Computability and Complexity

Lecture 8

Big-O and small-o notation
Time complexity class TIME($t(n)$)
The class P and examples

given by Jiri Srba

Complexity Theory

Question

Assume that a problem (language) is decidable. Does it mean that we can realistically solve it?

Complexity Theory

Question

Assume that a problem (language) is decidable. Does it mean that we can realistically solve it?

Answer

NO, not always. It can require too much resources (time, memory).

Complexity Theory aims at making general conclusions about the time and space requirements of decidable problems (languages).

Complexity Theory

Question

Assume that a problem (language) is decidable. Does it mean that we can realistically solve it?

Answer

NO, not always. It can require too much resources (time, memory).

Complexity Theory aims at making general conclusions about the time and space requirements of decidable problems (languages).

Our computational model is a Turing machine. How do we measure the time and memory?

- Time $=$ the number of computation steps.
- Memory $=$ the number of used tape cells.

From now on, we consider only deciders and decidable languages!

Big-O Notation

Definition (Asymptotic Upper Bound)

Let $f, g : \mathbb{N} \to \mathbb{R}^{>0}$ be functions. We write $f(n) = O(g(n))$ iff

- there are positive integers c and n_0 such that
- $f(n) \leq c \cdot g(n)$ for every $n \geq n_0$.

Big-O Notation

Definition (Asymptotic Upper Bound)

Let $f, g : \mathbb{N} \to \mathbb{R}^{>0}$ be functions. We write $f(n) = O(g(n))$ iff

- there are positive integers c and n_0 such that
- $f(n) \leq c \cdot g(n)$ for every $n \geq n_0$.

Examples:

- $2n^2 + 5 = O(n^2)$
- $7n^5 + 88n^4 + n^2 + 104 = O(n^5)$
- $\log_2(n^8) = O(\log n)$

Big-O Notation

Definition (Asymptotic Upper Bound)

Let $f, g : \mathbb{N} \to \mathbb{R}^{>0}$ be functions. We write $f(n) = O(g(n))$ iff

- there are positive integers c and n_0 such that
- $f(n) \leq c \cdot g(n)$ for every $n \geq n_0$.

Examples:

- $2n^2 + 5 = O(n^2)$
- $7n^5 + 88n^4 + n^2 + 104 = O(n^5)$
- $\log_2(n^8) = O(\log n)$

Other notation:

- $2^{O(n)}$ means an upper bound $O(2^{cn})$ for some constant c
- $n^{O(1)}$ is a polynomial upper bound $O(n^c)$ for some constant c

Small-o Notation

Definition (Strict Asymptotic Upper Bound)

Let $f, g : \mathbb{N} \rightarrow \mathbb{R}^{>0}$ be functions. We write $f(n) = o(g(n))$ iff

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \ .$$

Small-o Notation

Definition (Strict Asymptotic Upper Bound)

Let $f, g : \mathbb{N} \to \mathbb{R}^{>0}$ be functions. We write $f(n) = o(g(n))$ iff

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \ .$$

Examples:

- $n^2 = o(n^3)$
- $\sqrt{n} = o(n)$
- $n \log n = o(n^2)$
- $n^{100} = o(2^n)$

Small-o Notation

Definition (Strict Asymptotic Upper Bound)

Let $f, g : \mathbb{N} \to \mathbb{R}^{>0}$ be functions. We write $f(n) = o(g(n))$ iff

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \ .$$

Examples:

- $n^2 = o(n^3)$
- $\sqrt{n} = o(n)$
- $n \log n = o(n^2)$
- $n^{100} = o(2^n)$

Intuition

- $f(n) = O(g(n))$ means "asymptotically $f \leq g$"
- $f(n) = o(g(n))$ means "asymptotically $f < g$"

Motivation Example

$M =$ " On input w:

1. Scan the tape and reject if w is not of the form O^*1^*.
2. Scan the tape and cross one 0 and one 1.
3. If all O's crossed and some 1 left, or
   all 1's crossed and some O left, then reject.
4. If all symbols crossed then accept else goto step 2."

Clearly, $L(M) = \{0^k 1^k \mid k \geq 0\}$.

Motivation Example

$M =$ " On input w:

1. Scan the tape and reject if w is not of the form O^*1^*.
2. Scan the tape and cross one 0 and one 1.
3. If all O's crossed and some 1 left, or
   all 1's crossed and some O left, then reject.
4. If all symbols crossed then accept else goto step 2."

Clearly, $L(M) = \{0^k1^k \mid k \geq 0\}$.

Question

What is the running time of M?

Motivation Example

$M =$ " On input w:

1. Scan the tape and reject if w is not of the form O^*1^*.
2. Scan the tape and cross one 0 and one 1.
3. If all O's crossed and some 1 left, or
   all 1's crossed and some O left, then reject.
4. If all symbols crossed then accept else goto step 2."

