

Advanced Topics in Theoretical Computer Science

Part 5: Complexity (Part 1)

18.01.2023

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Contents

- Recall: Turing machines and Turing computability
- Register machines (LOOP, WHILE, GOTO)
- Recursive functions
- The Church-Turing Thesis
- Computability and (Un-)decidability
- **Complexity**

Motivation (The pragmatical view)

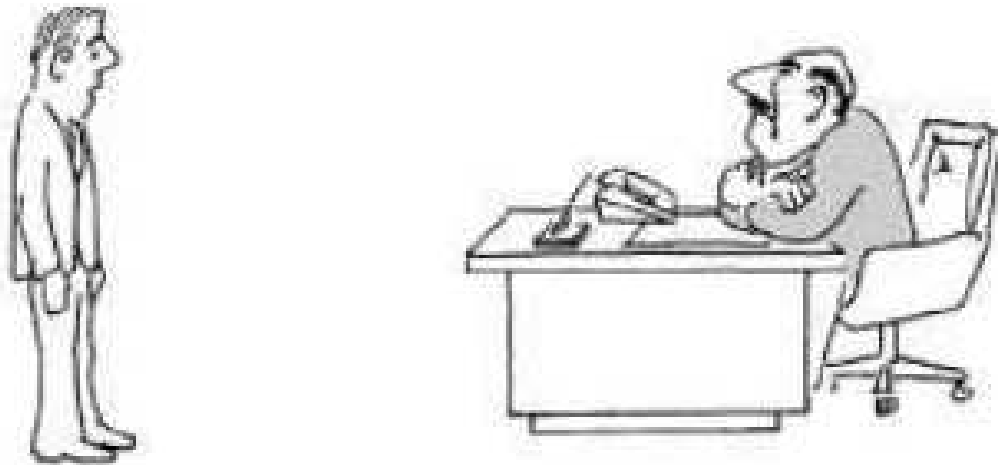
Assume you are employed as software designer.

One day, your boss calls you into his office and tells you that the company is about to enter a very competitive market, for which it is essential to know how to solve (efficiently) problem X .

Your charge is to find an efficient algorithm for solving this problem.

Motivation (The pragmatical view)

What you certainly don't want:



"I can't find an efficient algorithm. I guess I'm just too dumb"

(Garey, Johnson, 1979)

Motivation (The pragmatical view)

Much better:



"I can't find an efficient algorithm, because no such algorithm is possible!"

(Garey, Johnson, 1979)

Motivation

In this lecture we showed how to prove that certain problems do not have a (terminating) algorithmic solution

↳ undecidability results

In the next weeks we will show that even decidable problems are “intractable” in the sense that they have a high complexity.

Unfortunately, proving undecidability or inherent intractability can be just as hard as finding efficient algorithms.

The pragmatical view

However, we will see that you can often answer:



"I can't find an efficient algorithm, but neither can all these famous people."

(Garey, Johnson, 1979)

Motivation

Goals:

- Define formally time and space complexity
- Define a family of “complexity classes”: P, NP, PSPACE, ...
- Study the links between complexity classes
- Learn how to show that a problem is in a certain complexity class
 - Reductions to problems known to be in the complexity class
- Closure of complexity classes

We will give examples of problems from various areas and study their complexity.

Complexity

- Recall:
 - Big O notation
 - The structure of PSPACE
 - Complete problems; hard problems
 - Examples

Big O notation

Definition. Let $h, f : \mathbb{N} \rightarrow \mathbb{R}$ functions.

The function h is in the class $O(f)$ iff there exists $c \in \mathbb{R}$, $c > 0$ and there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ $|h(n)| \leq c|f(n)|$.

