Computability and Complexity

Lecture 9

More examples of problems in P
Closure properties of the class P
The class NP

given by Jiri Srba

Example: Relatively Prime

Definition

Natural numbers x and y are relatively prime iff $gcd(x, y) = 1$.

$gcd(x, y)$ ... the greatest common divisor of x and y

$$RELPRIME \stackrel{\mathrm{def}}{=} \{\langle x, y\rangle \mid \ x \text{ and } y \text{ are relatively prime numbers } \}$$

Remember our agreement about encoding of numbers:

- x and y are encoded in binary,
- so the length of $\langle x, y\rangle$ is $O(\log(x + y))$.

Brute-Force Algorithm is Exponential

Given an input $\langle x, y\rangle$ of length $n = |\langle x, y\rangle|$, going through all numbers between 2 and $\min\{x, y\}$ and checking whether some of them divide both x and y takes time exponential in n.

Euclidean algorithm for finding $gcd(x, y)$:

function $gcd(\langle x, y\rangle) \stackrel{\mathrm{def}}{=}$
if $(y == 0)$ return x else return $gcd(\langle y, x \mod y\rangle)$

Solving *RELPRIME* in P

Euclidean algorithm for finding $gcd(x, y)$:

function $gcd(\langle x, y\rangle) \stackrel{\text{def}}{=}$
if $(y == 0)$ return x else return $gcd(\langle y, x \mod y\rangle)$

Theorem

- The Euclidean algorithm called on input $\langle x, y\rangle$ runs in time $O(\log(x + y))$.
- Hence its running time is $O(n)$ (its input is encoded in binary).

Solving *RELPRIME* in P

Euclidean algorithm for finding $gcd(x, y)$:

function $gcd(\langle x, y\rangle) \stackrel{\text{def}}{=}$
if $(y == 0)$ return x else return $gcd(\langle y, x \mod y\rangle)$

Theorem

- The Euclidean algorithm called on input $\langle x, y\rangle$ runs in time $O(\log(x + y))$.
- Hence its running time is $O(n)$ (its input is encoded in binary).

Conclusion

$RELPRIME \in \mathrm{P}$

Example: Context-Free Languages

Theorem

Every context-free language is in P.

Proof:

- Let L be a CFL. Then there is CFG G in Chomsky normal form s.t. $L(G) = L$.
- For any given string $w = w_1 w_2 \dots w_n$ we want to decide in polynomial time whether $w \in L(G)$ or not.

Example: Context-Free Languages

Theorem

Every context-free language is in P.

Proof:

- Let L be a CFL. Then there is CFG G in Chomsky normal form s.t. $L(G) = L$.
- For any given string $w = w_1 w_2 \ldots w_n$ we want to decide in polynomial time whether $w \in L(G)$ or not.

Problem of the naive approach:

Brute-force algorithm (i.e. enumerating all derivations of the length $2n - 1$) takes exponential time!

Example: Context-Free Languages

Theorem

Every context-free language is in P.

Proof:

- Let L be a CFL. Then there is CFG G in Chomsky normal form s.t. $L(G) = L$.
- For any given string $w = w_1 w_2 \ldots w_n$ we want to decide in polynomial time whether $w \in L(G)$ or not.

Problem of the naive approach:

Brute-force algorithm (i.e. enumerating all derivations of the length $2n - 1$) takes exponential time!

Solution:

We use dynamic programming instead.

Checking whether $w \in L(G)$ using Dynamic Programming

Idea (for a given grammar G in Chomsky normal form):

On input $w = w_1 w_2 \ldots w_n$ create temporary sets of nonterminals called *table(i, j)* for $1 \leq i \leq j \leq n$ such that

- $A \in table(i, j)$ if and only if $A \Rightarrow^* w_i w_{i+1} \ldots w_j$.

Checking whether $w \in L(G)$ using Dynamic Programming

Idea (for a given grammar G in Chomsky normal form):

On input $w = w_1 w_2 \ldots w_n$ create temporary sets of nonterminals called $table(i,j)$ for $1 \leq i \leq j \leq n$ such that

- $A \in table(i,j)$ if and only if $A \Rightarrow^* w_i w_{i+1} \ldots w_j$.

