理论计算机科学导引

TCS - 毛宇尘老师班

shrike505

目录

语言、自动机、正则表达式

> 一言以蔽之，它（TCS）研究的是问题的上界与下界。

需要界定计算所需要解决的问题，以及计算所需要的设备（模型）。

这一节先规定前者。回顾一些经典的算法或数学上的问题：给定带权重的图 G，求其中的最短路/其的最小生成树；提供矩阵 A, B，求其乘积 AB。这些问题都可以看作一个函数：给定输入，求输出。

与程序设计中的函数强调 implementation（即 How to compute the answer）相比，这里的函数更多具有数学意义，强调 specification（即 What should the answer be）。

接下来聚焦这些函数的输入，计算机无法理解图、矩阵这些概念，只能理解二进制串（binary string），也就是一串又一串的 0 和 1——于是要通过某些编码方式将这些元素编码为 01 串。先定义一个字符表（Alphabet）：$\Sigma = \{0, 1\}$，于是长度为 n 的二进制串的集合可表示为 $\Sigma^n = \Sigma \times \Sigma \times ... \times \Sigma = \{(a_1, a_2, ..., a_n) \mid a_i \in \Sigma\}$

特别规定 Σ^0 是长度为 0 的串的集合，这个串用 e 表示，即 $\Sigma^0 = \{e\}$

$\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$ 即为所有长度的二进制串集合。

♣

前缀（Prefix）

$x = a_1 a_2 ... a_n, y = b_1 b_2 ... b_n$ 的拼接（Concatenation）为 $xy = a_1 a_2 ... a_n b_1 b_2 ... b_n$
x 是 y 的一个前缀（Prefix），当对于某些 $z \in \Sigma^*, y = xz$
类似的可以定义后缀（Suffix），不再赘述

可以将 Σ 中的 0 和 1 换成任意字符，例如 26 字母，方框三角圆，以此组建你自己的 Alphabet！

♡

编码（Encoding）

有了最基础的元素（字符），将图、矩阵、等等等等计算函数的输入转化为字符串的过程，称为编码，即一个映射 $E : A \to \{0, 1\}^*$。

编码性质

显然，这个映射需要是单射（injective，在下文中会频繁表示为 one-to-one），即不同元素的映射结果（得到的字符串）必须是不同的。

例子

- 自然数 $n \in N$（parity(n) 是 n 对 2 取余的结果）：$NtS(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & \text{if } n=1 \\ NtS(\lfloor \frac{n}{2} \rfloor)\ \text{parity(n)} & \text{if } n>1 \end{cases}$
  - 亦即 n 的二进制表示
- 自然数对 $(a, b) \in N \times N$，自然的想法是 a 的编码拼接 b 的编码，但是会出现编码重复，并不是单射
  - 对于 1110，可以解释为 $(1, 6)$ 和 $(3, 2)$，这实质上是因为在计算机读取完前两个 1 时，并不知道它代表 3 还是一个其他数的前缀

在第二个例子的教训下，我们需要找到的编码映射是 prefix-free 的，即对于任何的 $x \neq x'$，都有 $E(x) \neq E(x')$。

接下来 myc 老师突然就这个 prefix-free 证明了一个寻找另一种编码的引理，感觉很突兀。

引理 1

假设已经存在一个 prefix-free 的编码 $E : A \to \{0,1\}^*$，那么对于编码 $\bar{E} : A^* \to \{0,1\}^*$（$A^* = \bigcup_{n \geq 0} A^n$，我理解为一个由任意长待映射元素序列组成的集合，接下来要找到对这些元素序列的编码），命 $\bar{E}(a_1 a_2 ... a_n) = \begin{cases} E(a_1)E(a_2)...E(a_n) & \text{if } n \geq 1 \\ e & \text{if } n=0 \end{cases}$，那么 $\bar{E}$ 是 one-to-one 的