Notation: $f \in O(h)$, sometimes also $f(n) \in O(h(n))$;
by abuse of notation denoted also by $f = O(h)$

Examples:

$$5n + 4 \in O(n)$$

$$5n + n^2 \notin O(n)$$

$$\binom{n}{2} = \frac{n(n-1)}{2} \in O(n^2)$$

Let p be a polynomial of degree m . Then $p(n) \in O(n^m)$

Big O notation

Computation rules for O

- $f \in O(f)$
- $c \cdot O(f) = O(f)$
- $O(O(f)) = O(f)$
- $O(f) \cdot O(g) = O(f \cdot g)$
- $O(f \cdot g) = |f|O(g)$
- If $|f| \leq |g|$ then $O(f) \subseteq O(g)$

Lemma. The following hold:

- $\forall d > 0, n^{d+1} \notin O(n^d)$
- $\forall r > 1 \forall d (r^n \notin O(n^d) \text{ and } n^d \in O(r^n))$

Complexity

Types of complexity

- Time complexity
- Space complexity

DTIME and NTIME

Basic model: k -DTM or k -NTM M (one tape for the input)

If M makes for every input word of length n at most $T(n)$ steps, then M is $T(n)$ -time bounded.

In this case, the language accepted by M has time complexity $T(n)$; (more precisely $\max(n + 1, T(n))$).

Definition ($NTIME(T(n)), DTIME(T(n))$)

- $DTIME(T(n))$ class of all languages accepted by $T(n)$ -time bounded DTMs.
- $NTIME(T(n))$ class of all languages accepted by $T(n)$ -time bounded NTMs.

DSPACE and NSPACE

Basic model: k -DTM or k -NTM M with special tape for the input (is read-only) + k storage tapes (offline DTM) \mapsto needed if $S(n)$ sublinear

If M needs, for every input word of length n , at most $S(n)$ cells on the storage tapes then M is $S(n)$ -space bounded.

The language accepted by M has space complexity $S(n)$;
(more precisely $\max(1, S(n))$).

Definition ($NSPACE(S(n)), DSPACE(S(n))$)

- $DSPACE(S(n))$ class of all languages accepted by $S(n)$ -space bounded DTMs.
- $NSPACE(S(n))$ class of all languages accepted by $S(n)$ -space bounded NTMs.

Example

To which time/space complexity does the following language belong:

$$L_{\text{mirror}} = \{wcw^R \mid w \in \{0, 1\}^*\}$$

Example

To which time/space complexity does the following language belong:

$$L_{\text{mirror}} = \{wcw^R \mid w \in \{0, 1\}^*\}$$

Time: $DTIME(n + 1)$: copy input to the right of c in reverse order. When c is found, the rest is compared with the copy of w on the tape.

Space: $DSPACE(n)$: previous DTM

Example

To which time/space complexity does the following language belong:

$$L_{\text{mirror}} = \{wcw^R \mid w \in \{0, 1\}^*\}$$

Time: $DTIME(n + 1)$: copy input to the right of c in reverse order. When c is found, the rest is compared with the copy of w on the tape.

Space: $DSPACE(n)$: previous DTM

Even better $DSPACE(\log(n))$: use two tapes as binary counters.

1. the input is checked for the occurrence of just one c and an equal number of symbols to the left and right of c . This needs only constant space, resp. it can be done with a number of states (and thus needs no space at all).
2. we check the right and left part symbol by symbol: to do this we just have to keep in mind the two positions to be checked (for equality) (and they are coded on the two tapes).

Remember: definition of $DSPACE$ does not count the space used on the input tape.

Questions

Time: Is any language in $DTIME(f(n))$ decided by some DTM?

Space: Is any language in $DSPACE(f(n))$ decided by some DTM?

The functions f are usually very simple functions; in particular they are all computable.

We will consider e.g. powers $f(n) = n^k$.

Questions

Time: Is any language in $DTIME(f(n))$ decided by some DTM?

Space: Is any language in $DSPACE(f(n))$ decided by some DTM?

The functions f are usually are very simple functions; in particular they are all computable.

We will consider e.g. powers $f(n) = n^k$.

