Chapter 1 - Time and Space Complexity

- deterministic and non-deterministic Turing machine
- time and space complexity
- classes $\mathbf{P}, \mathbf{NP}, \mathbf{PSPACE}, \mathbf{NPSPACE}$

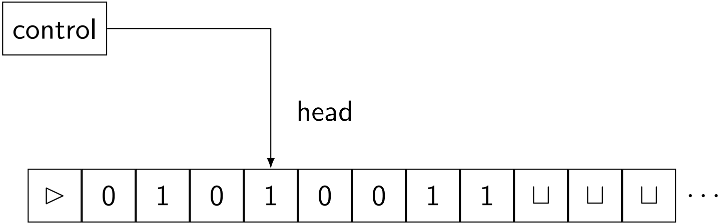
Deterministic Turing machines

Definition 1.1

A (deterministic 1-tape) Turing machine (DTM) is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, *where* Q, Σ, Γ *are finite sets and*

1. Q *is the set of states,*
2. Σ *is the input alphabet not containing the start symbol* $\triangleright$ *and the blank symbol* $\sqcup$,
3. Γ *is the tape alphabet, where* $\Sigma \subset \Gamma$ *and* $\sqcup, \triangleright \in \Gamma$,
4. $\delta : Q \backslash \{q_{\text{accept}}, q_{\text{reject}}\} \times \Gamma \to Q \times \Gamma \times \{R, L\}$ *is the transition function,*
5. q_0 *is the start state,* q_{accept} *is the accept state, and* q_{reject} *is the reject state.*

Schematic of a Turing machine



Transition function

Semantics of transitions

$\delta(q_i, a) = (q_j, b, X)$ means that, if the machine is in state q_i and the head reads symbols a, then

1. the machine goes to state q_j,
2. the head writes the symbol b on the tape,
3. the machine directs the head to move right ($X = R$) or to move left ($X = L$).

Restrictions

We always assume the following restrictions on δ:

- For all $q \in Q$:

$$\delta(q, \triangleright) = (p, \triangleright, R) \quad \text{for some} \quad p \in Q \ .$$

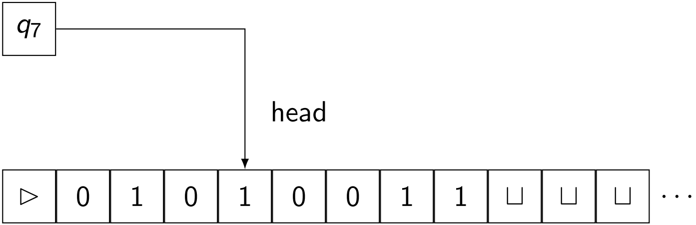
- For all $q \in Q$ and $a \in \Gamma, a \neq \triangleright$:

$$\delta(q, a) = (p, b, D) \quad \text{with} \quad p \in Q, b \in \Gamma, b \neq \triangleright, D \in \{L, R\} \ .$$

Configurations

- A *configuration* of a DTM M is an element in $\Gamma^* \times Q \times \Gamma^*$.
- M is in configuration $\alpha q \beta$, iff
  1. the left-most cells of the tape contain $\alpha\beta \in \Gamma^*$, all other tape cells contain $\sqcup$.
  2. the head of the Turing machine is on the first symbol of β,
  3. the state of the Turing machine is q.

Turing machine in configuration $\triangleright 010 q_7 10011$



Computations - single steps

- step of a DTM $\triangleq$ single application of transition function
- computation $\triangleq$ sequence of steps
- configuration C_1 *yields* configuration C_2 iff DTM M can legally go from C_1 to C_2 in one step

- $C_1 = uaq_ibv, C_2 = uq_jacv, q_i, q_j \in Q, a, b, c \in \Gamma, u, v \in \Gamma^*,$
- C_1 yields C_2, iff $\delta(q_i, b) = (q_j, c, L)$.
- $C_1 = uaq_ibv, C_2 = uacq_jv, q_i, q_j \in Q, a, b, c \in \Gamma, u, v \in \Gamma^*,$
- C_1 yields C_2, iff $\delta(q_i, b) = (q_j, c, R)$.

