

Computational Complexity Theory - Course Notes

CCT Team

October 18, 2006

2 Lecture 2, 1/9-2006

2.1 Reading Material

No books or notes are used in the course but the following are suggested:

- C.Papadimitriou: Computational Complexity. This book is good for intuition, but not without errors. (It is available at Stakbogladen, price: DKK 670 (excl. student discount)).
- Books by I.Wegener and M.Zimand

2.2 Turing Machine model

The Turing Machine model used in this course is:

- Deterministic
- Finite Control
- Tape: Multi tape machine with a constant number of tapes:
 - Special input tape: Read only, not infinite: It contains the input plus blanks at the ends to signal end-of-tape.
 - Special output tape: Write only, i.e. a “move right only”-tape where the tape head is moved to the right for every write operation.
 - Other tapes are work-tapes
- The input- and tape-alphabets includes $\{0, 1\}$.

Fact: A 1-tape machine can simulate a multitape machine running in time f with only a polynomial overhead (actually $O(f^2)$). Proof: By expanding the alphabet to marked tuples, the execution can be simulated by scanning the tape for marked cells.

This is the same as saying that the multitape TM is a reasonable sequential model of computation, cf. Polynomial Church-Turing Thesis.

2.3 Time

Def. 1 | DTIME (Deterministic Time)

$$\begin{aligned} L \in \mathbf{DTIME}(f), f : \mathbb{N} \rightarrow \mathbb{N}, f(n) \geq n \Leftrightarrow \\ \exists \text{TM } M : \forall x \in \{0, 1\}^* \text{ (except for finitely many inputs) :} \\ \text{if } x \in L, M \text{ accepts } x \text{ in at most } f(|x|) \text{ steps} \\ \text{and if } x \notin L, M \text{ rejects } x \text{ in at most } f(|x|) \text{ steps.} \end{aligned}$$

The restriction $f(n) \geq n$ is made to avoid sub-linear computations, since we would not otherwise be able to read the whole input. The reason for allowing the TM not to accept/reject within the time bound on finitely many inputs, is to allow inputs such as the empty string, for which the TM would not even be able to move to a halting state within the time bound.

Note: **TIME** = **DTIME**.

Def. 2 | P

$$\mathbf{P} = \bigcup_{\text{poly } q} \mathbf{DTIME}(q)$$

2.4 Space

Def. 3 | DSPACE (Deterministic Space)

$$\begin{aligned} L \in \mathbf{DSPACE}(f), f : \mathbb{N} \rightarrow \mathbb{N}, f(n) \geq \log n \Leftrightarrow \\ \exists \text{TM } M : \forall x \in \{0, 1\}^* \text{ (except for finitely many inputs) :} \\ \text{if } x \in L, M \text{ accepts } x \text{ touching at most } f(|x|) \text{ cells on the worktapes} \\ \text{and if } x \notin L, M \text{ rejects } x \text{ touching at most } f(|x|) \text{ cells on the worktapes.} \end{aligned}$$

The restriction $f(n) \geq \log n$ is made to avoid sub-logarithmic computations, since the control itself can hold some information in the position of the tape-heads.

Note: **SPACE** = **DSPACE**.

SPACE-Thesis Any reasonable sequential model of computation can be simulated by a TM using a constant factor of extra space.

2.5 Example: GAP (Graph Accessibility Problem)

Given a directed graph $G = (V, E)$, $V = \{1, \dots, n\}$ (stored as an adjacency matrix) is there a path between 1 and n .

With $n = \# \text{bits in input} = |V|^2$ we have

$$\mathbf{GAP} \in \mathbf{DSPACE}(O(n)) = \bigcup_{c \in \mathbb{N}} \mathbf{DSPACE}(cn)$$

since the naïve DFS/BFS algorithm uses $O(\sqrt{n} \log n)$ space.

For the undirected case, UGAP, we can use a randomized approach, using $O(\log n)$ space, thus establishing $\mathbf{UGAP} \in \mathbf{RSPACE}(O(\log n))$ (**RSPACE** is Randomized Space).

For the directed case the randomized approach does not work, since we might be stuck in a dead-end for which back-tracing is needed.

A recursive algorithm for GAP with better space-bound than $O(\sqrt{n} \log n)$ is:

```

GAP( $u, v, i$ ) : is there a path from  $u$  to  $v$  with length at most  $i$ 
  if ( $i = 0$ ) return  $u = v$ 
  if ( $i = 1$ ) return  $(u, v) \in E$ 
  for ( $r \in V$ ) :
    if (GAP( $u, r, i/2$ )  $\wedge$  GAP( $r, v, i/2$ )) return true
  return false

```

The initial call is GAP(1, n, n), WLOG $n = 2^i$. The space used amounts to the depth of the call stack, $\leq \log n$, times the size of the stack frame, $\leq \log n$, so we have GAP \in **DSPACE**($O(\log^2 n)$) (Savitch '71).

The time complexity of the recursive algorithm is given by the recurrence equation

$$T(n) = 2nT(n/2) \Rightarrow n^{O(\log n)}.$$

Note: GAP \in **DSPACE**($o(\log^2 n)$) is open.

