Algorithms and Complexity: Turing machines

Marie Schmidt & Tom van der Zanden

(Based on slides by Gerhard Woeginger and Jesper Nederlof)

Landelijk Netwerk Mathematische Besliskunde

LNMB

Complexity classes P and NP

Definition: Complexity class P

A decision problem X lies in the complexity class P,

- **if it can be solved on a deterministic Turing machine in polynomial time** (original, formal definition)
- (or, alternatively:) if it is solved by an algorithm with polynomial time complexity (definition that we use)

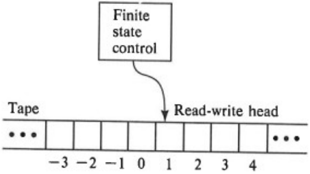
Definition: Complexity class NP

A decision problem X lies in the complexity class NP, if

- **if it can be solved in polynomial time on a non-deterministic Turing machine** (original, formal definition)
- (or, alternatively:) if it is solved by a *non-deterministic* algorithm with polynomial time complexity.
- (or, alternatively:) if the YES-instances of X possess certificates of polynomial length that can be verified in polynomial time.

Turing machines

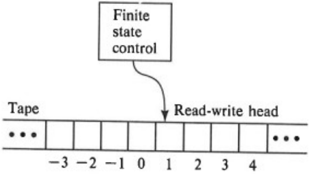
Not talking about this.



But this!

Turing machines

Not talking about this.

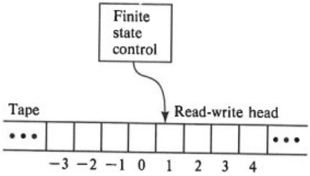


But this!
$\rightarrow$ Alternative model of computation

Deterministic one-tape Turing machine (DTM)

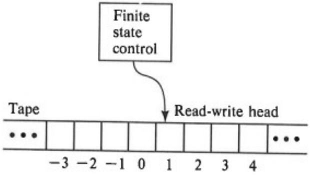
A DTM consists of

1. a *finite state control*
2. a *read-write head*
3. a *tape*: two-way infinite sequence of tape squares



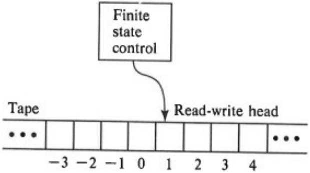
A program for a DTM specifies:

1. a finite set Γ of tape symbols, including a subset $\Sigma \subset \Gamma$ of *input symbols* and a distinguished *blank symbol* $b \in \Gamma \setminus \Sigma$



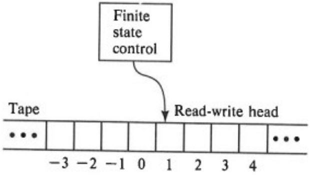
A program for a DTM specifies:

1. a finite set Γ of tape symbols, including a subset $\Sigma \subset \Gamma$ of *input symbols* and a distinguished *blank symbol* $b \in \Gamma \setminus \Sigma$
2. a finite set Q of *states*, inclusing a distinguished *start state* q_0 and two distinguished *halt states* q_Y and q_N



A program for a DTM specifies:

1. a finite set Γ of tape symbols, including a subset $\Sigma \subset \Gamma$ of *input symbols* and a distinguished *blank symbol* $b \in \Gamma \setminus \Sigma$
2. a finite set Q of *states*, inclusing a distinguished *start state* q_0 and two distinguished *halt states* q_Y and q_N
3. a *transition function* $\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{-1, 1\}$



Operation of a DTM program

Input: finite string $x \in \Sigma$

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

read-write head scans tape square 1

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

read-write head scans tape square 1

while $q \notin \{q_Y, q_N\}$ **do**

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

read-write head scans tape square 1

while $q \notin \{q_Y, q_N\}$ **do**

look up $(q', s', \Delta) := \delta(q, s)$ for current state q and read-write head pointing at square with symbol s

Operation of a DTM program

Input: finite string $x \in \Sigma$
Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),
state $q = q_0$,
read-write head scans tape square 1
while $q \notin \{q_Y, q_N\}$ **do**
look up $(q', s', \Delta) := \delta(q, s)$ for current state q and read-write head pointing at square with symbol s
erase s