Computations

- $q_0 \triangleright w \triangleq$ start configuration of DTM M on input w
- configuration C is an *accepting configuration* iff the state in C is q_{accept}
- configuration C is a *rejecting configuration* iff the state in C is q_{reject}
- accepting and rejecting configurations are *halting configurations*
- if a DTM M reaches a halting configuration the computation of M halts
- if M is started on some input and never reaches a halting state, we say that M *loops*

Computations

- DTM M accepts input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where
  1. C_1 is the start configuration of M on input w,
  2. each C_i yields C_{i+1},
  3. C_k is an accepting configuration.

Turing machines and languages

Definition 1.2

The set of words $w \in \Sigma^*$ that DTM M accepts is called the language accepted or recognized by M. We write

$$L(M) := \{w \in \Sigma^* \mid M \textit{ accepts } w.\}$$

Definition 1.3

DTM M decides $L(M)$, if M halts on every input $w \in \Sigma^*$.

Definition 1.4

1. *$L \subseteq \Sigma^*$ is called Turing-recognizable or recursively enumerable if some DTM M recognizes L.*
2. *$L \subseteq \Sigma^*$ is called Turing-decidable or decidable if some DTM M decides it.*

Time complexity

Definition 1.5

Let M be a DTM that halts on all inputs. The running time or time complexity of M is the function $f : \mathbb{N} \rightarrow \mathbb{N}$, where $f(n)$ is the maximum number of steps that M uses on any input of length n.

If $f(n)$ is the running time of M we say that M runs in time $f(n)$ and that M is an $f(n)$ time Turing machine.

Customarily, n denotes the length of the representation of the input.

Time complexity classes

Definition 1.6

Let $t : \mathbb{N} \to \mathbb{R}^+$ *be a monotonically increasing function. The time complexity class* $\mathbf{DTIME}(t(n))$ *consists of all languages that are decidable by an* $\mathcal{O}(t(n))$ *time DTM.*

Space complexity and space complexity classes

Definition 1.7

Let M be a DTM that halts on all inputs. The space complexity of M is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that M scans on any input of length n.

If the space complexity of M is $f(n)$ we say that M runs in space $f(n)$.

Definition 1.8

Let $s : \mathbb{N} \to \mathbb{R}^+$ be a monotonically increasing function. The space complexity class **DSPACE**$(s(n))$ *consists of all languages that are decidable by an $\mathcal{O}(s(n))$ space DTM.*

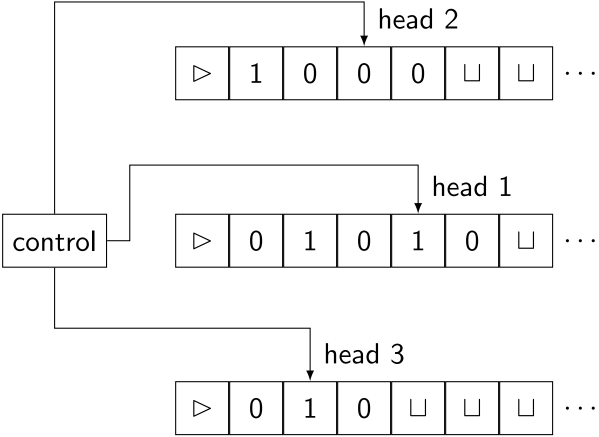
Multi-tape Turing machines

- A k-tape Turing machine (k-DTM) has k independent tapes, each with its own read/write head.
- The transition function of a k-tape Turing machine is of the form

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k \ .$$

- $\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, X_1, X_2, \ldots, X_k)$ means that, if the machine is in state q_i and the heads 1 through k read symbols $a_1, \ldots, a_k$, then
  1. the machine goes to state q_j,
  2. the heads 1 through k write the symbols $b_1, \ldots, b_k$ on their respective tapes,
  3. the machine directs each head to move right ($X_i = R$), to move left ($X_i = L$), or to stay put ($X_i = S$).

