Advanced Algorithms CS3172, 02/03

Time: **Monday and Friday, 10-11**.

www: **Do check the website regularly.**

Lecture Course comes in 2 parts:
Part I (Dix) Introduction to Complexity Classes,
Part II (Rydeheard) Specific Algorithms.

Organisation:

Part I (Dix): 8 lectures, one week free (3/7 March), one week to discuss the homework (10/14 March).

Part II (Rydeheard): 8 lectures, one week free (5/9 May), one week to discuss the homework (12/16 May).

Exam:

Overview

1. **Turing Machines**
2. **Complexity Classes**
3. **Hierarchies, Complete Problems**

2 Complexity Classes

2.1 Time/Space Complexity

2.2 Speed up

2.3 Relations between Time/Space

2.1 Time/Space Complexity

Definition 2.1 (NTIME$(T(n))$, DTIME$(T(n))$)
We consider as base model a multitape TM M with k two-way infinite tapes, one of which contains the input. If for every word of length n as input, M makes at most $T(n)$ moves, then M is called $T(n)$ time bounded.

The language accepted by M is said to be of time complexity $T(n)$ (actually we mean $\max(n+1, \lceil T(n) \rceil)$).

- **DTIME**$(T(n))$ is the class of languages accepted by $T(n)$ time bounded deterministic DTMs.
- **NTIME**$(T(n))$ is the class of languages accepted by $T(n)$ time bounded nondeterministic NDTMs.

Definition 2.2 (NSPACE (S(n)), DSPACE$(S(n))$)

We consider as base model an offline TM $\mathcal{M}$ with k one-way infinite tapes and a special input tape. If for every word of length n as input, $\mathcal{M}$ scans at most $S(n)$ cells on the storage tapes, then $\mathcal{M}$ is called $S(n)$ space bounded.

The language accepted by $\mathcal{M}$ is said to be of space complexity $S(n)$ (actually we mean $\max(1, \lceil S(n) \rceil)$).

- **DSPACE**$(S(n))$ is the class of languages accepted by $S(n)$ space bounded deterministic DTMs.
- **NSPACE** $(S(n))$ is the class of languages accepted by $S(n)$ space bounded nondeterministic NDTMs.

Why offline TM?

(tape bounds of less than linear growth)

To which time/space complexity class belongs

$$\mathcal{L}_{\text{mirror}} := \{wcw^R : \; w \in (0+1)^*\},$$

i.e. the set of words that can be mirrored on the middle letter c?

Time: $\mathbf{DTIME}(n+1)$. Just copy the input to the right of c in reverse order on another tape. Once a c is found, just compare the remaining part (the w) with the copy of w on the tape.

Space: $\mathbf{DSPACE}(\lg n)$. The machine just described gives us a bound of $\mathbf{DSPACE}(n)$. But we can do better. We use two tapes as binary counters. Firstly 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). Secondly 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).

A few words on terminology.

computable: A function is computable, if, by definition, there is a DTM computing it (given the input n, the DTM computes $f(n)$). The function can be partial or not. We also say the function is **partial recursive** (see slide 11).

accepted: A language is accepted (or recognised), if there is a DTM accepting it (given an input w, the DTM stops in an accepting state if and only if w is in the language).

decided: We say a DTM decides a language, if there is a DTM that accepts it and it always terminates. The language is then called decidable.

decidable: We say a problem is **decidable**, if there is a DTM that decides it. A problem can always be put in the form “Is $w \in \mathcal{L}$” for an appropriate language $\mathcal{L}$.

Recall that we also call a function **recursive**, if it is partial recursive and total.

If a language is decidable, then its complement is as well. This is not true for acceptance.

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

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

Time/Space: Same questions about **NTIME**$(,)$ **NSPACE**.

Homework 2 (Acceptability/Decidability)

Comment on the following statements.

