Complexity Theory

Instructor: Roei Tell

Basic complexity

Course outline

- Modeling
- Computability
- Basic complexity
- P vs NP
- Strong forms of P ≠ NP
- Lemons into lemonade

Complexity classes

Hierarchies

Classifying problems

- <u>Definition:</u>

  TIME[T] = { L : ∃ TM M with runtime O(T(n)) that decides L }

- Which problems can be solved within a certain resource bound?
- Slackness, allowing O(T)

Classifying problems

- Sanity check:

$$\text{TIME}[\,T\,] \subseteq \text{TIME}[\,T'\,]$$

for any $T'(n) > T(n)$.

- Is the containment strict? More resources = more power?

More resources = more power

- <u>Time hierarchy theorem (informal):</u>

  For any "nice" T(n)

  $$\mathrm{TIME}[\,T^c\,] \not\subset \mathrm{TIME}[\,T\,]$$

More resources = more power

- Definition:

  We say that $T(n)$ is time constructible if there's a machine that, on input 1^n, outputs the integer $T(n)$ (in binary) in time $O(T(n))$.

More resources = more power

- "Theorem": // informal, so not really a theorem

  Nice functions are time constructible.

  $T(n) = 5n$, $T(n) = n^{100} \cdot \log(n)$, $T(n) = 2^n$, $T(n) = n^{\sqrt{n}/\log n}$, ...

- Proof idea:

  Arithmetic operations in polynomial time (i.e., polylog(n)).

More resources = more power

- Time hierarchy theorem:

  For any time-constructible $T(n) \geq n$,

  $$\mathrm{TIME}[\,T^c\,] \not\subset \mathrm{TIME}[\,T\,]$$

  where $c > 1$ is a large enough universal constant.

More resources = more power

- Proof idea
  - recap: diagonalization
  - time-bounded version
- Proof

More resources = more power

- Notes:

1. hard function is convoluted; we'll amend that next.
2. tighter hierarchy if we fix a nicer computational model (e.g., two-tape TMs), usually time $T' = \omega(T \cdot \log(T))$ vs time T

A specific hard problem

› <u>Definition (time-bounded universal function):</u>

Let U_T(<M>, x) =

output of M(x) if it halts after T(|x|) steps with one output bit,

or “error” otherwise.

A specific hard problem

- <u>Definition (time-bounded universal function):</u>

  Let U_T(<M>, x) =

  output of M(x) if it halts after T(|x|) steps with one output bit,

  or “error” otherwise.

- U_T is computable in time poly(T) by imitating the construction of a universal machine.

A specific hard problem

- <u>Corollary of time hierarchy:</u>

  There is $\varepsilon > 0$ such that $U_T \notin \mathrm{TIME}[\, T^{\varepsilon} \,]$ for any large enough function T(n).

- <u>Interpretation:</u>

  It's impossible to speed-up the computation of the time-bounded universal function.

A specific hard problem

- Completeness phenomenon:

  If there is a hard problem in TIME[T], then the specific "complete" problem U_T is hard.

- Already encountered this idea, we'll revisit it again later.

Technicalities #1

- In the def of TIME[T] we allowed O(T) rather than only "strictly T"
- <u>Linear speedup theorem:</u>

  For any $c > 1$ and $T(n) = \omega(n^2)$, if there's a TM deciding L in time T then there's a TM deciding L in time T/c.

- Proof idea: Use a bigger alphabet & bigger transition function ("better hardware")

Technicalities #2

- Why hierarchy only for time-constructible T?
- T that isn't time constructible can be intentionally contrived
  - e.g., $T(n) = n$ if $\langle M_n \rangle$ halts and $T(n) = n^2$ otherwise
- <u>Theorem (time gap):</u>

  There exists $T(n)$ such that $TIME[T] = TIME[2^T]$.

Space complexity

- <u>Intuition</u>
  - memory usage when computing $x \mapsto f(x)$
  - memory = additional temporary storage
  - disallow cheating by using input/output for computation

Space complexity

- Machine model
  - read-only input tape
  - read-write worktape
  - unidirectional write-only output tape (“send to printer”)
- Configuration:
  - input tape, worktape, state (exclude output tape)

Space complexity

- <u>Definition:</u>

  The space complexity of M on input x is the maximum length of worktape content in any configuration in the computation of M(x).

- We're not counting the input tape (only "additional storage") or the output tape (no need to store it, "send and forget")

Space complexity

- Definition:

The space complexity of M on n-bit inputs is

$$\max_{x \in \{0,1\}^n} \{ \text{ space complexity of M on input x } \}$$

The space complexity of M is a function

S(n) = space complexity of M on n-bit inputs

Space complexity

- <u>Definition:</u>

  SPACE[S] = { L : ∃ TM M using space O(S(n)) that decides L }

- Some sources don't allow slackness, and instead count all bits in the configuration (state, head locations, log(|alphabet|))

Space complexity

- <u>Theorem (space hierarchy):</u>

  For any space-constructible $S(n) > \log(n)$ and any $S' = \omega(S)$

  $$\mathrm{SPACE}[\,S'\,] \not\subset \mathrm{SPACE}[\,S\,]$$

- "space-constructible": compute $1^n \mapsto S(n)$ in space $O(S)$
- diagonalization proof, analogous to time hierarchy

Space complexity

- <u>Theorem (time is at most exponential in space):</u>

  For any S(n) ,

$$\mathrm{SPACE}[\, S \,] \subseteq \mathrm{TIME}[\, n \cdot 2^{O(S)} \,]$$