Schematic of a 3-tape Turing machine



Time and space complexity classes for multi-tape DTMs

Definition 1.9
Let $t : \mathbb{N} \to \mathbb{R}^+$ *be a monotonically increasing function. The time complexity class* $\mathbf{DTIME}_k(t(n))$ *consists of all languages that are decidable by an* $\mathcal{O}(t(n))$ *time* k*-DTM.*

Definition 1.10
Let $s : \mathbb{N} \to \mathbb{R}^+$ *be a monotonically increasing function. The space complexity class* $\mathbf{DSPACE}_k(s(n))$ *consists of all languages that are decidable by an* $\mathcal{O}(s(n))$ *space* k*-DTM.*

1-tape vs. k-tape DTMs

Theorem 1.11
If language L can be decided by a $\mathcal{O}(s(n))$ space k-DTM, then L can be decided by a $\mathcal{O}(s(n))$ space 1-DTM.

Theorem 1.12
If language L can be decided by a $\mathcal{O}(t(n))$ time k-DTM, then L can be decided by a $\mathcal{O}(t(n)^2)$ time 1-DTM.

Corollary 1.13
For all $k \in \mathbb{N}$

$$\mathbf{DTIME}_k(t(n)) \subseteq \mathbf{DTIME}(t(n)^2).$$

1-tape vs. k-tape DTMs

Theorem 1.14

There is a language L that can be decided by a $\mathcal{O}(n)$ time 2-DTM, but that cannot be decided by a 1-DTM with time complexity $o(n^2)$.

Corollary 1.15

Let $t : \mathbb{N} \to \mathbb{R}^+$ be a function with $t(n) = o(n^2)$. For all $k \in \mathbb{N}, k \geq 2$

$$\mathbf{DTIME}_k(n) \not\subseteq \mathbf{DTIME}(t(n)).$$

Classes **P** and **PSPACE**

Definition 1.16

P *is the class of languages that are decidable in polynomial time on a (single- or multi-tape) deterministic Turing machine. That is*

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathbf{DTIME}(n^k).$$

Definition 1.17

PSPACE *is the class of languages that are decidable in polynomial space on a (single- or multi-tape) deterministic Turing machine. That is*

$$\mathbf{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathbf{DSPACE}(n^k).$$

Time and space

Theorem 1.18
Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function with $f(n) \geq n$ for all $n \in \mathbb{N}$. If a language L is in $\mathbf{DSPACE}(f(n))$, then there is a $2^{\mathcal{O}(f(n))}$ time DTM that decides L.

Boolean formula and fully quantified Boolean formula

Boolean formula

- A Boolean formula $\psi(x_1, \ldots, x_l)$ is an expression over Boolean variables $x_1, \ldots, x_l$ and the Boolean operators $\wedge, \vee, \neg$.
- Example: $\psi = (x_1 \vee \neg x_2) \wedge x_3 \vee (\neg x_3 \vee \neg x_1)$.
- A fully quantified Boolean formula ϕ in prenex normal form is an expression of the form $Q_1 x_1 \ldots Q_l x_l \; \psi(x_1, \ldots, x_l)$, where
  1. ψ is a Boolean formula,
  2. $Q_i \in \{\exists, \forall\}, i = 1, \ldots, l$.
- Example: $\phi = \forall x_1 \forall x_2 \exists x_3 \; (x_1 \vee \neg x_2) \wedge x_3 \vee (\neg x_3 \vee \neg x_1)$.

Remark

A fully quantified Boolean formula is either true $\triangleq 1$ or false $\triangleq 0$.

Definition 1.19

The language TQBF is defined as

$$TQBF := \{\langle\phi\rangle \mid \phi \textit{ is a true fully quantified Boolean formula in prenex normal form.}\}$$

Example

$\phi = \forall x_1 \forall x_2 \exists x_3\ (x_1 \vee \neg x_2) \wedge x_3 \vee (\neg x_3 \vee \neg x_1)$ is an element of TQBF.