1. The traditional algorithm for checking whether a number is prime, is decidable.
2. Any finite language is accepted by a DTM.
3. Any infinite language is accepted by a NDTM.
4. Any finite language is decided by a DTM.
5. There are algorithms that are undecidable.
6. If $\mathcal{L}$ is decidable, then its complement is decidable as well.
7. There is at least one DTM that is decidable.
8. There is at least one NDTM that is undecidable.

9. The following function is decidable:

$$f : \mathbb{N} \longrightarrow \mathbb{N}, n \mapsto \begin{cases} 1, & \text{if cricket is a sport for stupid people;} \\ 0, & \text{otherwise.} \end{cases}$$

2.2 Speed up

The aim of this section is to illustrate that only the functional rate of growth of a function matters in a complexity class: constant factors have to be ignored.

Theorem 2.1 (Tape compression)
For any $c > 0$ and space function $S(n)$:

$$\mathbf{DSPACE}(S(n)) = \mathbf{DSPACE}(cS(n))$$
$$\mathbf{NSPACE}(S(n)) = \mathbf{NSPACE}(cS(n))$$

Note that one direction is trivial. The proof for the other is by representing a fixed number r ($> \frac{2}{c}$) of adjacent tape cells by a new symbol. The states of the new machine keep track which of the many cells represented is actually scanned during simulation.

Theorem 2.2 (Time speed up)

For any $c > 0$ and time function $T(n)$ with $\inf_{n\to\infty} \frac{T(n)}{n} = \infty$:

$$\mathbf{DTIME}(T(n)) = \mathbf{DTIME}(cT(n))$$

$$\mathbf{NTIME}(T(n)) = \mathbf{NTIME}(cT(n))$$

Again one direction is trivial. The proof for the other is also by representing a fixed number r ($> \frac{16}{c}$) of adjacent tape cells by a new symbol (the states of the new machine keep track which of the many cells represented is actually scanned (when simulating the old machine)).

When simulating the old machine, the new one only needs to make 8 moves instead of r: 4 to check the immediate neighbours and another 4 to modify them.

What happens if we reduce the number of tapes? Let us consider again $\mathcal{L}_{\text{mirror}}$ from Slide 50. The linear complexity does no more hold if there is only one tape available. However, the following holds.

Theorem 2.3 (Reduction of tapes (1))

- If $\mathcal{L} \in \mathbf{DTIME}(T(n))$, then $\mathcal{L}$ is accepted in time $T^2(n)$ by a one-tape DTM.
- If $\mathcal{L} \in \mathbf{NTIME}(T(n))$, then $\mathcal{L}$ is accepted in time $T^2(n)$ by a one-tape NDTM.

The proof is simple. Remember that we need $6T^2(n)$ steps to simulate the k-tape DTM using a 1-tape DTM (see slide 34). Now we speed it up by $\frac{1}{\sqrt{6}}$.

The last theorem also holds for space bounded functions:

Theorem 2.4 (Reduction of tapes (2))

- If $\mathcal{L} \in \mathbf{DSPACE}(S(n))$, then $\mathcal{L}$ is accepted in space $S(n)$ by a one-tape DTM.
- If $\mathcal{L} \in \mathbf{NSPACE}(S(n))$, then $\mathcal{L}$ is accepted in space $S(n)$ by a one-tape NDTM.

This proof is as simple as the last one. Note that in simulating a k-tape TM with a 1-tape TM we need the same number of storage cells. So we do not even need to speed up to get our result.

I should have reached this point after the fifth lecture.

2.3 Relations between Time/Space

Theorem 2.5 (Time versus Space)

- $\mathbf{DTIME}(f(n)) \subseteq \mathbf{DSPACE}(f(n))$.
- If $f(n) \geq \lg n$, and $\mathcal{L} \in \mathbf{DSPACE}(f(n))$, then there is a $c > 0$ s.t. $\mathcal{L} \in \mathbf{DTIME}(c^{f(n)})$.
- If $\mathcal{L} \in \mathbf{NTIME}(f(n))$, then there is a $c > 0$ s.t. $\mathcal{L} \in \mathbf{DTIME}(c^{f(n)})$.