"On input $w = w_1 w_2 \ldots w_n$:

1. If $w = \epsilon$ then accept if $S \rightarrow \epsilon$ is a rule in G, else reject.
2. For i:=1 to n do: if $A \rightarrow w_i$ is a rule in G, add A to $table(i,i)$.
3. For ℓ:=2 to n do:

   for all i, k such that $1 \leq i \leq k < \underbrace{i + \ell - 1}_{j} \leq n$ do:

   for all rules $A \rightarrow BC$ in G do:

   if $B \in table(i,k)$ and $C \in table(k+1, j)$ then add A to $table(i,j)$.
4. If $S \in table(1,n)$ then accept, else reject."

The algorithm runs in $O(n^3)$.

Theorem (Closure Properties of the Class P)

The class P is closed under intersection, union, complement, concatenation and Kleene star.

In other words:

If L_1 and L_2 are decidable in deterministic polynomial time, then

- $L_1 \cap L_2, \quad L_1 \cup L_2, \quad \overline{L_1}, \quad L_1.L_2, \quad$ and L_1^*

are decidable in deterministic polynomial time too.

Proof: Closure of Decidable Languages under Union

Let $L_1, L_2 \in \mathrm{P}$. We want to show that $L_1 \cup L_2 \in \mathrm{P}$.

Because $L_1, L_2 \in \mathrm{P}$ then there is

- a decider M_1 for L_1 running in time $O(n^k)$ for some k, and
- a decider M_2 for L_2 running in time $O(n^\ell)$ for some ℓ.

The following 2-tape TM M is a decider for $L_1 \cup L_2$:

M= "On input x:
1. copy x on the second tape
2. on the first tape run M_1 on x
3. if M_1 accepted then <u>accept</u> else goto step 4
4. on the second tape run M_2 on x
5. if M_2 accepted then <u>accept</u> else <u>reject</u>."

Proof: Closure of Decidable Languages under Union

Let $L_1, L_2 \in \mathrm{P}$. We want to show that $L_1 \cup L_2 \in \mathrm{P}$.

Because $L_1, L_2 \in \mathrm{P}$ then there is

- a decider M_1 for L_1 running in time $O(n^k)$ for some k, and
- a decider M_2 for L_2 running in time $O(n^\ell)$ for some ℓ.

The following 2-tape TM M is a decider for $L_1 \cup L_2$:

M= "On input x:

1. copy x on the second tape
2. on the first tape run M_1 on x
3. if M_1 accepted then <u>accept</u> else goto step 4
4. on the second tape run M_2 on x
5. if M_2 accepted then <u>accept</u> else <u>reject</u>."

M runs in time $O(n^k) + O(n^\ell) = O(n^c)$ where $c = \max\{k, \ell\}$.

Proof: Closure of Decidable Languages under Union

Let $L_1, L_2 \in \mathrm{P}$. We want to show that $L_1 \cup L_2 \in \mathrm{P}$.

Because $L_1, L_2 \in \mathrm{P}$ then there is

- a decider M_1 for L_1 running in time $O(n^k)$ for some k, and
- a decider M_2 for L_2 running in time $O(n^\ell)$ for some ℓ.

The following 2-tape TM M is a decider for $L_1 \cup L_2$:

M=

"On input x:

1. copy x on the second tape
2. on the first tape run M_1 on x
3. if M_1 accepted then accept else goto step 4
4. on the second tape run M_2 on x
5. if M_2 accepted then accept else reject."

M runs in time $O(n^k) + O(n^\ell) = O(n^c)$ where $c = \max\{k, \ell\}$.
M can be simulated by a single-tape TM running in time $O((n^c)^2) = O(n^{2c})$, hence $L(M) \in \mathrm{P}$ because $2c$ is a constant. □

Running Time of a Nondeterministic TM

Definition (Running Time of a Nondeterministic TM)

Let M be a nondeterministic decider. The running time or (worst-case) time complexity of M is a function

$$f : \mathbb{N} \to \mathbb{N}$$

where $f(n)$ is the maximum number of steps that M uses on any branch of its computation tree for any input of length n.

Running Time of a Nondeterministic TM

Definition (Running Time of a Nondeterministic TM)

Let M be a nondeterministic decider. The running time or (worst-case) time complexity of M is a function

$$f : \mathbb{N} \to \mathbb{N}$$

where $f(n)$ is the maximum number of steps that M uses on any branch of its computation tree for any input of length n.