Theorem 1.20

$TQBF \in$ **PSPACE**.

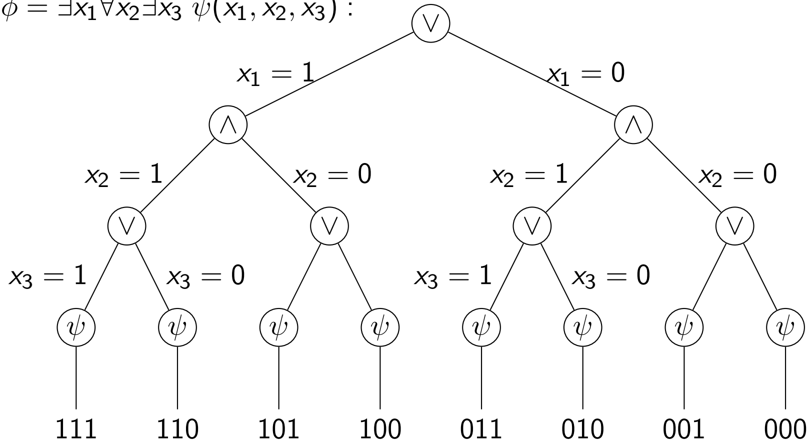
A space efficient algorithm for TQBF

$T =$ "On input $\langle \phi \rangle$, a fully quantified Boolean formula:

1. If ϕ contains no quantifiers, then ϕ contains only constants. Evaluate the expression.
2. If ϕ equals $\exists x\ \phi'$, recursively call T on ϕ', first with 0 substituted for x and then with 1 substituted for x. If either result is accept, then *accept*, else *reject*.
3. If ϕ equals $\forall x\ \phi'$, recursively call T on ϕ', first with 0 substituted for x and then with 1 substituted for x. If both results are accept, then *accept*, else *reject*."

Recursion tree

$\phi = \exists x_1 \forall x_2 \exists x_3 \ \psi(x_1, x_2, x_3)$:



Nondeterministic Turing machines

Power sets

For a set M, we denote by $\mathcal{P}(M)$ the power set of M, i.e. the set of all subsets of M.

Definition 1.21

A nondeterministic (1-tape) Turing machine (NTM) is a 7-tuple $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{accept}}, q_{\mathrm{reject}})$, where $Q, \Sigma, \Gamma, q_0, q_{\mathrm{accept}}, q_{\mathrm{reject}}$ are as for deterministic Turing machines. The transition function δ of a nondeterministic Turing machine is of the form

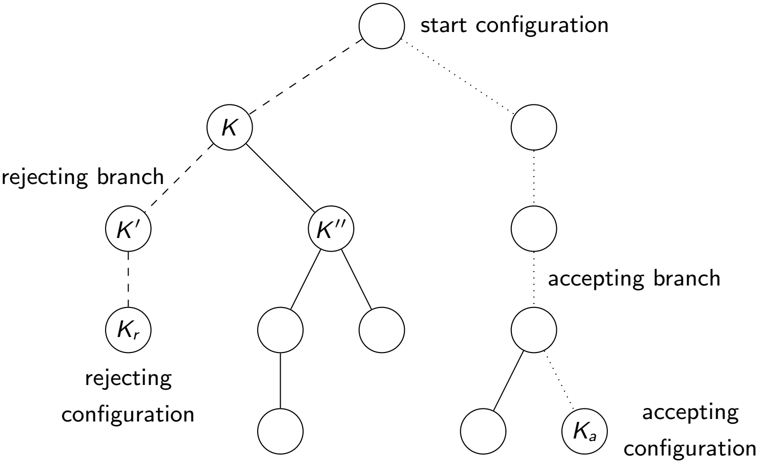
$$\delta : Q \backslash \{q_{\mathrm{accept}}, q_{\mathrm{reject}}\} \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{R, L\}).$$

Nondeterministic multi-tape Turing machines are defined similarly.

