Algorithms and Complexity

Spring 2018

Aaram Yun

This page is intentionally left blank

Last time...

- We've studied
  - Turing machines

Today

- Encoding
- Search problems & decision problems
- Church–Turing Thesis
- Uncomputable problems
- Universal Turing machine

Encoding

O : some object

$\langle O \rangle \in \{0,1\}^*$: the encoding of O.

- We *encode* every objects we deal with as bit strings in $\{0, 1\}^*$
  - Number : use binary encoding
  - Pair : $x, y \in \{0,1\}^*$. $\langle x, y \rangle$ = write 'doubled' version of $x \| 10 \| y$

    And tuples : $\langle x, y, z \rangle = \langle x, \langle y, z \rangle \rangle$

    $= \overline{x}\, 10 \langle y, z \rangle$

    $= \overline{x}\, 10\, \overline{y}\, 10\, z$
  - Set : by a tuple
  - Graph : use adjacency matrix representation

More or less, any "reasonable" encoding scheme would do,
So often we don't have to worry too much about the details.

Search problem

- $R \subseteq \{0,1\}^* \times \{0,1\}^*$: a relation of strings
- Let $R(x) := \{y : (x,y) \in R\}$ the set of all y which are related to x by R.
- This R is a search problem
- $f : \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ *solves* R if …

For every $x \in \{0,1\}^*$, if $R(x) = \emptyset$, then $f(x) = \perp$

and if $R(x) \neq \emptyset$, then $f(x) \neq \perp$, and $R(x, f(x))$

that is, $f(x) \in R(x)$.

Search problem

» Examples

Sorting: R_{sort} is a set of (x, y) where
x is an encoding of a list of numbers
and y is an encoding of a list of numbers
with the property that y is a permutation of x
and y is sorted

Equation solving: R_{eq} is a set of (x, y) where
x is an encoding of an int.-coeff. polynomial
and y is an encoding of an integer
such that $x(y) = 0$.

Decision problem

$PRIMES \subseteq \mathbb{N} \subseteq \{0,1\}^*$.

$PRIMES = \{\langle n \rangle \mid n \text{ is a prime number}\}$.

$\langle n \rangle \in PRIMES$ iff n is prime.

- $S \subseteq \{0,1\}^*$
- This S is a decision problem
- $f : \{0,1\}^* \to \{0,1\}$ *solves* S if ...

For every $x \in \{0,1\}^*$, if $x \in S$, then $f(x) = 1$

and if $x \notin S$, then $f(x) = 0$.

If $S \subseteq \{0,1\}^*$, then a characteristic function χ_S of S is defined as

$$\chi_S(x) = 1 \quad \text{if } x \in S$$

$$\chi_S(x) = 0 \quad \text{if } x \notin S$$

A special case

$R_{Euler} = \{(x, y) \mid x$ is a graph and
y is an Eulerian path in $x\}$

$S_{R_{Euler}} = \{x \mid x$ is an Eulerian graph$\}$

» $R \subseteq \{0,1\}^* \times \{0,1\}^*$

» R gives a decision problem as follows:

   » $S_R := \{x : R(x) \neq \emptyset\}$

» If you can solve R, then you can also solve S_R

$R_{Hamilton} = \{(x, y) \mid x$ is a graph and
y is a Hamiltonian path in $x\}$

$S_{R_{Hamilton}} = \{x \mid x$ is a Hamiltonian graph$\}$

Church–Turing thesis

» Computability = Turing computability

Justification?

- From psychology
- From equivalence
- From modern computers

Uncomputable functions

- Not all functions are computable
  - Easy to prove by counting

Of course, proving existence is different from showing a concrete example.

Similar situation had happened for the existence of transcendental numbers: existence easy to prove by counting, but showing a concrete example is harder. (Now we know that e, π are transcendental.)

Halting problem

- We can encode a Turing machine, too.
  - For each Turing machine M, $\langle M \rangle \in \{0,1\}^*$
- The halting function $h : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ is defined as
  - $h(\langle M \rangle, x) = 1$ iff M halts on input x
- The halting function h is not computable

proof) Suppose there exists a turing machine H which computes h

$$H(\langle M, x\rangle) = h(\langle M\rangle, x) \text{ for all TM } M \text{ and } x \in \{0,1\}^*$$

We define a machine D which works as follows:

$D(\langle M\rangle)$ halts and outputs 1 iff $H(\langle M, \langle M\rangle\rangle)$ outputs 0.

loops forever iff $H(\langle M, \langle M\rangle\rangle)$ outputs 1.

$\Rightarrow$ $D(\langle M\rangle)$ halts and outputs 1 iff $h(\langle M\rangle, \langle M\rangle)$ outputs 0.

loops forever iff $h(\langle M\rangle, \langle M\rangle)$ outputs 1.

Then, consider the computation $D(\langle D \rangle)$

Suppose $D(\langle D \rangle)$ halts $\iff H(\langle D, \langle D \rangle \rangle)$ outputs 0

$\iff h(\langle D \rangle, \langle D \rangle) = 0$

$\iff D(\langle D \rangle)$ loops forever

$\big(h(\langle M \rangle, x) = 0$ iff $M(x)$ loops forever. $\big)$

Then what if $D(\langle D \rangle)$ loops forever ?

$\Rightarrow H(\langle D, \langle D \rangle \rangle) = 1.$

$\Rightarrow h(\langle D \rangle, \langle D \rangle) = 1$

$\Rightarrow D(\langle D \rangle)$ halts.

Universal Turing machine

» $\exists\, \mathcal{U}$: Turing machine such that

$$\mathcal{U}(\langle M, x\rangle) = M(x)$$