Proof:

- Obvious.
- Suppose we have s states and t tape symbols. By using at most $f(n)$ cells, the number of different IDs on an input of length n is bounded by $s(n+2)(f(n)+1)t^{f(n)}$ (we assume in view of Theorem 2.4 that we are dealing with an offline DTM with just one storage tape). Because of $f(n) \geq \lg n$, there is a c such that for $n \geq 1$: $c^{f(n)} \geq s(n+2)(f(n)+1)t^{f(n)}$. We can construct a 3-tape DTM: one tape is used to count up to $c^{f(n)}$, the other two to simulate the old machine. When no accepting state is reached until the maximal count, it will never accept (the old machine actually loops): we then simply terminate in a non accepting state. If an accepting state is reached, the new machine accepts as well.

- Similar to the last case, but now we have to take into account the number k of tapes as well (and that it is a regular k-tape NDTM). Number of IDs is bounded by ... We construct a multitape DTM to simulate the old NDTM. Our machine first constructs a list L of all accessible IDs (from the initial input): This can be done in time bounded by $\mathrm{length}^2(L)$ (why?) and we have $\mathrm{length}(L) \leq \ldots$ and therefore $\leq c^{f(n)}$. It then checks whether any of the IDs leads to an accepting state or not ...

In the following we want to state some more relations between complexity classes. Unfortunately they do not hold for all time functions $T(n)$ or space functions $S(n)$, but for almost all that do occur naturally. We therefore define the following space of functions.

Definition 2.3 (Well-behaved functions)
We consider the vector space of functions from $\mathbb{N}$ into $\mathbb{N}$ containing $\log_a n, n^k, 2^n, n!$ and closed under multiplication, exponentiation and composition. We call such functions *well-behaved*.

Theorem 2.6 (Det. versus Non-Det. Space)
Let $S(n)$ be well-behaved. Then:

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

Theorem 2.7 (Time/Space Hierarchies)
Let $S_1(n), S_2(n)$ and $T_1(n), T_2(n)$ be well-behaved. We assume further that $S_1(n) < S_2(n)$ and $T_1(n) < T_2(n)$ for all $n > n_0$ for a $n_0 \in \mathbb{N}$.

1. If $\inf_{n\to\infty} \frac{S_1(n)}{S_2(n)} = 0$ then $\textbf{DSPACE}(S_1(n)) \subsetneqq \textbf{DSPACE}(S_2(n))$.
2. If $\inf_{n\to\infty} \frac{T_1(n)\lg T_1(n)}{T_2(n)} = 0$ then $\textbf{DTIME}(T_1(n)) \subsetneqq \textbf{DTIME}(T_2(n))$.

The last theorem implies the following hierarchies:

$$\mathbf{DSPACE}(n) \subsetneqq \mathbf{DSPACE}(n^2) \subsetneqq \ldots \subsetneqq \mathbf{DSPACE}(n^r) \subsetneqq \ldots$$

and

$$\mathbf{DTIME}(n) \subsetneqq \mathbf{DTIME}(n^2) \subsetneqq \ldots \subsetneqq \mathbf{DTIME}(n^r) \subsetneqq \ldots$$

as well as

$$\mathbf{DSPACE}(\log n) \subsetneqq \mathbf{DSPACE}(\log^2 n) \subsetneqq \ldots \subsetneqq \mathbf{DSPACE}(\log^r n) \subsetneqq \ldots$$

and

$$\mathbf{DTIME}(\log n) \subsetneqq \mathbf{DTIME}(\log^2 n) \subsetneqq \ldots \subsetneqq \mathbf{DTIME}(\log^r n) \subsetneqq \ldots$$

What about similar results for **nondet.** space and time?

Theorem 2.8 (Nondeterministic Hierarchies)
Let $S_1(n), S_2(n)$ and $f(n)$ be well-behaved. We assume further that $S_2(n) \geq n$ and $f(n) \geq n$ for all $n \in \mathbb{N}$.