Clearly, $L(M) = \{0^k 1^k \mid k \geq 0\}$.

Question

What is the running time of M?

Answer (Worst Case Analysis)

The machine M will on w perform $O(|w|^2)$ steps in the worst case.

Running Time of a Turing Machine

Definition (Running Time of a TM)

Let M be a deterministic 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 performs on any input of length n.

Running Time of a Turing Machine

Definition (Running Time of a TM)

Let M be a deterministic 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 performs on any input of length n.

Convention

From now on n always denotes the length of the input string.

Example: Running time of M from the previous slide is $O(n^2)$.

Growth Rates of Different Running Times

Assume that your computer performs 1 billion steps per second.
The table shows the CPU time when computing on input of size n.

n	$f(n) = n$	$f(n) = n^2$	$f(n) = n^3$	$f(n) = 2^n$
10	0.01 microsec	0.1 microsec	1 microsec	1 microsec
20	0.02 microsec	0.4 microsec	8 microsec	1 millisec
50	0.05 microsec	2.5 microsec	125 microsec	13 days
100	0.1 microsec	10 microsec	1 millisec	4×10^{13} years

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

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

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

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

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

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

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

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

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

Fact: $\text{TIME}(n) \subset \text{TIME}(n^2) \subset \text{TIME}(n^3) \subset \ldots \subset \text{TIME}(2^n) \subseteq \ldots$

Examples:

- $\{0^k 1^k \mid k \geq 0\} \in \text{TIME}(n^2)$

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

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

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

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

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

Fact: $\text{TIME}(n) \subset \text{TIME}(n^2) \subset \text{TIME}(n^3) \subset \ldots \subset \text{TIME}(2^n) \subseteq \ldots$

Examples:

- $\{0^k 1^k \mid k \geq 0\} \in \text{TIME}(n^2)$
- $\{0^k 1^k \mid k \geq 0\} \in \text{TIME}(n \log n)$ ... see the next slide

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

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

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

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

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

Fact: $\text{TIME}(n) \subset \text{TIME}(n^2) \subset \text{TIME}(n^3) \subset \ldots \subset \text{TIME}(2^n) \subseteq \ldots$

Examples:

- $\{0^k 1^k \mid k \geq 0\} \in \text{TIME}(n^2)$
- $\{0^k 1^k \mid k \geq 0\} \in \text{TIME}(n \log n)$ ... see the next slide
- $\{w\#w \mid w \in \{0,1\}^*\} \in \text{TIME}(n^2)$

Example: $\{0^k1^k \mid k \geq 0\} \in \mathrm{TIME}(n \log n)$

$M =$ "On input w:

1. Scan the tape and reject if w is not of the form O^*1^*.
2. Repeat as long as some 0's and some 1's are on the tape:
3. If there is an odd number of uncrossed symbols, reject.
4. Else, cross off every second 0 and every second 1.
5. If everything is crossed then accept, else reject."

Example: $\{0^k1^k \mid k \geq 0\} \in \mathrm{TIME}(n \log n)$

$M =$ "On input w:

1. Scan the tape and reject if w is not of the form O^*1^*.
2. Repeat as long as some 0's and some 1's are on the tape:
3. If there is an odd number of uncrossed symbols, reject.
4. Else, cross off every second 0 and every second 1.
5. If everything is crossed then accept, else reject."

Observe:

- $L(M) = \{0^k1^k \mid k \geq 0\}$
- M runs in time $O(n \log n)$

Conclusion: $\{0^k1^k \mid k \geq 0\} \in \mathrm{TIME}(n \log n)$

Example: $\{0^k1^k \mid k \geq 0\} \in \mathrm{TIME}(n \log n)$

$M =$ "On input w:

1. Scan the tape and reject if w is not of the form O^*1^*.
2. Repeat as long as some 0's and some 1's are on the tape:
3. If there is an odd number of uncrossed symbols, reject.
4. Else, cross off every second 0 and every second 1.
5. If everything is crossed then accept, else reject."

Observe:

- $L(M) = \{0^k1^k \mid k \geq 0\}$
- M runs in time $O(n \log n)$

Conclusion: $\{0^k1^k \mid k \geq 0\} \in \mathrm{TIME}(n \log n)$

Fact:

Language $\{0^k1^k \mid k \geq 0\}$ is decidable on 2-tape TM in time $O(n)$.

Relationship between k-Tape and Single-Tape TMs

Theorem

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

Every k-tape TM running in time $t(n)$ has an equivalent single-tape TM running in time $O(t^2(n))$.

Proof: Use the simulation of a multi-tape TM M by a single-tape TM M' (from Lecture 2) and analyze the running time of M'. $\square$

Relationship between k-Tape and Single-Tape TMs

Theorem

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

Every k-tape TM running in time $t(n)$ has an equivalent single-tape TM running in time $O(t^2(n))$.