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

read-write head scans tape square 1

while $q \notin \{q_Y, q_N\}$ **do**

look up $(q', s', \Delta) := \delta(q, s)$ for current state q and read-write head pointing at square with symbol s

erase s

write s' in its place

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

read-write head scans tape square 1

while $q \notin \{q_Y, q_N\}$ **do**

look up $(q', s', \Delta) := \delta(q, s)$ for current state q and read-write head pointing at square with symbol s

erase s

write s' in its place

move one square to the left if $\Delta = -1$, one square to the right if $\Delta = 1$

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

read-write head scans tape square 1

while $q \notin \{q_Y, q_N\}$ **do**

look up $(q', s', \Delta) := \delta(q, s)$ for current state q and read-write head pointing at square with symbol s

erase s

write s' in its place

move one square to the left if $\Delta = -1$, one square to the right if $\Delta = 1$

set $q := q'$

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),
state $q = q_0$,
read-write head scans tape square 1
while $q \notin \{q_Y, q_N\}$ **do**
  look up $(q', s', \Delta) := \delta(q, s)$ for current state q and read-write head pointing at square with symbol s
  erase s
  write s' in its place
  move one square to the left if $\Delta = -1$, one square to the right if $\Delta = 1$
  set $q := q'$
end while
if $q = q_Y$ **then**
  return YES
else
  return NO
end if

Operation of a DTM program

Input: finite string $x \in \Sigma$

Initialize: write string in tape squares 1 to $|x|$, one symbol per square (all other tape squares are blank),

state $q = q_0$,

read-write head scans tape square 1

while $q \notin \{q_Y, q_N\}$ **do**

look up $(q', s', \Delta) := \delta(q, s)$ for current state q and read-write head pointing at square with symbol s

erase s

write s' in its place

move one square to the left if $\Delta = -1$, one square to the right if $\Delta = 1$

set $q := q'$

end while

if $q = q_Y$ **then**

return YES

else

return NO

end if

Each iteration of the while-loop counts as a step

Example: A program for a DTM machine

$\Gamma = \{0, 1, b\}$, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_Y, q_N\}$

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$\delta(q, s)$

- What does this program do?

Example: A program for a DTM machine

$\Gamma = \{0, 1, b\}$, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_Y, q_N\}$

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$\delta(q, s)$

- What does this program do?
- Try it out on different strings, e.g., 0111, 010, 11110, 000000, ...

Example: A program for a DTM machine

$\Gamma = \{0, 1, b\}$, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_Y, q_N\}$

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$\delta(q, s)$

- What does this program do?
- Try it out on different strings, e.g., 0111, 010, 11110, 000000, ...
- When does it return 'yes' and when 'no'?

Example: A program for a DTM machine

$\Gamma = \{0, 1, b\}$, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_Y, q_N\}$

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$\delta(q, s)$

- What does this program do?
- Try it out on different strings, e.g., 0111, 010, 11110, 000000, ...
- When does it return 'yes' and when 'no'?
- How many steps do we need in these examples?

Example: A program for a DTM machine

$\Gamma = \{0, 1, b\}$, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_Y, q_N\}$

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$\delta(q, s)$

- What does this program do?
- Try it out on different strings, e.g., 0111, 010, 11110, 000000, ...
- When does it return 'yes' and when 'no'?
- How many steps do we need in these examples?
- How many steps do we need at most for a string of length n?

Example: A program for a DTM machine

$\Gamma = \{0, 1, b\}$, $\Sigma = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_Y, q_N\}$

q	0	1	b
q_0	$(q_0, 0, +1)$	$(q_0, 1, +1)$	$(q_1, b, -1)$
q_1	$(q_2, b, -1)$	$(q_3, b, -1)$	$(q_N, b, -1)$
q_2	$(q_Y, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$
q_3	$(q_N, b, -1)$	$(q_N, b, -1)$	$(q_N, b, -1)$

$\delta(q, s)$

- What does this program do?
- Try it out on different strings, e.g., 0111, 010, 11110, 000000, ...
- When does it return 'yes' and when 'no'?
- How many steps do we need in these examples?
- How many steps do we need at most for a string of length n?
- How much space do we need at most for a string of length n?

Would you rather own a RAM or a DTM?

Would you rather own a RAM or a DTM?

Equivalence of computational models

A RAM and a DTM are equivalent in the sense that any function that can be computed on a DTM can be computed on a RAM, and vice versa.

Would you rather own a RAM or a DTM?

Equivalence of computational models

A RAM and a DTM are equivalent in the sense that any function that can be computed on a DTM can be computed on a RAM, and vice versa.