1. $\mathbf{NSPACE}(S_1(n)) \subseteq \mathbf{NSPACE}(S_2(n))$ implies $\mathbf{NSPACE}(S_1(f(n))) \subseteq \mathbf{NSPACE}(S_2(f(n)))$.
2. $\mathbf{NTIME}(S_1(n)) \subseteq \mathbf{NTIME}(S_2(n))$ implies $\mathbf{NTIME}(S_1(f(n))) \subseteq \mathbf{NTIME}(S_2(f(n)))$.

This theorem is applied as follows. Suppose $\mathbf{NSPACE}(n^4) \subseteq \mathbf{NSPACE}(n^3)$. We then apply the theorem for n^3, n^4 and n^5 separately and get: $\mathbf{NSPACE}(n^{20}) \subseteq \mathbf{NSPACE}(n^9)$. By Theorem 2.6 we know that $\mathbf{NSPACE}(n^9) \subseteq \mathbf{DSPACE}(n^{18})$ and by Theorem 2.7 $\mathbf{DSPACE}(n^{18}) \subsetneq \mathbf{DSPACE}(n^{20})$, which is a contradiction.

The last theorem implies the following hierarchies:

$$\mathbf{NSPACE}(n) \subsetneqq \mathbf{NSPACE}(n^2) \subsetneqq \ldots \subsetneqq \mathbf{NSPACE}(n^r) \subsetneqq \ldots$$

and

$$\mathbf{NTIME}(n) \subsetneqq \mathbf{NTIME}(n^2) \subsetneqq \ldots \subsetneqq \mathbf{NTIME}(n^r) \subsetneqq \ldots$$

as well as

$$\mathbf{NSPACE}(\log n) \subsetneqq \mathbf{NSPACE}(\log^2 n) \subsetneqq \ldots \subsetneqq \mathbf{NSPACE}(\log^r n) \subsetneqq \ldots$$

and

$$\mathbf{NTIME}(\log n) \subsetneqq \mathbf{NTIME}(\log^2 n) \subsetneqq \ldots \subsetneqq \mathbf{NTIME}(\log^r n) \subsetneqq \ldots$$

Homework 3 (Simple Relations)

Consider the problem of testing whether a given natural number is prime. Use unary representation. Is there a DTM solving this problem in polynomial time? I.e. is this problem in $\mathbf{DTIME}(n^r)$ for a $r \in \mathbb{N}$? Note that you do not have to actually construct a DTM, it suffices if you can argue convincingly.

Complete the third part of the proof of Theorem 2.5.

Discuss the two notions of decidability versus acceptability of a language wrt. NDTM's and DTM's. What happens, if we take time bounds into consideration? If a language is acceptable in time $T(n)$, it is also decidable in $T(n)$?

What, if any, is the relationship between the following pairs of complexity classes?

1. $\mathbf{DSPACE}(n^2)$ and $\mathbf{DSPACE}(f(n))$ where $f(n) := n$ for odd n and $f(n) := n^3$ for even n.
2. $\mathbf{DTIME}(134^n)$ and $\mathbf{DTIME}((\ln n)^n)$.
3. $\mathbf{DTIME}(2^n)$ and $\mathbf{DTIME}(3^n)$.

4. **NSPACE** (2^n) and **DSPACE**(5^n).
5. **DSPACE**(n) and **DTIME**$(\lceil \lg n \rceil^n)$.

This is what I did until the end of the sixth lecture.

References

Hopcroft, J. and J. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison Wesely.

Lin, S. and T. Rado (1965). Computer studies of turing machine problems. *Journal of the ACM 12*(2), 196–212.

Papadimitriou, C. (1994). *Computational Complexity*. Addison-Wesley.

Rado, T. (1962). On non-computable functions. *The Bell System Technical Journal XLI*(3), 877–884.

Turing, A. M. (1936). On Computable Numbers with an Application to the Entscheidungsproblem. *Proc. London Mathematical Soc., series 2 42*, 230–265. corrections ibid., 43:544–546.