Computations of NTMs

- If $\delta(q_i, a) = \{(r_1, b_1, X_1), \ldots, (r_l, b_l, X_l)\}$ and if NTM N is in state q_i and reads symbol a, then it can perform any of the l steps described by the triples (r_j, b_j, X_j) in $\delta(q_i, a)$.
- Configurations, start configurations, accepting and rejecting configurations for NTMs are defined as for DTMs.
- Depending on the set $\delta(q_i, b)$ a configuration $C = uaq_ibv$ of an NTM can yield different configurations.
- Started with input $w \in \Sigma^*$ an NTM $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{accept}}, q_{\mathrm{reject}})$ can perform different computations that can be represented in a *computation tree*.

Computation tree of an NTM



NTMs and languages

- NTM $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ *accepts* $w \in \Sigma^*$ if there is a computation of N started with w that ends in an accepting configuration.
- The language $L(N)$ of words *recognized* by N is defined as

$$L(N) := \{w \in \Sigma^* \mid N \text{ accepts } w\}.$$

- N *always halts* if for every $w \in \Sigma^*$ every computation branch of N with input w is finite. An NTM N that always halts is called a *decider*.
- NTM N *decides* language $L(N)$ if N is a decider.

Nondeterministic time complexity

Definition 1.22
Let NTM N be a decider. The running time or time complexity of N is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of steps that N uses on any computation branch on any input of length n.

Definition 1.23
Let $t : \mathbb{N} \to \mathbb{R}^+$ be a monotonically increasing function. The time complexity class $\mathbf{NTIME}(t(n))$ consists of all languages that are decidable by an $\mathcal{O}(t(n))$ time NTM.

Nondeterministic space complexity

Definition 1.24

Let N be a decider. The space complexity of N is the function $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximum number of tape cells that N scans on any computation branch on any input of length n.

Definition 1.25

Let $s : \mathbb{N} \to \mathbb{R}^+$ be a monotonically increasing function. The space complexity class $\mathbf{NSPACE}(s(n))$ *consists of all languages that are decidable by an $\mathcal{O}(s(n))$ space NTM.*

Example of a linear space NTM

Problems on NFAs

- $ALL_{\mathsf{NFA}} := \{\langle A \rangle \mid A \text{ is an NFA and } L(A) = \Sigma^*\}$
- $\overline{ALL_{\mathsf{NFA}}}$ language consisting of all NFAs that reject at least one word over their input alphabet.

Example of a linear space NTM

$N =$ "On input $\langle A \rangle$, where A is an NFA:

1. Place a marker on the start state of A.
2. *Accept* if the start state is not an accept state.
3. Repeat 2^q times, where q is the number of states of A:
4. Nondeterministically select an input symbol and change the positions of the markers on A's states to simulate reading that symbol.
5. *Accept* if none of the markers lie on accept states of A.
6. *Reject*."

Theorem 1.26

N is a decider for $\overline{ALL_{NFA}}$ with space complexity $\mathcal{O}(n)$, i.e. $\overline{ALL_{NFA}} \in \mathbf{NSPACE}(n)$.

Nondeterministic polynomial time and space

Definition 1.27
$\mathbf{NP} = \bigcup_{k\in\mathbb{N}} \mathbf{NTIME}(n^k)$.

Definition 1.28
$\mathbf{NPSPACE} = \bigcup_{k\in\mathbb{N}} \mathbf{NSPACE}(n^k)$.

Deterministic and nondeterministic time

Theorem 1.29

Let $t : \mathbb{N} \to \mathbb{N}$ be a function with $t(n) \geq n$ for all $n \in \mathbb{N}$. If language L can be decided by an $\mathcal{O}(t(n))$ time (single-tape) NTM, then L can be decided by a $2^{\mathcal{O}(t(n))}$ time (single-tape) DTM.