Def. 4 | **SC** (Steve's class)

$$\mathbf{SC} = \mathbf{TISP}(\text{poly}, \text{polylog}).$$

That is, **SC** is the class of languages which can be decided by the same deterministic Turing machine in polynomial time using polylogarithmic ($O(\log^k n)$) space. In addition

$$\mathbf{SC}^i = \mathbf{TISP}(\text{poly}, O(\log^i n)).$$

Note: GAP \in **SC** is open.

Note: **DCFL** (Deterministic Context Free Language) is in **SC** :

$$\forall x \in \mathbf{DCFL} : x \in \mathbf{SC} \Rightarrow \mathbf{DCFL} \subseteq \mathbf{SC}.$$

2.6 PSPACE and EXP

Def. 5 | **EXP**

$$\mathbf{EXP} = \bigcup_{\text{poly } q} \mathbf{DTIME}(2^q)$$

Def. 6 | **PSPACE**

$$\mathbf{PSPACE} = \bigcup_{\text{poly } q} \mathbf{DSPACE}(q)$$

Thm. 7 | **PSPACE** \subseteq **EXP**.

Proof. We consider an ℓ -tape **PSPACE**-machine using $q(n)$ space. WLOG all tapes have $|\Sigma| = k$. The number of configurations for the machine is bounded by the number of cell configurations, $k^{2q(n)\ell}$, the number of head positions, $(2q(n))^\ell$, and the input tape head positions, $n + 2$, thus giving a total of

$$k^{2q(n)\ell} (2q(n))^\ell (n + 2)$$

different configurations. □

2.7 P and EXP

Note: $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ and $\mathbf{EXP} \stackrel{?}{=} \mathbf{NP}$ are open.

Def. 8 | Time Constructable

$f : \mathbb{N} \rightarrow \mathbb{N}$ is a time constructable iff f can be computed in time $O(f)$, i.e. there is a TM which on input 1^n outputs $1^{f(n)}$ in at most $O(f(n))$ steps.

Fact: Every nice function $\geq n$ you know and like is time constructable.

Thm. 9 | For all time constructable functions $f : \mathbb{N} \rightarrow \mathbb{N}$,

$$\exists \text{poly } q : \mathbf{DTIME} \subsetneq \mathbf{DTIME}(q(f)).$$

Note: Actually $q(n) = n^2$ works.

Proof. Define $h : \mathbb{N} \rightarrow \mathbb{N}$ by

$$\begin{aligned} h(0) &= 0 \\ h(1) &= 0 & h(2) &= 1 \\ h(3) &= 0 & h(4) &= 1 & h(5) &= 2 \\ h(6) &= 0 & h(7) &= 1 & h(8) &= 2 & h(9) &= 3 \\ h(10) &= 0 \dots \end{aligned}$$

Define D_f :

$$x \in D_f \Leftrightarrow M_{h(|x|)} \text{ fails to accept } x \text{ in } f(|x|) \text{ steps.}$$

- $D_f \in \mathbf{DTIME}(q(f))$ follows from the Polynomial Church-Turing Thesis.
- $D_f \notin \mathbf{DTIME}(f)$: Assume $D_f \in \mathbf{DTIME}(f)$. Let $M = M_{h(i)}$ be the TM which decides D_f in time f . Take an x , $|x| = i$ then

$$\begin{aligned} x \in D_f &\Leftrightarrow M_{h(|x|)} \text{ fails to accept } x \text{ in } f(|x|) \text{ steps} \\ &\Leftrightarrow M \text{ fails to accept } x \text{ in } f(|x|) \text{ steps} \\ &\Leftrightarrow M \text{ rejects } x \\ &\Leftrightarrow x \notin D_f \end{aligned}$$

□

Note: The use of $h(|x|)$ instead of $|x|$ is motivated by definition of the **DTIME**, which allows M to fail to decide in $f(|x|)$ steps in finitely many inputs of which x could be one. If this was the case the second \Leftrightarrow would not hold.

Note: Thm. 9 gives for example

$$\mathbf{DTIME}(n^3) \subsetneq \mathbf{DTIME}(n^6) \subsetneq \mathbf{DTIME}(n^{12}) \subsetneq \dots$$

and leads to $\mathbf{P} \neq \mathbf{EXP}$.

Prop. 10 | $\mathbf{P} \subseteq \mathbf{DTIME}(2^n)$.

Proof. $\forall \text{poly } p : \mathbf{DTIME}(p) \subseteq \mathbf{DTIME}(2^n)$ since for all large enough $n : p(n) \leq 2^n$. □

Thm. 11 | $\mathbf{P} \neq \mathbf{EXP}$.

Proof. Using Thm. 9 we get

$$\mathbf{P} \subseteq \mathbf{DTIME}(2^n) \subsetneq \mathbf{DTIME}(q(2^n)) \subseteq \mathbf{EXP} \Rightarrow \mathbf{P} \subsetneq \mathbf{EXP} \Rightarrow \mathbf{P} \neq \mathbf{EXP}.$$

□