,
    - See whether $B$ is in table entry $(i, k)$ and $C$ is in table entry $(k + 1, i + j)$.
    - If so, then add $A$ to table entry $(i, i + j)$.
  - The idea is that if $B \Rightarrow^* x_i \ldots x_k$ and $C \Rightarrow^* x_{k+1} \ldots x_{i+j}$, then $A \Rightarrow BC \Rightarrow^* x_i \ldots x_{i+j}$. 
Example (The CFL Membership Problem)

- For example, when \( j = 1 \), consider the case where \( i = 1 \).
- In this case, \( i + j = 2 \), so \( k \) must equal 1.
- Table entry \((1, 1) = \{A\}\) and table entry \((2, 2) = \{B\}\).
- There are rules \( S_0 \rightarrow AB \) and \( S \rightarrow AB \).
- So table entry \((1, 2) = \{S\}\).
- That is, \( S \Rightarrow^* ab \).
Example (The CFL Membership Problem)

For $j = 1$
Example (The CFL Membership Problem)

- When $j = 2$, consider the case where $i = 1$.
- In this case, $i + j = 3$, so $k$ may equal 1 or 2.
- Case $k = 1$:
  - Table entry $(1, 1) = \{A\}$ and table entry $(2, 3) = \emptyset$.
  - This produces nothing.
- Case $k = 2$:
  - Table entry $(1, 2) = \{S\}$ and table entry $(3, 3) = \{B\}$.
  - There is the rule $X \rightarrow SB$.
- So table entry $(1, 3) = \{X\}$.
- That is, $X \xrightarrow{2} abb$. 

Robb T. Koether  (Hampden-Sydney College)  
The Classes P and NP  
Mon, Nov 24, 2014  
46 / 62
**Example (The CFL Membership Problem)**

For $j = 2$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$X$</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$S$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>$\emptyset$</td>
<td>$Y$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>$S$</td>
<td>$X, Y$</td>
<td>$A$</td>
<td>$S$</td>
</tr>
<tr>
<td>$A$</td>
<td>$B$</td>
<td>$S$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$A$</td>
<td>$S$</td>
<td></td>
<td>$A$</td>
</tr>
</tbody>
</table>

Robb T. Koether (Hampden-Sydney College)
### Example (The CFL Membership Problem)

For $j = 3$.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$S$</td>
<td>$X$</td>
<td>$S$</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>$\emptyset$</td>
<td>$Y$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>$S$</td>
<td>$X, Y$</td>
<td>$S$</td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>$S$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$B$</td>
<td>$S$</td>
<td></td>
<td>$A$</td>
</tr>
</tbody>
</table>
Example (The CFL Membership Problem)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>S</th>
<th>X</th>
<th>S</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Ø</td>
<td>Y</td>
<td>Ø</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>S</td>
<td>X, Y</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>S</td>
<td>Ø</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For $j = 4$
Example (The CFL Membership Problem)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>S</th>
<th>X</th>
<th>S</th>
<th>X</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Ø</td>
<td>Y</td>
<td>Ø</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>S</td>
<td>X, Y</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>S</td>
<td>Ø</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For $j = 5$
Finally, see whether the start symbol is in table entry \((1, m)\).
- If it is, then accept \(w\).
- If it is not, the reject \(w\).

In the example, the start symbol \(S\) is in table entry \((1, 6)\).
CFLMember(graph G, string w)
{
    for (int i = 1; i <= m; i++)
        if ((A → w[i]) ∈ R)
            Add A to table[i, i];
    for (int len = 2; len <= m; len++)
        for (int i = 1; i <= m - len + 1; i++)
            { 
                int j = i + len - 1;
                for (int k = i; k <= j - 1; k++)
                    for each ((A → BC) ∈ R)
                        if (B ∈ table[i, k] && C ∈ table[k+1, j])
                            Add A to table[i, j];
            }
    if (S ∈ table[1, m])
        accept();
    else
        reject();
}
Example (The CFL Membership Problem)

- We can analyze each step in the function \texttt{CFLMember}().
- Each step is done in polynomial time.
- Thus, CFL $\in$ P.
Outline

1. Assignment
2. Time Complexity Classes
   - Polynomial Time
   - Exponential Time
   - Factorial Time
3. Comparison of Run Times
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
When we solve a problem nondeterministically in polynomial time, we do not just magically write down the solution. We must be able to
- Generate a solution in polynomial time, and
- Verify the solution in polynomial time.
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
The COMPOSITES Problem

Given an integer $m$, determine whether it is composite.
The COMPOSITES Problem

- How do we generate a solution in polynomial time?
- Let $n$ be the length of the integer $m$, in bits.
- To choose a divisor $a$, choose $n$ bits to create the binary representation of $a$.
- Then do the same for $b$.
- That can be done in polynomial time (in fact, linear time).
The COMPOSITES Problem

- How do we verify that \( m = ab \)?
- We just multiply \( a \) times \( b \) and compare to \( m \).
- That too can be done in polynomial time.
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
Verifiers

Definition (Verifier, Certificate)

A verifier for a language $A$ is an algorithm $V$ such that

$$A = \{ w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c \}$$

where $c$ is a string, called a certificate, containing additional information needed for verification.
Verifiers

- The language of the COMPOSITES problem is \{\langle m \rangle \mid m \text{ is composite}\}.
- The verifier is a Turing machine that accepts \langle m, a, b \rangle whenever \( m = ab \).
- The certificate is \langle a, b \rangle.
Outline

1. Assignment
2. Time Complexity Classes
3. Comparison of Run Times
   - Polynomial Time
   - Exponential Time
   - Factorial Time
4. Other Machine Models
5. The Class P
   - The PATH Problem
   - The CFL Membership Problem
6. Nondeterministic Algorithms
   - Generators
   - Verifiers
7. The Class NP
The Class NP

Definition (The class NP)

Let $\text{NTIME}(t(n))$ be the set of all languages that are decidable nondeterministically in time $O(t(n))$. Then

$$\text{NP} = \bigcup_{k=0}^{\infty} \text{NTIME}(n^k).$$

The class NP is the class of all languages whose members can be generated in polynomial time and that have polynomial-time verifiers.