Space constructible functions

Definition 1.30

Let $f : \mathbb{N} \rightarrow \mathbb{N}$ be a function with $f(n) \geq n$ for all $n \in \mathbb{N}$. Function f is called space constructible if there is a $\mathcal{O}(f(n))$ space DTM that on input 1^n (i.e. n 1's) computes the binary representation of $f(n)$.

Examples

- $f(n) = n$ is space constructible.
- $f(n) = n^2$ is space constructible.
- $f(n) = 2^n$ is space constructible.

Remark

We will later generalize space constructibility to functions that grow slower than linear.

Deterministic and nondeterministic space

Theorem 1.31 (Savitch's theorem)

Let $s : \mathbb{N} \to \mathbb{N}$ *be a space constructible function with* $s(n) \geq n$ *for all* $n \in \mathbb{N}$, *then*

$$\mathbf{NSPACE}(s(n)) \subseteq \mathbf{DSPACE}(s(n)^2).$$

Yieldability problem

Given Two configurations c_1, c_2 of NTM N and time bound $t \in \mathbb{N}$.

Test Whether the NTM N can get from c_1 to c_2 within t steps.

Procedure CANYIELD

$T =$ "On input configurations c_1, c_2 and $t \in \mathbb{N}$:

1. If $t = 1$, then test directly whether $c_1 = c_2$ or whether c_1 yields c_2 in one step according to the rules of N. *Accept* if either test succeeds; *reject* if both fail.
2. If $t > 1$, then for each configuration c_m of N on w using space $s(n)$:
3. Run $\textsc{CanYield}(c_1, c_m, \frac{t}{2})$.
4. Run $\textsc{CanYield}(c_m, c_2, \frac{t}{2})$.
5. If steps 3 and 4 both accept, then *accept*.
6. If have not accepted yet, *reject*."

Deterministic $s(n)^2$ space TM

Preliminaries

- $c_{\text{start},w} := q_0 \rhd w$, i.e. start configuration of N on input w
- Modify N so that there is single accepting configuration c_{accept}: when N accepts,
  1. it first clears its tape,
  2. then moves the head to the leftmost cell.
- d chosen such that N has at most $2^{d \cdot s(n)}$ configurations using $s(n)$ tape cells.

M, on input w:

1. Output the result of $\textsc{CanYield}(c_{\text{start},w}, c_{\text{accept}}, 2^{d \cdot s(n)})$.

PSPACE and **NPSPACE**

Corollary 1.32

PSPACE $=$ **NPSPACE**.

Configuration graphs

Definition 1.33

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{accept}}, q_{\mathrm{reject}})$ *be Turing machine and* $w \in \Sigma^*$. *The configuration graph of* M *on input* w *is the graph* $G = (V, E)$, *where*

1. V *consists of the configurations of* M *on its computation branches on input* w,
2. *for all* $c_1, c_2 \in V$ *the tuple* (c_1, c_2) *is in* E *if* c_1 *yields* c_2.

Remark

If M has space complexity $s(n)$, then the configuration graph of M on input w has $2^{\mathcal{O}(s(|w|))}$ vertices.

Procedure $\textsc{CanYield}$

$T =$ "On input configurations c_1, c_2 and $t \in \mathbb{N}$:

1. If $t = 1$, then test directly whether $c_1 = c_2$ or whether c_1 yields c_2 in one step according to the rules of N. *Accept* if either test succeeds; *reject* if both fail.
2. If $t > 1$, then for each configuration c_m of N on w using space $s(n)$:
3. Run $\textsc{CanYield}(c_1, c_m, \frac{t}{2})$.
4. Run $\textsc{CanYield}(c_m, c_2, \frac{t}{2})$.
5. If steps 3 and 4 both accept, then *accept*.
6. If have not accepted yet, *reject*."

Remark

For a NTM $N = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ and $w \in \Sigma^*$, the procedure $\textsc{CanYield}$ decides whether there is a (directed) path from c_1 to c_2 of length at most t.