Time/Space: What about $NTIME(f(n))$, $NSPACE(f(n))$

Time vs. Space: What are the links between $DTIME(f(n))$, $DSPACE(f(n))$, $NTIME(f(n))$, $NSPACE(f(n))$

Questions

- Time bounded** What does it mean that a DTM makes at most n steps?
Strictly speaking, after n steps it should halt or hang.
- Halt?** Input is accepted
- Hang?** DTM on band which is infinite on both sides **cannot hang!**

Questions

Time bounded What does it mean that a DTM makes at most n steps?
Strictly speaking, after n steps it should halt or hang.

Halt? Input is accepted

Hang? DTM on band which is infinite on both sides **cannot hang!**

Stop after n steps

Stop: We understand the following under M makes at most n steps:

- It halts (and accepts the input) within n steps
- It hangs (and does not accept the input) within n steps
- It halts after n steps, but not in halting mode, so it does not accept the input.

Answers

Answers (Informally)

Time: Every language from $DTIME(f(n))$ is decidable:
for an input of length n we wait as long as the value $f(n)$.
If until then no answer “YES” then the answer is “NO”.

Space: Every language from $DSPACE(f(n))$ is decidable:
There are only finitely many configurations. We write all configurations.
If the TM does not halt then there is a loop. This can be detected.

Answers

Answers (Informally)

NTM vs. DTM: Clearly, $DTIME(f(n)) \subseteq NTIME(f(n))$ and
 $DSPACE(f(n)) \subseteq NSPACE(f(n))$

If we try to simulate an NTM with a DTM we may need exponentially more time. Therefore:

$$NTIME(f(n)) \subseteq DTIME(2^{h(n)}) \text{ where } h \in O(f).$$

For the space complexity we can show that:

$$NSPACE(f(n)) \subseteq DSPACE(f^2(n))$$

Time vs. Space: Clearly, $DTIME(f(n)) \subseteq DSPACE(f(n))$ and
 $NTIME(f(n)) \subseteq NSPACE(f(n))$
 $DSPACE(f(n)), NSPACE(f(n))$ are much larger.

Question

What about constant factors?

Constant factors are ignored. Only the rate of growth of a function in complexity classes is important.

Theorem.

For every $c \in \mathbb{R}^+$ and every storage function $S(n)$ the following hold:

- $DSPACE(S(n)) = DSPACE(cS(n))$
- $NSPACE(S(n)) = NSPACE(cS(n))$

Proof (Idea). One direction is trivial. The other direction can be proved by representing a fixed amount $r > \frac{2}{c}$ of neighboring cells on the tape as a new symbol.

The states of the new machine simulate the movements of the read/write head as transitions. For r -cells of the old machine we use only two: in the most unfavourable case when we go from one block to another.

Time acceleration

Theorem For every $c \in \mathbb{R}^+$ and every time function $T(n)$ with $\lim_{n \rightarrow \infty} \frac{T(n)}{n} = \infty$ the following hold:

- $DTIME(T(n)) = DTIME(cT(n))$
- $NTIME(T(n)) = NTIME(cT(n))$

Proof (Idea). One direction is trivial. The other direction can be proved by representing a fixed amount $r > \frac{4}{c}$ of neighboring cells on the tape as a new symbol.

The states of the new machine simulate also now which symbol and which position the read/write head of the initial machine has. When the machine is simulated the new machine needs to make 4 steps instead of r : 2 in order to write on the new fields and 2 in order to move the head on the new field and then back on the old (in the worst case).

Big O notation

Theorem: Let T be a time function with $\lim_{n \rightarrow \infty} \frac{T(n)}{n} = \infty$ and S a storage function.

(a) If $f(n) \in O(T(n))$ then $DTIME(f(n)) \subseteq DTIME(T(n))$.

(b) If $g(n) \in O(S(n))$ then $DSPACE(g(n)) \subseteq DSPACE(S(n))$.