Theorem

Let $t(n)$ be a function s.t. $t(n) \geq n$.

Every nondeterministic TM running in time $t(n)$ has an equivalent deterministic TM running in time $2^{O(t(n))}$.

Proof: Simulate a nondeterministic TM M by a deterministic TM M' (from Lecture 2) and analyze the running time of M'. $\square$

The Complexity Class NTIME($t(n)$)

Definition (Time Complexity Class NTIME($t(n)$))

Let $t : \mathbb{N} \to \mathbb{R}^{>0}$ be a function.

NTIME($t(n)$) $\stackrel{\text{def}}{=}$
$\{L(M) \mid M$ is a nondeterministic decider running in time $O(t(n))\}$

The Complexity Class NTIME($t(n)$)

Definition (Time Complexity Class NTIME($t(n)$))

Let $t : \mathbb{N} \to \mathbb{R}^{>0}$ be a function.

NTIME($t(n)$) $\stackrel{\text{def}}{=}$
$\{L(M) \mid\ M$ is a nondeterministic decider running in time $O(t(n))\}$

In other words: NTIME($t(n)$) is the class (collection) of languages that are decidable by nondeterministic TMs in time $O(t(n))$.

Example:

- $\mathit{HAMPATH} \stackrel{\text{def}}{=} \{\langle G, s, t\rangle \mid$
  G is a directed graph with a Hamiltonian path from s to t $\}$
- $\mathit{HAMPATH} \in$ NTIME(n^2)

$HAMPATH \in \mathrm{NTIME}(n^2)$

Consider the following nondeterministic decider for $HAMPATH$:

"On input $\langle G, s, t\rangle$:

1. Nondeterministically select a sequence of nodes $v_1, v_2, \ldots, v_m$ where m is the number of nodes in G.
2. Verify that every node appears in the sequence exactly once. If not then reject.
3. Verify that $v_1 = s$ and $v_m = t$. If not then reject.
4. For each $i := 1$ to $m - 1$ verify if there is an edge from v_i to v_{i+1}. If not then reject.
5. All test passed so accept."

$HAMPATH \in \mathrm{NTIME}(n^2)$

Consider the following nondeterministic decider for $HAMPATH$:

"On input $\langle G, s, t\rangle$:

1. Nondeterministically select a sequence of nodes $v_1, v_2, \ldots, v_m$ where m is the number of nodes in G.
2. Verify that every node appears in the sequence exactly once. If not then reject.
3. Verify that $v_1 = s$ and $v_m = t$. If not then reject.
4. For each $i := 1$ to $m - 1$ verify if there is an edge from v_i to v_{i+1}. If not then reject.
5. All test passed so accept."

The nondeterministic decider runs in time $O(n^2)$.

The Class NP

Definition

The class NP is the class of languages decidable in polynomial time on nondeterministic single-tape Turing machine, i.e.,

$$\mathsf{NP} \stackrel{\mathrm{def}}{=} \bigcup_{k \geq 0} \mathsf{NTIME}(n^k) \ .$$

The Class NP

Definition

The class NP is the class of languages decidable in polynomial time on nondeterministic single-tape Turing machine, i.e.,

$$\mathrm{NP} \stackrel{\text{def}}{=} \bigcup_{k \geq 0} \mathrm{NTIME}(n^k) \ .$$

Example: $\mathit{HAMPATH} \in \mathrm{NP}$

The Class NP

Definition

The class NP is the class of languages decidable in polynomial time on nondeterministic single-tape Turing machine, i.e.,

$$\mathrm{NP} \stackrel{\text{def}}{=} \bigcup_{k \geq 0} \mathrm{NTIME}(n^k) \ .$$

Example: $\quad HAMPATH \in \mathrm{NP}$

Discussion:

- The class NP is robust (the class remains the same even if we choose some other nondeterministic model of computation).
- Every problem from NP can be solved in exponential time on a deterministic TM.
- $\mathrm{P} \subseteq \mathrm{NP}$ (every determin. TM is a nondetermin. TM too)
- The question whether P=NP is open.

Exam Questions

- *RELPRIME* and any context-free language are in P.
- Closure properties of the class P.
- Nondeterministic time complexity, the classes NTIME($t(n)$) and NP.
- Simulation of nondeterministic TM by a deterministic one with exponential increase in a running time.