Proof: Use the simulation of a multi-tape TM M by a single-tape TM M' (from Lecture 2) and analyze the running time of M'. $\square$

Note:

- The single-tape TM is only polynomially slower than the multi-tape TM.
- If the multi-tape TM runs in polynomial time, the single-tape TM will also run in polynomial time.

Polynomial running time is defined by $O(n^k)$ for some constant k.

Polynomial Time Equivalence of Deterministic Models

Church-Turing Thesis

Turing machines capture exactly the informal notion of algorithm.

- But, the presented simulation of a 2-tape TM on a single-tape TM needs quadratically more time.
- In fact, 2-tape TMs cannot be simulated by single-tape TMs with the same time complexity.

Polynomial Time Equivalence of Deterministic Models

Church-Turing Thesis

Turing machines capture exactly the informal notion of algorithm.

- But, the presented simulation of a 2-tape TM on a single-tape TM needs quadratically more time.
- In fact, 2-tape TMs cannot be simulated by single-tape TMs with the same time complexity.

Polynomial Time Equivalence of Deterministic Models

All reasonable deterministic models of computation are polynomial time equivalent, i.e., they can simulate each other with only polynomial increase in the respective running times.

The Class P

Definition

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

$$\mathsf{P} \stackrel{\text{def}}{=} \bigcup_{k \geq 0} \mathsf{TIME}(n^k) \ .$$

The Class P

Definition

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

$$\mathsf{P} \stackrel{\text{def}}{=} \bigcup_{k \geq 0} \mathsf{TIME}(n^k) .$$

Discussion:

- The class P is robust (the class remains the same even if we choose some other deterministic model of computation).
- The class P roughly corresponds to the class of problems realistically solvable on a computer.

Examples of Languages in the Class P

$PATH \stackrel{\text{def}}{=} \{\langle G, s, t\rangle \mid G \text{ is a graph with a path from node } s \text{ to } t\}$

- Depth-first search algorithm in pseudo-code takes quadratic time $O(n^2)$ to decide $PATH$ (note that n is the total size of the graph, not the number of nodes).
- Because the class P is robust, it is decidable in polynomial time also on deterministic single-tape TM.
- Hence $PATH \in \mathrm{P}$.

Examples of Languages in the Class P

$$PATH \stackrel{\text{def}}{=} \{\langle G, s, t\rangle \mid \; G \text{ is a graph with a path from node } s \text{ to } t \;\}$$

- Depth-first search algorithm in pseudo-code takes quadratic time $O(n^2)$ to decide $PATH$ (note that n is the total size of the graph, not the number of nodes).
- Because the class P is robust, it is decidable in polynomial time also on deterministic single-tape TM.
- Hence $PATH \in$ P.

$$EVEN \stackrel{\text{def}}{=} \{\langle k\rangle \mid \; k \text{ is an even number } \}$$

- $EVEN \in$ P

Examples of Languages in the Class P

$$PATH \stackrel{\text{def}}{=} \{\langle G, s, t\rangle \mid \; G \text{ is a graph with a path from node } s \text{ to } t\ \}$$

- Depth-first search algorithm in pseudo-code takes quadratic time $O(n^2)$ to decide *PATH* (note that n is the total size of the graph, not the number of nodes).
- Because the class P is robust, it is decidable in polynomial time also on deterministic single-tape TM.
- Hence $PATH \in \mathrm{P}$.

$$EVEN \stackrel{\text{def}}{=} \{\langle k\rangle \mid \; k \text{ is an even number }\}$$

- $EVEN \in \mathrm{P}$

Agreement:

When encoding numbers, we use a binary encoding (or any other encoding with base at least 2) but not the unary encoding.

Conclusion

Focus of Algorithms and Data Structures:

- study of concrete algorithms and the analysis of their precise complexity
- running times of algorithms $O(n^3)$ vs. $O(n^2)$ make a difference
- running times depend on the chosen model of computation (usually a pseudo-code)

Conclusion

Focus of Algorithms and Data Structures:

- study of concrete algorithms and the analysis of their precise complexity
- running times of algorithms $O(n^3)$ vs. $O(n^2)$ make a difference
- running times depend on the chosen model of computation (usually a pseudo-code)

Focus of Complexity Theory:

- study of problems (languages) rather than concrete algorithms
- the difference between $O(n^3)$ and $O(n^2)$ is not crucial (it depends on the chosen model anyway)
- the conclusions should be general enough to be valid for any choice of a (deterministic) computational model

Exam Questions

- Big-O and small-o notation.
- Running time (worst-case time complexity) of a TM.
- Complexity classes TIME($t(n)$), P, and polynomial time equivalence of deterministic models.
- Simulation of multi-tape TMs by single-tape TMs with quadratic increase in running time.