Church-Turing thesis

Anything that can be calculated by an *effective method* can be computed by a deterministic Turing machine.

Complexity classes P and NP

Definition: Complexity class P

A decision problem X lies in the complexity class P,

- **if it can be solved on a deterministic Turing machine in polynomial time** (original, formal definition)
- (or, alternatively:) if it is solved by an algorithm with polynomial time complexity (definition that we use)

Non-deterministic Turing machine

Non-deterministic Turing machine (NDTM)

1. guessing module: write-only head
2. checking module: deterministic Turing machine

Non-deterministic Turing machine

A program for a NDTM specifies:

exactly the same as a DTM program:

1. finite set of tape symbols Γ of tape symbols, including blank symbol
2. finite set Q of *states*, i
3. *transition function* δ

Non-deterministic Turing machine

A program for a NDTM specifies:

exactly the same as a DTM program:

1. finite set of tape symbols Γ of tape symbols, including blank symbol
2. finite set Q of *states*, i
3. *transition function* δ

Operation of a NDTM program

- write input string in tape squares 1 to $|x|$

Non-deterministic Turing machine

A program for a NDTM specifies:

exactly the same as a DTM program:

1. finite set of tape symbols Γ of tape symbols, including blank symbol
2. finite set Q of *states*, i
3. *transition function* δ

Operation of a NDTM program

- write input string in tape squares 1 to $|x|$
- **guessing module: writes finite string of symbols from Γ in left tape squares starting from -1 in arbitrary manner**

Non-deterministic Turing machine

A program for a NDTM specifies:

exactly the same as a DTM program:

1. finite set of tape symbols Γ of tape symbols, including blank symbol
2. finite set Q of *states*, i
3. *transition function* δ

Operation of a NDTM program

- write input string in tape squares 1 to $|x|$
- **guessing module: writes finite string of symbols from Γ in left tape squares starting from -1 in arbitrary manner**
- checking module: operates like a DTM

Non-deterministic Turing machine

Operation of a NDTM program

- write input string in tape squares 1 to $|x|$
- guessing module: writes finite string of symbols from Γ in tape squares starting from -1 in arbitrary manner
- checking module: operates like a DTM

Note: For a given string x and a given NDTM program, there is an *infinite* number of possible computations possible (one for each 'guessed' string)

Non-deterministic Turing machine

Operation of a NDTM program

- write input string in tape squares 1 to $|x|$
- guessing module: writes finite string of symbols from Γ in tape squares starting from -1 in arbitrary manner
- checking module: operates like a DTM

Note: For a given string x and a given NDTM program, there is an *infinite* number of possible computations possible (one for each 'guessed' string)

Terminology & definitions

Accepting computation: all computations that terminate in accepting state (q_Y).
Non-accepting computations: all computations that terminate in non-accepting-state (q_N) or do not terminate at all.

NDTM program M **accepts** x if *there is* an accepting computation for x on M.

The **time complexity** of an NDTM program for a string x is defined as the *minimum* running time over all accepting computations of x by M.
The worst-case time-complexity of an NDTM program is the maximum time complexity over all strings x of a certain length n that are accepted by M.

Non-deterministic algorithm

non-deterministic algorithm $\hat{=}$ program for a non-deterministic Turing machine

1. Oracle/guessing stage
2. Checking stage

time complexity of a non-deterministic algorithm
$\hat{=}$ time complexity of the corresponding program

Complexity classes P and NP

Definition: Complexity class P

A decision problem X lies in the complexity class P,

- **if it can be solved on a deterministic Turing machine in polynomial time** (original, formal definition)
- (or, alternatively:) if it is solved by an algorithm with polynomial time complexity (definition that we use)

Definition: Complexity class NP

A decision problem X lies in the complexity class NP, if

- **if it can be solved in polynomial time on a non-deterministic Turing machine** (original, formal definition)
- (or, alternatively:) if it is solved by a *non-deterministic* algorithm with polynomial time complexity.
- (or, alternatively:) if the YES-instances of X possess certificates of polynomial length that can be verified in polynomial time.

Warnings:

1. A non-deterministic Turing machine is a *theoretical* construct, not an actual machine!
2. The Church-Turing thesis relates to *deterministic* Turing machines. (*'Anything that can be calculated by an effective method can be computed by a 'deterministic' Turing machine.'*